Developing an Integrated System for Automated Picking and Sorting using an ABB Flexpicker Robot

A thesis presented in partial fulfillment of the

requirements of the degree of

Master of Engineering

In

Mechatronics

At

Massey University,

Auckland, New Zealand

Hongda Wu

2010

ABSTRACT

In the rapid development of flexible automation and the broad application of computer technology, industrial monitor software has played an integral role in all kinds of industrial areas. It allows operators to monitor and control a plant in real-time with feedback from any number of processes. Traditionally industrial monitor software exhibits low efficiency, lack of reliability, non-reconfigurable and does not support multi-communication protocols, as is required for the exchange of data from outside of the factory. (Fan, 2006) Configuration software is basically type of the industrial automation and process monitor and control application. It supports Human Machine Interface (HMI), Supervisory Control and Data Acquisition (SCADA) system, realizes interlink between low level device and upper management network. Nowadays, with the advent of Configuration Software, engineers can readily construct field control systems with minimal developmental time and cost while allowing the combination of a plethora of user requests and control.

The primary objective of this thesis is to develop a web base application with surveillance ability to realize remote control of an ABB IRB 340 Flexpicker robot through Siemens Programmable Logic Controller (PLC) system. The communication between the application and robot system is to be built using configuration software to link a number of third party devices through the inclusion of OLE for Process Control (OPC) techniques, graphical design editors, web navigators, and tag management. The thesis also introduces a vision system with trig-board design for object recognition and tracking.

Acknowledgements

First and foremost, I would like to offer my deepest gratitude to the mentors of this research: Dr. Johan Potgieter, who, with his guidance and tuition allowed for the completion of this dissertation.

For their invaluable advice and services, I would also like to thank Oliver Grant, James Chan, Jamie McIntyre, and Graeme Paulin.

As usual, the unconditional support of my family and loved ones is appreciated; as such, I would like to acknowledge my mother and father; my brother and sister and my partner. Their support, both direct and indirect, provided a bastion of confidence during times of difficulty.

For those who I have gained knowledge from indirectly, your work has provided a rich source of information that has only furthered my own abilities, and I thank you.

Lastly, I would like to thank the staff and lecturers of Massey University's School of Engineering and Advanced Technology at Albany for the interest shown in the project and their freely given advice.

Table of Contents

List of Figures

List of Table

Chapter 1 - Introduction

In the 21st century there is a huge demand in industry for new types of automation solutions, especially in the packaging industries as the market tends to be more matured. The traditional pick and pack operations are labor-intensive, often requiring workers to pack and sort products into boxes, trays or cover board for further packing. Naturally, using manpower for high speed and high repeatability operation often results in a bottleneck, thus affecting the yield. For a long period of time, manufacturers have been searching for a new way to reduce the cost of pick and place operations. To this end Flexible automation solutions offer considerable advantage and the ability to cope with rapid change of products and its packaging.

Figure 1-1 Manual operation picking and sorting line (Source: ABB Ltd)

However, in the early industry sectors, there were no communication solutions to narrow down the geographical dispersion between factories. Later on, industry communication became another part of the basic technology automation process and has been integrated into all aspects of social development, promoting the introduction of industrial automation and control. In more recent years, industrial automation has become even more flexible, utilizing intelligent real-time control along with a wide variety of open and closed proprietary communication protocols. (Mahn, 2005) Since industrial equipment tend to be located behind firewalls, engineers are envisaging a way to integrate internet protocol into industry

automation communication through firewalls and proxies allowing a greater flexibility in working hours and location. The wide application of Internet/Intranet has brought about new possibilities in modern enterprise production monitoring that is easy to deploy, install, debug, maintain, increasing the interoperability between different systems and equipment.

With the rapid development of network technology, the Internet has become the main information carrier of the modern age, and it is widely applied in industrial bus-like protocols such as PROFIBUS. The development of Industrial internet technologies change the structure of the control system so that the direction of development towards network-based. One hand, the field bus technology development starts from the bottom of industrial equipment, and gradually extended to the network; On the other hand, the computer network start from top-level Internet penetration downward until the bottom level equipment communications. Through the field bus network and the Internet to completely interconnect computers that spread in the field in order to achieve resource sharing, collaborative work, remote monitoring and centralized management.

Speaking from the field of industrial control, more and more factories and production lines are geographically dispersed. Internet is the only solution to gather production information and plant control or to realize control network status monitoring on different factories and production lines, especially for some large multinational corporations. Therefore, the combination of control network and the Internet will become a new developing trend in control area.

Minimizing the geographical dispersion between factories through internet communications has provided motivation for this dissertation; as such, the research has aimed at developing a remote control web application integrated with a SCADA system to provide visualization of a real life automated production process. The research is to focus on the use of IRB 340

Flexpicker robot with PickMaster software and a Siemens WinCC platform that enables the control of production processes via a web browser.

1.1. Thesis Layout

The overall structure of the report is thus organized as follows: chapter one discuss the development of flexible automation, geographical dispersion control problems and the purpose of the research. Chapter 2 continues with a brief literature review on remote access control and SCADA systems with some case studies, in addition to the study of Delta robot technology. In chapter 3, the structure of remote system architecture and the design of remote control software will be demonstrated. The chapter also includes a detail description of the WinCC configuration software. Chapter 4 basically introduces the development of the automatic pick and place application with aid of machine vision system and achieves remote control of the IRB 340 robot through PLC. This section involves studies of the robot mechanical structure, PickMaster control program and the development of the machine vision system, hardware design. Chapter 5 will validate the functionality and performance of the system by conducting a serial of testing for each individual component as well as the combined system. Chapter 6, Conclusion, discusses the research's contribution, implications and identifies areas where further improvement could be done.

1.2. Project Objective

The following six major objectives of the research have been identified:

- Develop a PickMaster program for pick/place tasks and PLC program for remote control.
- Setup the connection between the Flexpicker system and PLC module using ADVANTECH PCI-1750 digital input/output; Configure the PickMaster Remote

Integrate Service (RIS) to define the ports for each command and interfacing with PCI-1750.

- Develop and design a vision system for object tracking and a trigger strobe for triggering the robot.
- Establish the communication between PLC and WinCC system using OPC technique.
- Develop and design a graphical user interface SCADA application using WinCC configuration software and allows specific user to perform remote control through Internet Explorer (IE) browser.
- Setup a surveillance server and broadcast video to the web.

Chapter 2 - Literature Review

2.1. Remote Access Control via Internet

With the rapid development of network technology, Internet/Intranet has become the main information carriers over the last decade, and had been widespread used in the automation processes using different protocols. The popularity of Internet technology provides many application fields for remote control as it use basic protocols for communication between components of automation systems and use of the Web browser as universal control interface for automated devices, machines and plants.

In early 1994, University of Southern California's Mercury project allowed users to control a robot via the Internet to simulate the activities of excavations. In 1995, University of Southern California's TeleGarden project allowed Web users to manage a ADEPT robot to manage a small garden. In 1997, University of Western Australia connected a ASIA IRb26 robot to the Internet so that user could manipulate it to crawl and move some pieces of wood on the table to build various model structures. Also in recent years, internet-based robot remote operating systems are getting more and more used. Generally speaking control method can be divided into Web-based and non-web-based. (Chou Wusheng , Wang Tianmiao, 2003)

Nowadays, due to the global market place, internet-based communication becomes more and more important such as (Prof. Dr.-Ing.Gruhler Gerhard, 2003):

- More reliable communication channels

- Use of PC-based standard technologies and software like Internet Protocol Suite (commonly known as TCP/IP)-stack, web servers, browsers, Java, Extensible Markup Language (XML) etc.

- Significant cost reduction for development, equipment and operation

- Less effort in particular on the client's side: usually the client just needs a standard web browser

- Worldwide availability of internet access at low costs

Typical remote access applications are large:

- Remote production and machine data collection

- Remote system monitoring and error detection

- Customer support for programming and running the system, help for bug fixing

- Software/firmware updates via internet

- Better support for maintenance staff

- Sometimes even avoiding service trips

- Direct system control by remote operator interface

General speaking, there are two type of internet based remote control system: 1) TCP/IP based protocol which is a basic internet communication technique. It uses client/server communication type to exchange data from one point in the network to another. 2) Hypertext Transfer Protocol (HTTP) remote control, which is implemented base on HTML page with a Java applet and visualization using Virtual Reality Modelling Language (VRML) language. All the contents are stored in the remote server and user can access and download the contents through the web browser. (Quanyu Wang; Siyin Liu; Zhe Wang, 2006)

With respect to internet technology, remote control system can facilitate the development of a real time monitoring and control application, capable of realizing data exchange between systems in different location; as such, the purpose of this literature review has been to investigate the feasibility of operating a Flexpicker system through the web.

2.2. SCADA and Web-based SCADA Systems

The SCADA system is a computer-based production process monitoring and control automation system that plays an integral role in modern industry. It can communicate to lower level hardware controllers, allowing a factory operator to control the plant and provides feedback about the current state of the factory processes, including any alarms.

SCADA is type of a software application specially designed to work on computers in the production control that gets data about a system (independent controllers, programmable robots, etc.) and controlling the process from HMI device or computer. A SCADA system performs four functions:

- Data acquisition
- Networked data communication
- Data presentation
- Control

The SCADA system also provides a way for fine-tuning your knowledge of your systems. By simply placing sensors at each connection within the process, we can monitor and control the process more detailed with real time data feedback. These functions are performed by four kinds of SCADA components: Sensors, Remote telemetry units, SCADA master units and communications network. The main communication is made by means of special buses or Local Area Network (LAN) networks. All this is executed normally in real time, and is designed to give to the plant operator the possibility of supervising and controlling these processes.

An eagle-eye view of every event while it is happening is important in decision-making for very complex manufacturing processes. In many cases that large modern companies are

usually being so geographically diverse due to their individual business departments, thus, network communication has played an important role in real time data exchanging and the high speed of the internet connection allow Web-based SCADA system to be readily accessible at anytime. (Berry, 2008) There are several different Web-based architectures currently available for an Internet based SCADA system. The systems are unique in design and functionality. Each has its own set of advantages and disadvantages and these must be considered according to the application. In general, Internet and Web-based SCADA systems have advantages over traditional SCADA systems for reasons of (e-scada.com 2002):

- Wide Area Connectivity
- Routable connection rather than direct connection
- Parallel Polling
- Redundancy and Hot Standby
- Large addressing range
- Convergence of IT and Automation and Monitoring Networks
- Standardization

In turn of the SCADA software, the modules that allow the activities of acquisition, supervision and control are the following ones:

- Configuration: it allows the user to define the work based on the SCADA application
- Graphical interface of the operator: it provides to the operator the ability to create the process pictures of the plant and control of the systems through synoptic graphs.
- Module of process: it executes the pre-programmed actions of control from the present values of the read variables.
- Management and data file: it is in charge of the storage and ordered processing of the data, so that another application or device can have access to them.

- Communications: it is in charge of the transference of information between the plant and the hardware architecture that supports the SCADA.

2.3. Delta Robot Technology

"The idea of parallel mechanisms resorting to a non-rigid moving platform which includes passive joints and dedicated to Scam motions has been introduced recently and a few academic prototypes have already demonstrated the effectiveness of this principle." (Krut, S., Nabat, V., Company, O., Pierrot, F., 2004)

In the early 80's, Reymond Clavel, professor at École Polytechnique Fédérale de Lausanne in Switzerland, created the Delta concept by using parallelograms to construct the parallel robot with three translational and one rotational degree of freedom. Below the original technical drawing from U.S. Patent 4,976,582 is shown. With sponsorship from ABB, Reymond Clavel presents his doctoral thesis 'Conception d'un robot parallèle rapide à 4 degrés de liberté' in 1991 and receives the golden robot award in 1999 for his creative work. There is no doubt that his creation was truly original and brilliant, which it is the most successful parallel robot design at the turn of the century. (Bonev, 2001)

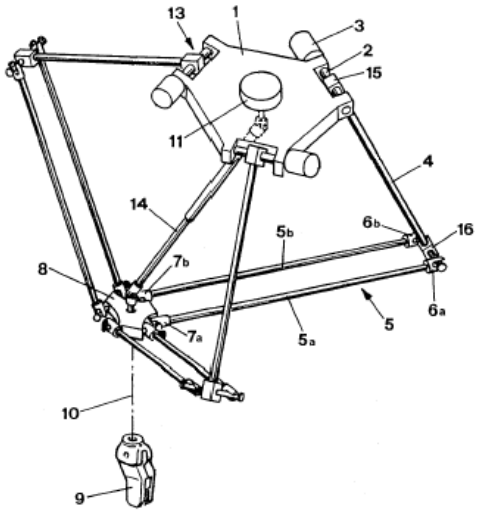


Figure 2-1 Schematic of the Delta robot

In 1998, ABB launched its Delta robot – IRB 340 FlexPicker (Figure 2-2). The design has an acceleration of up to 10g's, and 2 kg of handling using integrated vacuum system. It claims to be the fastest robot in the world with the help of optimized drive-chains and ABB's patented QuickMoveTM functions, up to 180 picks per minute. The FlexPicker also has a positional repeatability of 0.1mm and an angular repeatability of 0.48. The IRB340 was launched by ABB in 1998, targeting the food, pharmaceutical and electronics industries. Current in the market, there are three versions of the robot available: Standard version, Wash Down and Stainless Wash Down. Standard version is to be used in dry applications. Wash Down versions are designed to be cleaned and disinfected, and therefore useful in wet applications. IRB 340 is designed for use together with the Cognex vision system, the S4Cplus controller and robot control Software BaseWare OS. BaseWare OS supports every aspect of the robot system, such as motion control, development and execution of application programs and communication etc.

Figure 2-2 ABB Flexpicker IRB 340 (Source: ABB Ltd)

2.3.1. The Design

The theory of parallelograms is used for the Delta parallel robot design. The parallel robots

are spatial mechanical structures that consist of kinematic closed chains. According to Staicu & Carp-Ciocardia, every parallel manipulator usually has two platforms. One is attached to the base frame and the other for arbitrary motions in the workspace. The end-effector is connected by three arms linked in parallel via revolute joints. The fourth leg is used to transmit rotary motion from base frame to an end-effector mounted on the mobile platform. Actuators are placed in a non-moving base frame and all the moving parts are made of lightweight materials such carbon fibers and plastics. The lightweight arms of the robot can accelerate of up to 50 G in experimental environments and 12 G in industrial applications. Therefore, it is ideal to pick and pack lightweight objects such as candy. Also, the standard Delta robot arm can reach up to 1 m in diameter and 0.2 m high. (Staicu St. & Carp-Ciocardia D.C, 2003)

2.3.2. Delta Robot on the Market

The Delta robot has been existed in the market for a long period of time. It all began in 1983 when Marc-Olivier and Pascal Demaurex set up the company Demaurex. They started focusing on the packaging industry by commercializing the parallel robot into production after they purchased the license in 1987. The success of the Delta robot in packaging lines helps them establish themselves in this new market. The model of the robot was also improved or modified due to the customers' requirement and updated technology. (Bonev, 2001)

Figure 2-3 Demaurex's Line-Placer installation for the packaging of pretzels

Before Demaurex bought the patent in 1996, many companies like Aid, Deemed, Elekta IGS, and Medtronic purchased the license for the Delta robot in different specification and gradually manufactured their own robots. Also in 1999, ABB launched its Delta robot, IRB 340 FlexPicker into this lucrative market to compete with Demaurex, who has been a sole player for 15 years in the market. ABB's strong market shares threaten Demaurex and forced the company started to look for partners. By the end of 1999, Demaurex became acquired by the Swiss Group SIG. (Bonev, 2001)

The Delta robot technology has also attracted the academic studies. Many Universities in the world have came out their own Delta robots design or modified models.

2.3.3. Flexpicker System

Electrical and Automation Group ABB launched FlexPick, PickMaster software and combined with S4Cplus robot controller to achieve production efficiency and flexible packing. ABB release latest second generation IRB340, which is the fastest robot in the world with a speed of up to 150 picks per minute in the assembly line. The outstanding performance advances from the use of carbon fiber material which allows a high strength to weight ratio

for the moving parts, and the “Delta” design by confined all the heavy gears and motors to the fix base. “The three linked parallelograms of the Delta structure make the robot stiff and highly controlled in its movements, and produce a positional repeatability of 0.1mm and an angular repeatability of 0.48° . It has a cylindrical workspace with a vertical movement range of 300mm and a horizontal range within a circle of 1,130mm diameter. Its end effecter has limitless rotation capability, and is equipped with an integrated vacuum system. The robot is available in wash down and stainless steel wash-down forms and operates in temperatures down to 0 ℃, which is useful in chilled food handling environments.” (Connolly, 2007)

Chapter 3 - Web-based SCADA Application Design

3.1. System Design Overview

3.1.1. Control Architecture

There are now numerous architectures available for implementing a web-based robotic system such as Communication Gateway Interface (CGI), Java technology, HTTP and TCP/IP protocols for real-time communication that replace traditional field-bus systems. Figure 3-1 below introduced a flexible and extendable approach of using the central server architecture by (A. Sayouti, F. Qrichi Aniba, H. Medromi, 2008). As explained, all the clients and services are connected to a central web server, and only the IP address of the web server is needed in order to communicate with each other through the web server. By applying this architecture, all the services in this project such as image service, robot control service, web service and surveillance service can be put together in one computer. Another advantage of this architecture is that it is easily upgradable through adding more servers for the services to realized multi-robot control.

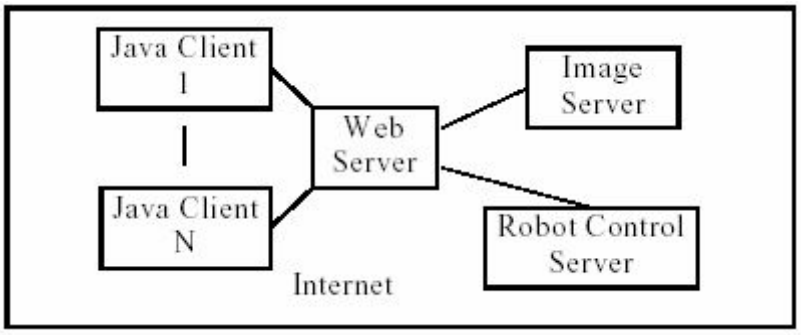


Figure 3-1 Control Architecture

The application of Internet technology in industrial sector gets wider and cheaper in implementation, which permits the use of this technology in new fields such as remote teaching, remote manufacturing, remote maintenance etc. Some well known works have been

carried out based on the very different remote access scenarios, varying methods have been developed which provide system access at different levels. Examples hereof are described in this thesis and may be accessed worldwide anytime by everybody at http://130.123.254.55. The control system does not require the implementation of any specific software pre-installed on the client′s computer, only a web browser is needed.

The remote control server is the main integrated automation component of the project, thus enabling uniform access to automation plants by means of a Web browser. The Web-based extension solution provides wider mobility to existing Flexpicker systems for operating, monitoring as well as for remote service purposes.

3.1.2. Proposed control architecture

The concept of high-level remote access is introduced in this system shown in Figure 3-2, as it provides easy access to real-time data and for use by any clients' PC. The system is usually designed using graphical user interfaces, thus allowing remote clients to access only certain data or specific control functions. (Prof. Dr.-Ing.Gruhler Gerhard, 2003)

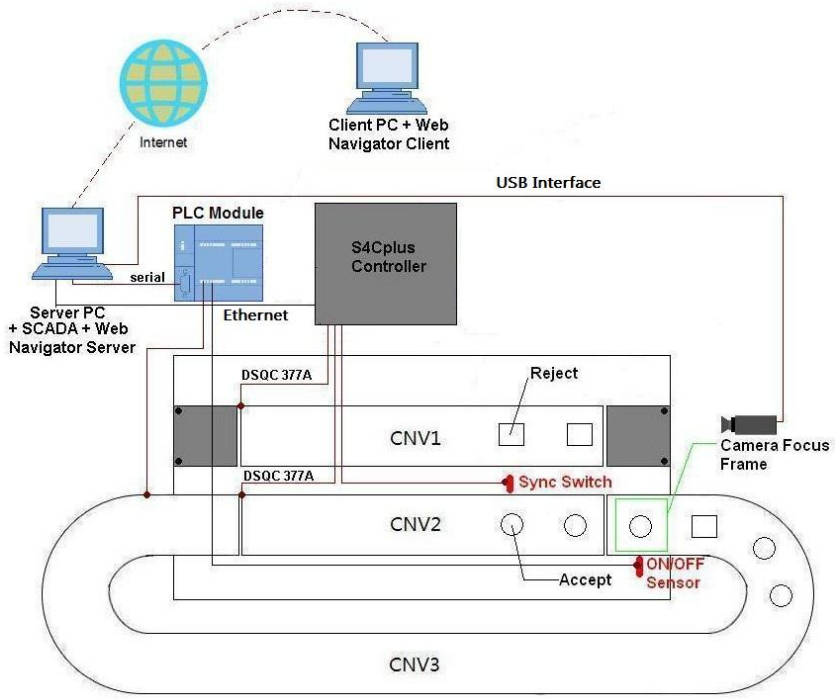


Figure 3-2 System Overview

The thesis proposes hybrid control architecture for automation system, including a remote control agent, a central server and the reactive part. It is made up of three parts, each using distinct method to solve problems. The remote control agent is developed and run based on Siemens S7-200 PLC software and hardware. The human machine interface program is designed using SIMATIC WinCC which runs under Windows operating system. The central server also contains applications such as camera broadcasting server, vision software and robot control software. The reactive part is basically the Flexpicker system for event reaction. Figure 3-1 shows a high-level web-interface from a client's PC to an ABB IRB340 Flexpicker system in Massey University. Each of the sub-systems is interconnected through different protocols. The PLC controller functions as a communication bridge between the server PC and Flexpicker system. PPI/PC serial cable establishes the communication for PLC and PC while the Digital I/O board used for receiving commands from the PLC to the Pickmaster robot program. A camera was added as machine vision to achieve object tracking and detection for typical robot applications like pick and place. The algorithm for object

tracking and recognition was developed. Another reason for having remote access is that a user can monitor the process in near real-time on a computer. A webcam provides an internet video link to the remote user and it runs independently from the built-in stand-alone broadcasting server.

3.2. Process Visualization with SIMATIC Wincc Configuration Software

In most control systems the requisites for an adequate control system is simply not enough. (Zhou, 2004) In many such cases is also necessary to monitor the configuration interface of the system. The capacity of human-computer interaction can therefore be increased by system monitoring, easing the load of system operational management as is offered by real time system monitoring systems.

Generally speaking there are three ways of monitoring a control system like that of the S7-200 series PLC: configuration software monitoring, third-party monitoring software and the touch-screen monitor. Configuration Software such as WinCC has a number of powerful functions, good flexibility and high reliability. However, the software price of such systems can be prohibitive, and communication between the WinCC and S7-200 series PLC, as utilized within the scope of this project, is not, as yet, supported.

SIMATIC WinCC provides a complete basic system for operator control and monitoring. It supports different bus system and provides a number of editors and interfaces that allow you to create highly efficient configurations for specific application. All relevant configuration data is stored in a WinCC project. The WinCC basic system already contains all the editors and tools to perform the different configuration tasks. (Siemens AG, 2008)

WinCC also provides an add-on option called WebNavigator, which supports full-fledged operator control and monitoring via the web. A WinCC web server can be installed on a

WinCC server or client computer, thus a web client connected to the web server with authentication can gain access to the projects in a plant from anywhere in the world.

The whole design of the program is focused on establishing software hardware communication and graphical programming using the integrated graphical editor. The main functions of the WinCC software have been used to achieve the following tasks:

- OPC server, a driver for fieldbus system or an embedded controller,
- Monitoring, i.e. visualization of the process status and history,
- Allowing operator control, like system ON/OFF, switching subsystems,
- Archiving of process variables and alarms,
- Allowing Web navigation for remote control, and
- Web surveillance service

3.2.1. Advantages of WinCC Configuration Software

❖ Expandability

The flexibility and expandability of WinCC simplifies integration in existing automation systems or a tele-controlled concept, which guarantees the future investment security. SIMATIC WinCC is expandable by industries due to its open standardized interfaces; industries can have technology specific options and add-ons for their applications.

❖ Efficiency

WinCC is a 32-bit applications, its multitasking feature ensured that rapid response to the process event, and also provides a variety of prevent data loss protection. WinCC is using object-oriented programming technology, and all the engineering tools are integrated with configuration properties, as well as provided all the basic functions for process visualization

and operation.

❖ Security and traceability

WinCC provides with a central, plant-wide user administration system that is integrated into Windows to prevent unauthorized access while message system records process messages and local events, saves them in message archives.

❖ Openness and integration

WinCC Visual Basic Script also offers a dynamic graphical runtime environment by means of actions. For example, the ability to start programs, creates directories, and control applications such as Microsoft Excel, over an intranet or the Internet. WinCC also offers ANSI C language allowing access the entire runtime environment.

3.2.2. Graphical Design of Remote Control Application

3.2.2.1. Graphic Designer Editor

Configuration and Runtime are the two main components in Graphics System. The Graphics Design editor is the configuration component that provides graphical drawings while the Graphics Runtime is the runtime component of the Graphics System displaying the screens in runtime and administering all inputs and outputs. (Siemens AG, 2008) Figure 3-3 presents a broad view of the intercommunications link between the screens and automation system. As can be seen, the process starts from the Input Device (in this case a mouse/keyboard). The signal follows down the arrow, activates the automation system and sends the status back to the process screen.

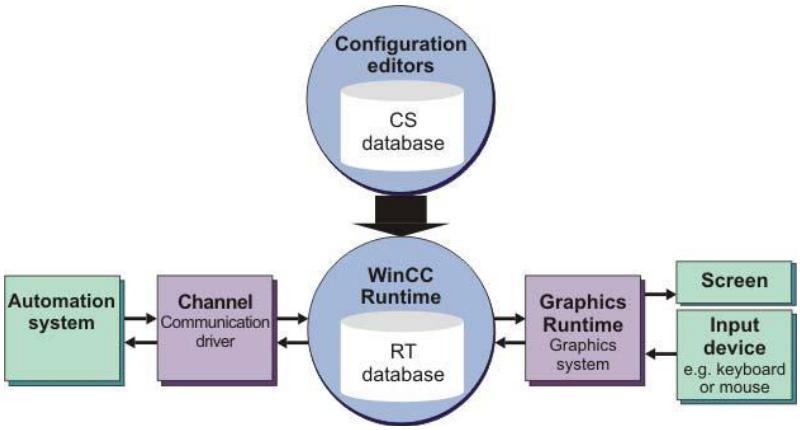


Figure 3-3 Screens and Automation Communication

The friendly user interface is an important criterion to determine software good or bad. WinCC graphical interface provides a powerful graphic editing tool, which can monitor the whole control plant using static display picture.

3.2.2.2. Creating Process Pictures

The process screens are the main elements of a project. They represent a process and allow the operation and observation of this process. Each process screen is made up of several objects:

- Statistic objects that remain unchanged in runtime.
- Dynamic objects that change in accordance with the individual process values. A bar is an example of a dynamic object. The length of the bar will depend on the current temperature value.
- Controllable objects that allow individuals to influence the process during run time. These include buttons, sliders, or I/O fields used for entering certain process parameters (input/output field).

The process pictures that have been used for the visualization and operation of this project have been created with the Graphics Designer editor. The editor is comprised of several process screens. Each process screen shows a different process or displays special process data. Process pictures are composed of individual picture objects ranging from simple circle, square, and polygon to self-contained, preconfigured smart objects and controls with a process connection, i.e., an interface for the input and output of process values.

Figure 3-4 Starting GUI

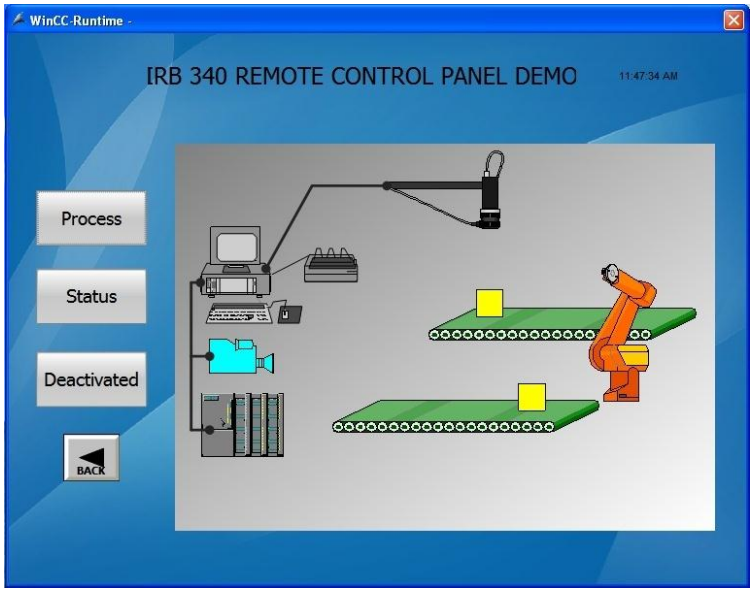


Figure 3-5 Process Picture

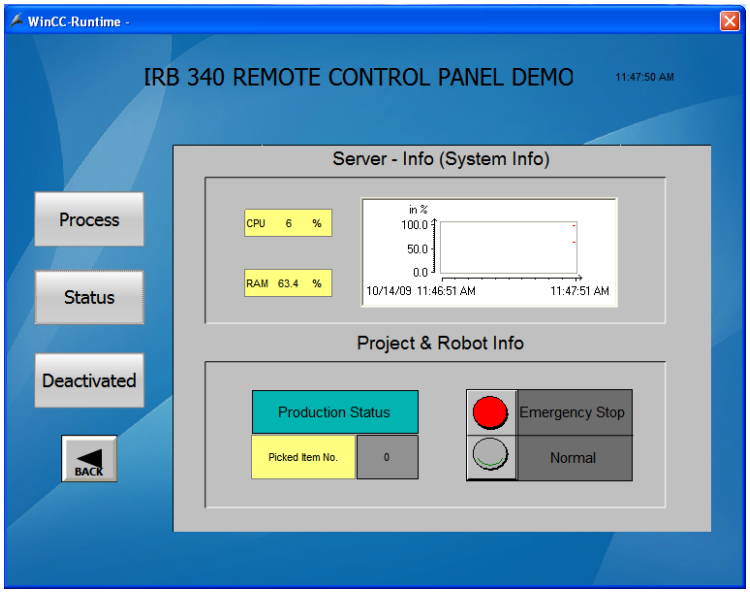


Figure 3-6 Status Picture

Figure 3-4 to 3-6 demonstrates the application used for remote control of the Flexpicker robot as designed with the use of WinCC Graphical Designer. This application also provides control buttons for switching monitor screen and human-controlled hardware devices running such as program start/stop, robot start/stop, etc. The first frame is used as the starting picture when the WinCC server or Touch Panel is initially booted up. The remained of the graphical interface having been designed to fully replicate the physical control module, with each button action linked to the S7-200 PLC I/O port through the use of the WinCC Tag Management and PC Access.

The video steam window within the same interface connects the surveillance camera to stream live video data. A graphical representation of the overall process is provided under the Process button, and, along with the Status button, provides server status information such as CPU, RAM and HDD usages.

3.2.2.3. Tag Assignment and List of Variable

The general procedure for creating ‘button actions’ as utilized within the graphical interface

has been carried out through the use of the "Tag management" editor which is used to configure the communication channels between the software and automation system and consists of the following components:

- One channel with channel units
- One connection
- One process tag

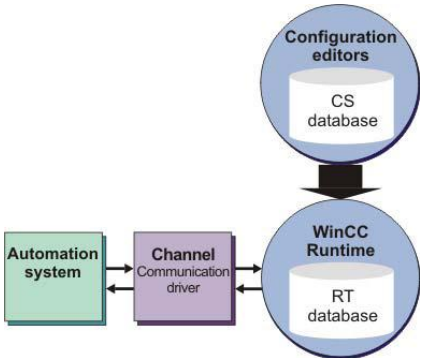


Figure 3-7 Work Flow Chart from WinCC Manual

The channels play the role of communication drivers enabling the supply of process values from the automation system to the process tags. WinCC provides a variety of channels to connect different automation systems. However, each of the channel units is dedicated to a different communication network. The channel unit is then used to access to a certain type of automation system.

The tags system can represent either real values or internal values. The internal values are calculated or simulated within WinCC. The connecting links for the exchange of data between WinCC and the automation system is provided by external tags. External tags are therefore referred to as process tags because each external tag, in WinCC, corresponds to a certain process value in the memory of the connected automation system. In Runtime, the process values of the process tags are determined and entered by WinCC. In Figure 3-7 the values for the process tags can be determined and the values are transferred to the automation

system for process control via the stipulated channel.

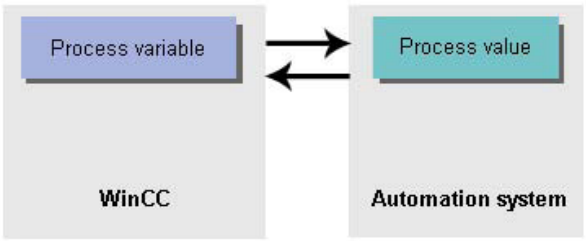


Figure 3-1 Tag System from WinCC Manual

Table 3-1 shows the tag assignment for communication between WinCC and S7-200 PLC I/O ports with the binary type value assigned in the system.

Name	Type	Parameter	
I 0	Binary Tag	I 0.0	**INPUT**
I 1	Binary Tag	I 0.1	
I 2	Binary Tag	I 0.2	
I 3	Binary Tag	I 0.3	
I 4	Binary Tag	I 0.4	
Q 1	Binary Tag	Q 0.2	**OUTPUT**
Q 2	Binary Tag	Q 0.4	
Q 3	Binary Tag	Q 0.5	
Q 4	Binary Tag	Q 1.0	
Q 5	Binary Tag	Q 1.1	
Q 6	Binary Tag	Q 1.2	
Q 7	Binary Tag	Q 1.3	
Q 8	Binary Tag	Q 1.4	
L1	Binary Tag	M 0.0	**LATCH**
L2	Binary Tag	M 0.1	
L3	Binary Tag	M 0.3	
L4	Binary Tag	M 0.4	
L5	Binary Tag	M 0.5	

Table 3-1 Tag Assignment for PLC I/O

3.2.2.4. Archiving and displaying values

WinCC allows user to save the process values in the process value archive. The archive can, for example, be used at a later point in time to display and evaluate the temporal development of the process values.

The Archive System for process values is made up of a configuration and a Runtime component: the *Tag Logging* editor and *Tag Logging Runtime*. The Tag Logging is mainly used to configure process value archives, compressed archives, and defining process values to be archived. Process value archiving is controlled by cycles and events. The acquisition and archiving cycles enable continuous acquisition and archiving of process values. Furthermore, process value archiving may also be triggered or ended by events. Tag Logging Runtime is primarily used to write process values into the process value archive and reading archived process values from the process value archive

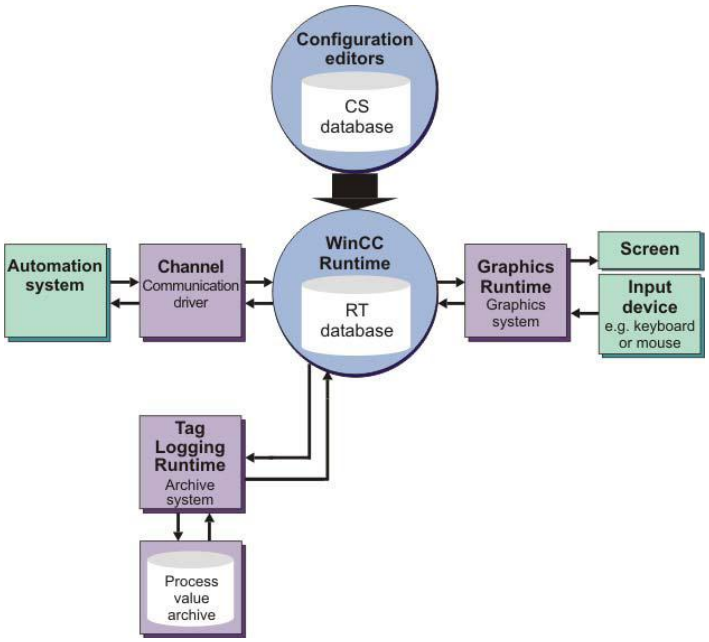


Figure 3-8 Components of the Archiving System

3.3. SIMATIC WinCC and PLC Communication

There are several communication methods for PLC and Configuration software, the differences are primarily in terms of their implementation and costs. SIMATIC WinCC process monitoring software is using the latest 32-bit technology which has good openness and flexibility. Whether single-user system, or a redundant multi-server / multi-user system, WinCC is a better choice. Through ActiveX, OPC, SQL and other standard interfaces, WinCC can easily communicate with other software. Either PPI or Profibus can be used as the communication protocol between WinCC and S7-200 series PLC. (SIEMENS)

3.3.1. Types of Communication

3.3.1.1. *WinCC and S7-200 PLC via Profibus Protocol*

(1) Software and Hardware Requirement:

- A PC with minimum WIN98 OS；
- S7-200 series PLC
- CP5412 Board or other similar boards like CP5611，CP5613
- EM277 Profibus DP Module
- Profibus cable and connector
- CP5412 Drivers Installed
- WinCC 4.0 or above installed
- COM Profibus installed

(2) Advantage and Disadvantage of this method:

The advantage of this method is that high data transmission speed with real-time and easy expansion. On the other hand the transfer data area is limited (maximum 64 bytes). High cost of hardware and software is another issue as the needs of CP5412, EM277 Profibus-DP,

Profibus bus, other hardware, and also Com Profibus software. However, such requirements are suitable when applied to high-speed data communications and the real-time requirements of high speed systems.

3.3.1.2. *WinCC and S7-200 PLC via PPI Protocol*

PPI protocol is the commonly used communication protocol in Siemens S7-200 series PLC, but this protocol is not integrated into the WinCC system, which means that WinCC can not directly monitor the control system setup by S7-200 PLC. Siemens launched S7-200 PC-Access to address the host computer monitoring S7-200 series PLC. Therefore, WinCC is very easy to establish communication with the S7-200 series PLC via the software. In addition, other OPC software is also capable to communicate with S7-200, for example KEPServerEx, which is using WinCC as an OPC client to read and write within the S7-200 data area.

(1) Software and Hardware Requirement:

- A PC with minimum WIN98 OS；
- S7-200 series PLC
- PC/PPI Cable
- WinCC 4.0 or above installed
- S7-200 PC-Access installed

(2) Advantage and Disadvantage of this method:

The advantage of this method is easy to setup and small investment; it can read and write to all memory areas of the S7-200 PLC. The disadvantages are relatively slow speed of communication, and the corresponding need for OPC software licensing as well as limited system expansion. This type of protocol is suitable for applications with low-speed, non-real-

time and limited investment funds systems.

3.3.2. Setting up the Communication

PPI communication is selected in this project since it is more realistic and economical method compared to Profibus protocol, with the purpose of only demonstrating the applicable of the control method, therefore corresponding configuration procedures are taken to carry out the communication.

The first step is to load the S7-200 program symbol table into the PC Access and then configures the PG/PC interface parameter in “Computing” under the menu bar, then selects PC/PPI Cable as the interface parameters. In the WinCC Variable Manager, add a new driver called OPC.CHN follow by creating a new connection in the OPC GROUP. Next, open the Properties window, select the OPC Group Setting, OPC server name “OPCServer.MicroComputing”. Finally in the new connections tag explorer, add a new variable, the variable Item Name must correspond to the variable names that is used to monitor in S7-200. For example: Item Name of M0.0.

Up to this step, a communication link is successfully created. All the variable symbol status used in S7-200 can now be accessed through the WinCC Tag Management and Process Picture. The figure 3-9 below shows the imported symbol table from the PLC program and quality of the connection status in PC Access.

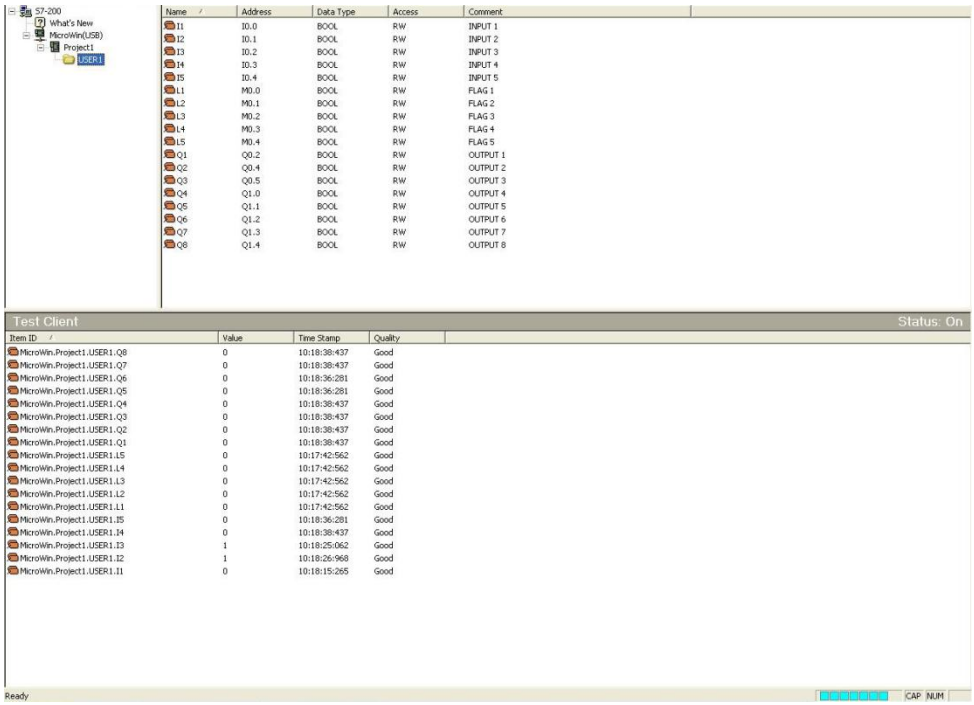


Figure 3-9 Communication Established in PC Access

3.4. WinCC Web Navigator based Remote Plant Control

WinCC Web Navigator remote control is based on the rapid development of Internet network. Such high-level web remote control system composed of the use of computer automatic control technology, network communication technology, simulation technology, multimedia technology, network technology, database technology and Web technology etc. that provided a solution to industries that are geographically dispersed.

3.4.1. Remote Plant Software Architecture Design

The PLC remote plant control is based on WinCC Web Navigator which consists of the following components: the Web Server, Client Platform and SQL Database Design. The structure of each component is shown in 3-10:

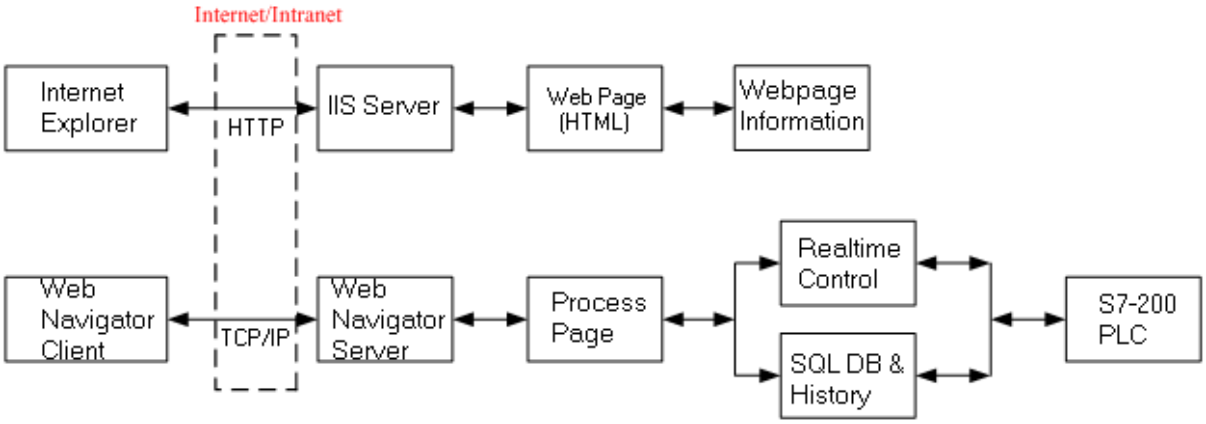


Figure 3-10 Remote control structure based on WinCC Web Navigator

(1) IIS (Internet Information Service) server is a socket that links up the web clients to the web navigator server. In this project, the IIS service in Windows is used to create the web server. The web server is running a web site that generated by the WinCC Web Navigator for users to browse information, logon and control panel operation.

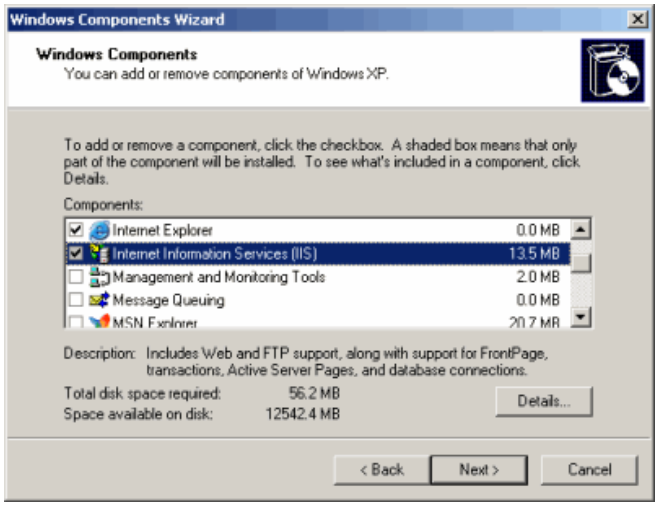


Figure 3-11 Installing the IIS Service

(2) SQL database design mainly includes two functions: stores client and public information; provides devices alarm message and client browsing history etc.

(3) Web Navigator Server is used to receive commands and parameters from the remote end and provides users with process monitor screen to achieve on-site object control.

WinCC Web Navigator server will transfer all parameters to PLC through network.

3.4.2. WinCC Web Navigator Remote Access Method

The WinCC Web Navigator is based on standard HTTP protocol, and supports all conventional security mechanisms. WinCC Web Navigator allows users to remotely monitor and operate devices through Internet. Web Navigator includes the Web Navigator Server component and Web Navigator Client component. The monitor screen on the client's PC is just the same as the one running on WinCC server, and the running projects can be monitored at any location with LAN or WAN connection. (Siemens AG, 2008)

Configuration of Web Navigator under the following steps:

(1) Web Navigator Server Configuration Project

On the server site, create a new WinCC project, and then open the Web Configurator windows with the following setting: assign the website with a name "IRB 340" and an IP address "130.123.254.55:80". We can verify the connection status of the website through the IIS.

(2) Publish Process Picture

In the WinCC project, open the Web View Publisher dialog box, and select the remote publish path; then choose pictures that want to publish, as shown in Figure 3-11. Next, select the C function and images to complete the web server configuration.

Figure 3-12 Publishing Wizard

(3) User Management Configuration

Within the WinCC project, open the User Administrator dialog box and create a new user. Once created, select the specific user in the user manager navigation window to define the corresponding permission, initial screen and language.

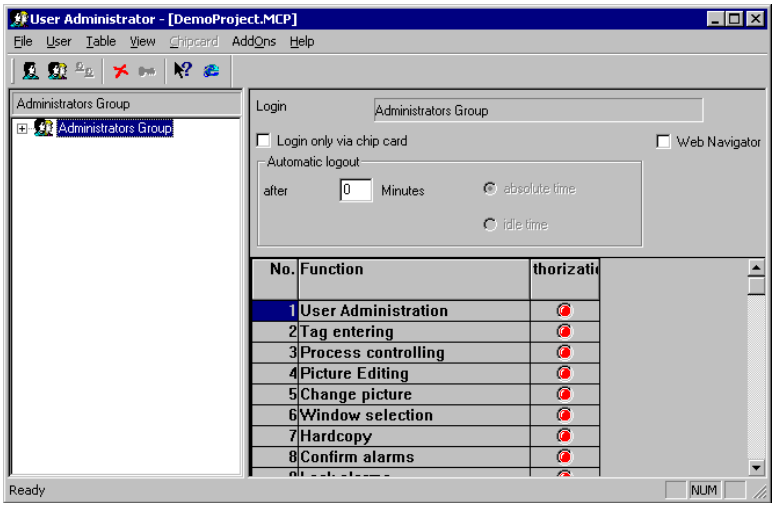


Figure 3-13 User Administrator

(4) Internet Explorer setting

Open the IE browser, within the “Internet Options” and “Security” property page, enables the “Script ActiveX controls Marked Safe For Scripting” and “Down Load Signed

ActiveX Controls".

(5) Install Web Navigator Client

Enter the Web Navigator Server address into the IE address bar, if it is the first time visit Web Navigator Server, the IE will prompt to download and install Web Navigator Client. Once the installation complete, the client PC will be connected to the running projects in the server and the defined initial screen will be displayed.

3.4.3. PLC Remote Plant Control Website Design

The remote plant control system website is a remote plant control system for data service centre that responsible for providing different services to users with different administration right. Website features include news release, users login and registration functions, user managements, plant information, remote control function.

The Home page design based on HTML, ASP and JSP is mainly used to provide direct on-site information such as surveillance screen as shown in figure 3-12. The video stream window is pre-defined with the fixed address http://130.123.254.55/camera.html. Thus, every time the home page is loaded, the video stream window connects to the camera server for transferring live images.

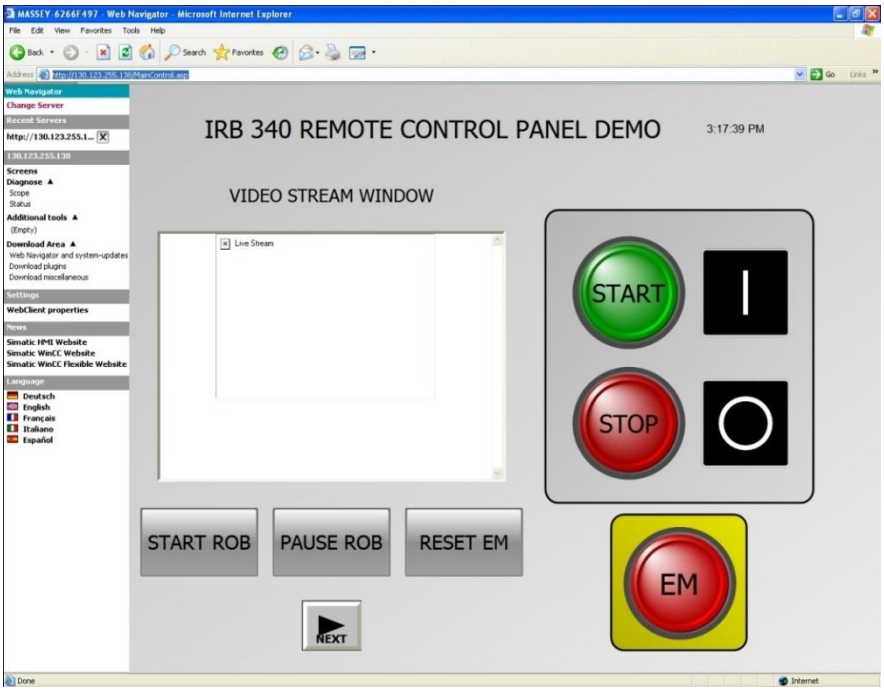


Figure 3-14 Homepage

Once the users logged in, they can directly control the Flxipicker robot by clicking the start button on the control panel. The image display applet shows the visual feedback in a continuous jpeg image. The Web Navigator panel on the left is designed to provide customization options for the users to optimize based on their personal predilection such as language. It also provides the server diagnostic tools and necessary software for downloads. This user interface allows users to undergo a distance operation with the opportunity to test their program on line. Once again, the humanization design and interface match the high-level remote access concepts that provides easy operation environment for non-expert users.

3.5. Web Surveillance System

This section of the thesis focuses on the use of a USB webcam utilizing WebcamXP Free (a program that streams webcam video). WebcamXP is a powerful webcams and network cameras monitoring, recording and streaming software for private and professional use. It offers unique features and unequaled ease of use to let multiple video sources be controlled

on the same computer. It is the ideal tool to secure goods and keep an eye on them remotely over the internet. This method is not dependent on a web streaming website and is also more bandwidth efficient for secure remote communication of video information between multiple clients of a network, as shown in Figure 3-13. The main interest purpose here is its ability to provide overarching support of a wide range of camera device standards while at the same time forming a web server capable of integrating to the software on the local computer. This enables the clients connecting to the webcamXP server to receive the video stream using a JavaScript based technology with an advanced smoothing effect between consecutive frames. This client-server type application is based on TCP/IP protocol to provide reliable connection for streaming data. In this project, a method of streaming over HTTP with JavaScript was chosen since our web application is built over HTTP and also HTTP requires no permission for any normal internet communication to occur, thus overcoming problems such as firewall blocking. (Darkwet Network/Moonware Studios)

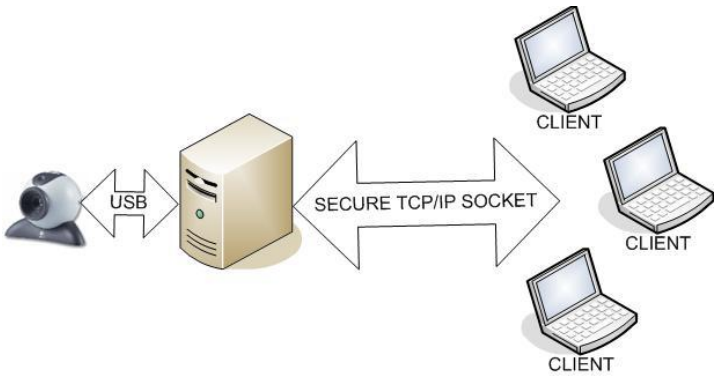


Figure 3-15: Web Surveillance Architecture.

A video is a sequence of framed images that are displayed at a rate one after another. If we had all frames of a video clip in our browser, we could display them one after another at a frame rate. Below are the steps that webcamXP server carried out to achieve live broadcasting:

Step 1: Getting the frames, frame rate and other necessary information from a video file or a live stream

Step 2: Transport our frames over HTTP to the client's browser.

Step 3: Animate the frames at the client, response to user interaction and request for more frames if needed.

Chapter 4 - Develop IRB360 Pick and Place Application

4.1. Description of the Prototype

4.1.1. Robot Structure

The IRB340 robot is inspired from the Delta design. Figure 4-1 shows the structure of a Delta-3 robot, which consist of a fixed base **2** with three motors **1** mounted on it and a moving platform **6** with an end effector, linked together by three closed kinematic chains and a rotation arm **5**. Since the robot is a 4-axis version, each chain contains an upper arm **3**, and a forearm **4** which is a four-bar parallelogram with four spherical joints. The end effector is mounted on the moving plate and it is linked by a fourth actuator secured either on the top plate or directly on the traveling plate. The core advantage of delta robots is speed. When a typical robot arm has to move not only payload, but also all servos in each joint, the only moving part of delta robot is its frame, which is usually made of lightweight composite materials. Due to its speed, delta robots are widely used in pick-n-place operations of relatively light objects (up to 1 kg).

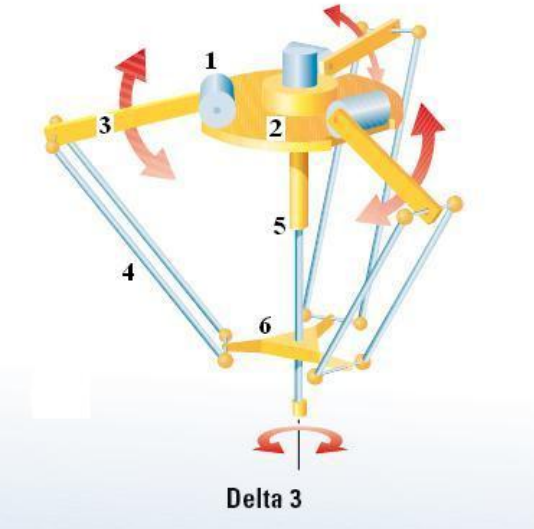


Figure 4-1 Delta-3 robots architecture

Compared to serial structure, parallel structure robots have a number of advantages. They have higher payload since the payload is divided by several links; parallel also has higher accuracy due to non-cumulative joint error; parallel has higher structural rigidity, since the load is usually carried by several links; the location of motors are closed to the base, thus simpler solution of the inverse kinematics equations. (Ronen Ben-Horin,Moshe Shoham,Shlomo Djerassi)

4.1.2. Robot Kinematics

a) Geometric Model

The geometric model of the robot is developed from the Figure 4-2 below. An absolute reference system $\sum$ is placed on the center of the fixed base **2**, with rotation arm **5** perpendicular to the base and the motor **1** lying on the *x* axis. A second reference system $\sum_{eff}$ is placed on the center of the mobile platform which is the working position of the robot. Based on the mark up on the figure, the number of arms *i =1, 2, 3*, with each arm has a reference system $\sum_i$ that is located at distance *R* form the global reference system $\sum$ and rotated at θ_i. (J.M.Sebastian, R.Saltaren, R.Aracil,J. Sanpedro, 2005)

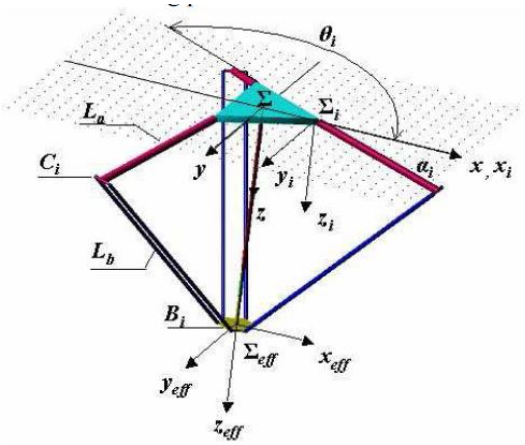


Figure 4-2 Geometric Model

Therefore, from the above description, the geometric parameters relationship and the kinematic equation for each arm of the robot based on vector relation are:

$$\|\sum\sum_i\| = R\left\|\sum_{eff} B_i\right\| = \text{r},\ \|B_iC_i\| = L_i\ and\ \|\sum_i C_i\| = L_a,$$

where i = 1,2,3, and,

$$B_iC_i = C_{i\sum}, i = 1,2,3,$$

By substitute the geometric parameters of the robot and the position of the moving platform B_i into Equation 2, we can obtain:

$$B_iC_i = {}^{\Sigma}R_{\sum_i}\begin{pmatrix} L_a\cos(a_i) \\ 0 \\ L_a\sin(a_i) \end{pmatrix} + \begin{pmatrix} \Delta r\cos(\theta_i) \\ \Delta r\sin(\theta_i) \\ 0 \end{pmatrix} - \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

The restriction equations for the robot are also given by taking the forearms into consideration:

$$\|B_iC_i\|^2 = {L_a}^2,\ i = 1, 2, 3,$$

equal to

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = {L_a}^2,\ i = 1, 2, 3,$$

where,

$$x_i = (\Delta r + L_a\cos(a_i))\ \cos(\theta_i),$$

$$y_i = (\Delta r + L_a\cos(a_i))\ \sin(\theta_i),$$

$$z_i = L_a\sin(a_i).$$

b) Inverse Kinematics

Inverse kinematics is used to calculate the three arm angles a_i for the robot by providing a known position $\sum_{eff}$. The Flexpicker model uses the result found by R.Clavel, 1988. Equation 5 forming a sphere located at the center of B_i with radius L_a. Intersection of this sphere and XZ plane is a circle with center in point C_i and the restriction forced by the circular trajectory of C_i is

$$(x - R)^2 + z^2 = {L_a}^2.$$

By solving the equation 6 we can get two possible locations for point C_i based on the angle calculation.

$$x - R \geq 0 \qquad \text{where } a_i = \sin^{-1}\frac{z}{L_a},$$

$$x - R < 0 \qquad \text{where } a_i = \pi - a_i.$$

c) Forward Kinematics

The forward kinematics was used to transform the output angles from the three motors into the actual value of the Tool Centre Point (TCP). In this case the three angles a_i are given to determine the value of $\sum_{eff}$ and the center of the mobile platform is given by equation 5.

d) Jacobian Matrix

To map the speed between the actuators and moving platform, it can be calculated using Jacobian by differentiating equation 5 with respect to time.

$$L_a[\Delta r - x\cos(\theta_i) - y\sin(a_i) + z\cos(a_i)]\dot{a}_i$$

$$=(x - x_i)\dot{x} + (y - y_i)\dot{y} + (z - z_i)\dot{z}, i = 1,2,3,$$

$$= \boldsymbol{A\dot{r}} = \boldsymbol{B\dot{q}},$$

where $(\dot{x}\ \dot{y}\ \dot{z})^T$ is the operational speed vector and $(\dot{a_1}\ \dot{a_2}\ \dot{a_3})^T$ is the manipulator joint speed vector. With $i = 1,2,3$, A and B can be expressed as

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\mathbf{B} = \mathrm{diag}(b_{11}, b_{22}, b_{33}).$$

Thus, the Jacobian matrix of the robot can be written as

$$\mathbf{J} = \mathbf{A^{-1}B}.$$

4.2. PickMaster™ Control Program

PickMaster is the Windows based control software from ABB that allows the programming, control and operation of picking robots via the integration of high performance S4Cplus robot controller. PickMaster can control up to 8 robots and 8 cameras with the conveyor tracking simultaneously in one application or concurrently in independent cells. The software distributes the work evenly between the robots, and compensates even if one robot goes temporarily out of service. PickMaster is the best tool for guiding robots in the packaging process as it gives the IRB 340 a capacity of 150 picks per minute and more. Figure 4-3 shows a screen view of the PickMaster studio environment interface. The project view, as

markup in the figure, is the most important area of the main application. From here, all objects are created and configured. Every object in this view, as well as the view itself, can show a popup menu with a set of commands that can be performed on that object.

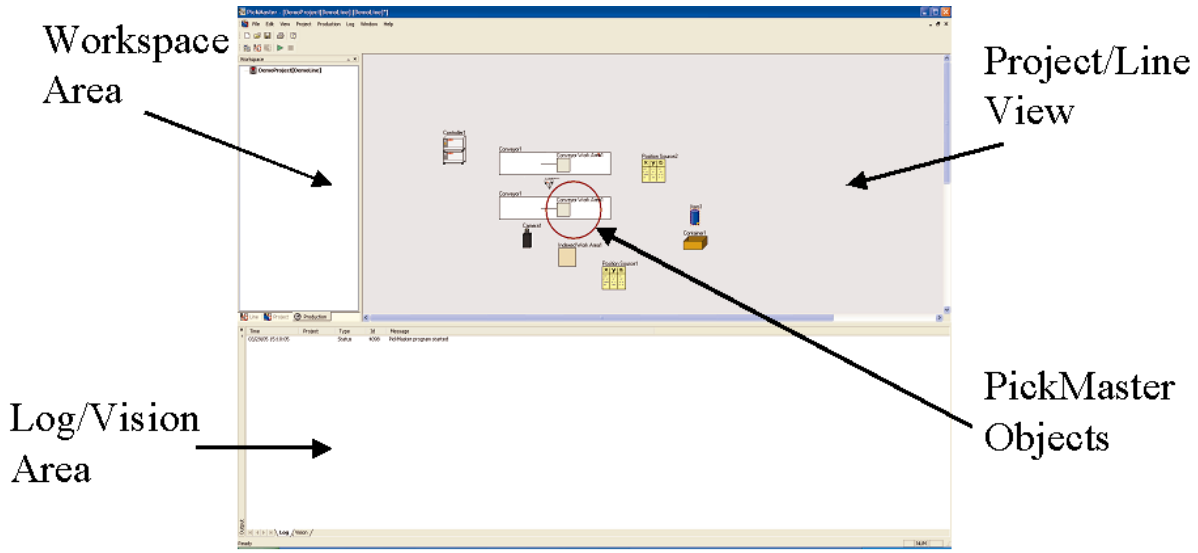


Figure 4-3 PickMaster Interface Environment

The PickMaster package also come with the RIS build in function that can be used to realize remote control by any production panels or external devices. A vision system is also developed to co-operate with the PickMaster to find randomly placed items on a feeder.

For an application program, it consists of two different parts: a line and project, which need to be set up separately. All the physical objects, such as robots, cameras and conveyors are available within a line, whereas the project stores information of items, connections between work areas and cameras. The reason for this segmentation is to keep all things that will not be changed very often in a separate line file. In general, a line will be created and configured once and for all. There may be several existing lines but mostly only one is used. Many different projects can be created to use one line. In Figure 4-4, an organization chart has been

constructed to summarize the components that need to be configured for the whole PickMaster project.

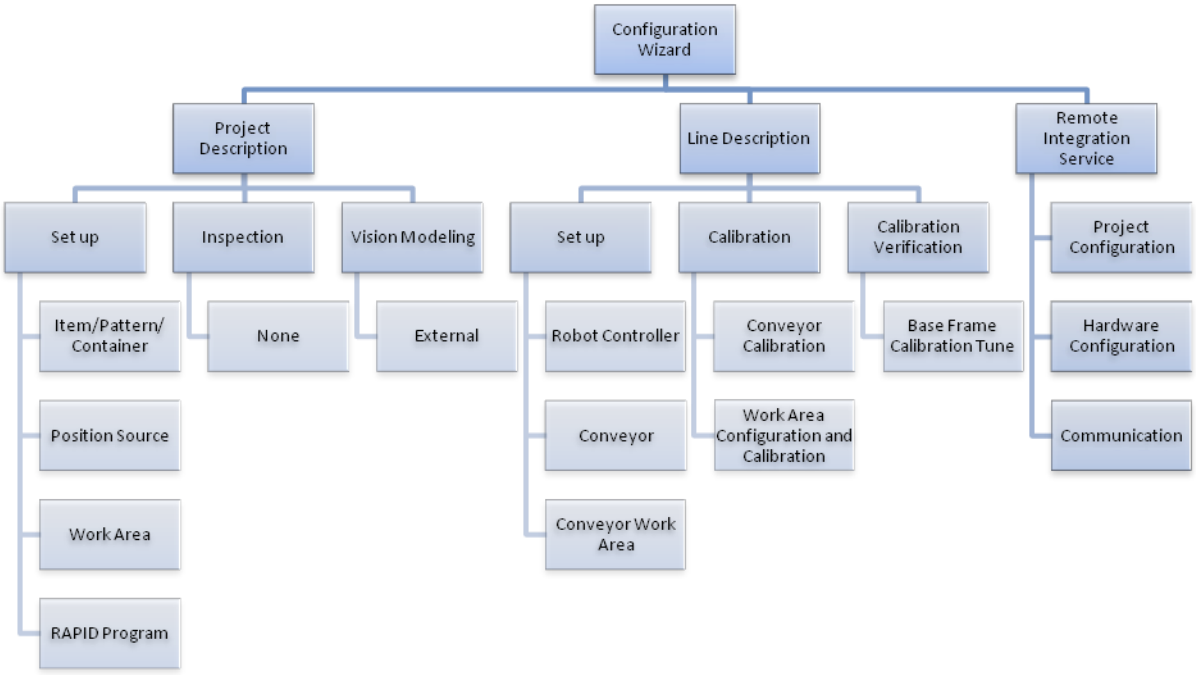


Figure 4-4 PickMaster Project Design Flow Chart

For the project design, all the major configuration and description are carried out under the PickMaster environment except the calibration portion under the line description, which are done from the teach pendant.

4.2.1. Line Description

4.2.1.1. Set up

To be able to run a project with PickMaster, a line must be created first with proper description and configuration of the physical line. Usually the line is only needed to configure once, unless the equipment in the physical line have changed. The PickMaster program keeps references to all available lines on the computer and only registered lines can be used to create projects.

Before adding components to the line, a robot controller has to be created first. In a line, at least one robot must be defined and the line can have up to eight robots. By right clicking the robot controller, it brings up the configuration window for robot IP address setting and connection method. The following step is to define whether the conveyor work area and index work area to be a pick, place or other.

4.2.1.2. System Calibration

The impact on high preferment and accuracy of robot action depends on preliminary setting, where the system calibration plays an essential role. The key concept is to define a coordinate system origin that is the same for a robot base frame and the work object. If the system is not calibrated precisely, this may result in incorrect positioning and will crash the equipment. A comprehensive calibration process stated in the manual includes vision calibration and robot calibration. Since the vision system is developed separately, therefore it is mainly focused on robot calibration.

The robot calibration consists of three main steps: controller preparation, calibration of counts per meter, and calibration of the conveyor base frame. According to PickMaster manual, it is recommended to calibrate with the following order if a system is to be calibrated for the first time:

- Calculate counts_per_meter parameter value (for conveyors only)
- Perform the base frame calibration

Controller preparation:

- Update the encoder by running the conveyor a small distance and stop
- Controller was set to automatic mode
- RAPID program of Pmppacal.prg was loaded to controller

Calibration of counts per meter:

The I/O parameter *CountsPerMetre* for the encoders defines the number of counts that the encoder generates when the conveyor has moved one meter. The values of the Counts per Meter can be obtained from Equation 1 and the procedures are shown below:

- Position 1: Read the conveyor position in the jog window on the teach pendant for an object/mark on the conveyor.
- Position 2: Move the conveyor some distance, and then read the conveyor position in the jog window again.
- Measure the distance between two positions.
- The calculated value is entered into the controller for accuracy testing and then run the calculation again until the counts per meter equal to old counts per meter.

$$\text{Counts Per Meter} = \frac{10000 \times \left((\text{Position2-Position 1}) \times \text{Old Counts Per Meter} \right)}{\text{Measure Distance}}$$

Base frame calibration:

The accuracy of the conveyor tracking is highly dependent on the accuracy in calibrating the conveyor base frame. The conveyor base frame calibration method will use the measurement of 3 positions of the same object on the conveyor to determine the conveyor base frame.

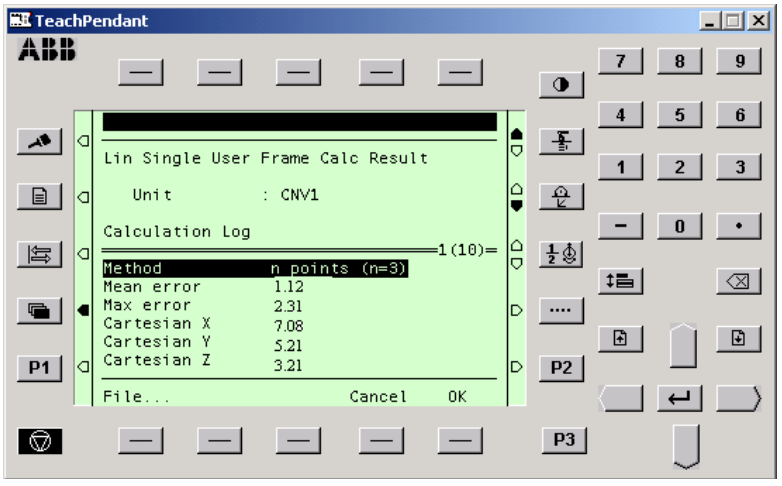


Figure 4-5 Calibration Result

4.2.2. Project Description

4.2.2.1. Set up

The purpose of a project is to define the pick/place sequence of a robot and also the orientation of an item as well as its coordination. The dimension of the item can be defined in the *Item Configuration* window and the defined pick height is always added to items found by a pre-defined position source. Also, the grip location of an item defines the pick/place position relative to the item position can be adjusted in the *Grip Location* by right clicking an item in the project view.

Position source is the main object in the project as it defines how to generate item positions and which work area to send them to. Varying configurations of pick/place motion, item distribution method and source type can be defined in the *Position Source* menu. In this case, the *External Sensor* option is selected as the *Source type* of the project with the *Triggering Type* of Digital I/O. The I/O triggers the position source with the position generator signal defined for the selected work area

4.2.2.2. Work Area

The PickMaster contains two types of work area for different tasks: conveyer work area and index work area. The conveyer work area is mainly for the robot picking or placing, while an indexed work area is for picking or placing without conveyor tracking function. In a project, a robot usually associates to one conveyer work area on a conveyor but may have several if the robot has more than one encoder card for that conveyor. Since the conveyor tracking is not used in indexed work area, but the products may be random placed with the need of vision recognition and I/O signals. They are used to indicate when the work area is in position and when to indicate when the robot can start the pick and place execution.

Once the type of work area is defined for each conveyor, the conveyor actions and robot's workspace needs to be defined. As can be seen in figure 4-6, a fully defined conveyor work area is demonstrated. The distance from the center of robot to *Enter/Exit* needs to specify as *Enter* defines the limit from which the robot starts to execute item targets on the work area, while *Exit* is where the robot determines an item target out of the work area. The positions of the *Enter* and *Exit* must be chosen well within the robot's maximum reachable workspace, thus the robot can reach a target before it goes out of range.

The *Start* and *Stop* limit define the behavior of the conveyor. When the next item in a queue on the conveyor is above *Start* limit, the conveyor is started, while an item reaches the *Stop* limit, the conveyor is stopped. The advantage of having *Start/Stop* feature enabled is to allow more flexible conveyor motion during the pick and place process as sometimes the robot may not be able to pick all the items in a queue, therefore, by supervising the start and stop limits allows PickMaster to perform more efficiency.

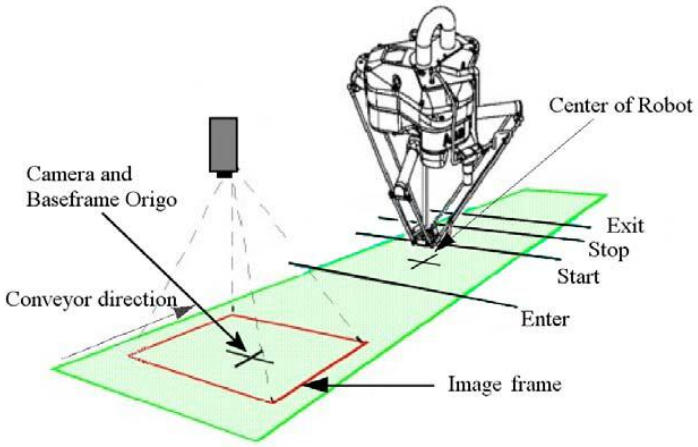


Figure 4-6 Defined Conveyor Work Area

4.2.3. RAPID Program

The RAPID language is aimed to support a leveled programming concept where new routines, data objects and data types may be installed at a specific IRB site. This concept makes it

possible to customize the programming environment and must be fully supported by the programming language. RAPID language is exceptional by not only having most functionality found in other high level programming languages, but also it is specially designed to control robots. Most importantly, there are instructions for making the robot move.

With RAPID, a number of powerful features are introduced:

- Modular Programming - Tasks/Modules
- Procedures and Functions
- Type definitions
- Variables, Persistents and Constants
- Arithmetic
- Control Structures
- Backward Execution Support
- Error Recovery
- Undo Execution Support
- Interrupt Handling
- Placeholders

An RAPID application is called a task. A task is composed of a set of *modules*. A module contains a set of type definitions, data and routine declarations. As shown in Figure 4-7, the task buffer is designed to store modules that are currently in use. RAPID program code in the task buffer may be exchanged with external devices either as separate modules or as a group of modules. (ABB, 2004-2007)

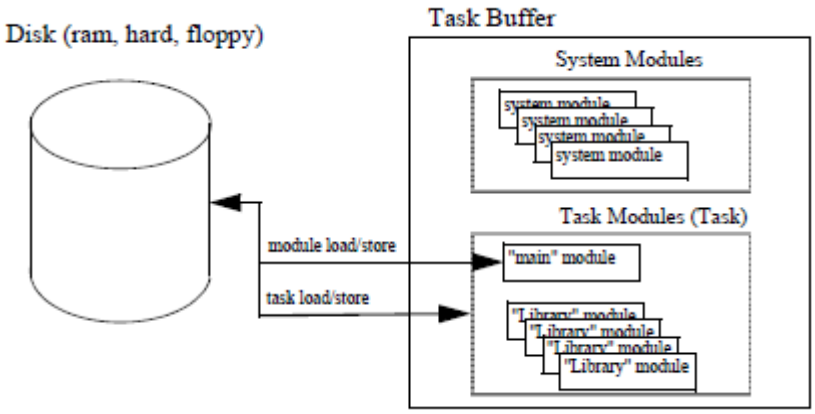


Figure 4-7 RAPID Application Overview

A module initialization contains the name, attributes and body of a module. The body of a module declaration contains a sequence of data and routine declarations. (ABB, 2004-2007) The default RAPID program contains three program modules, *PPAMAIN*, *PPASERVICE* and *PPAEXECUTING*. Among the three, only PPAMAIN module is not recommended to edit for customization purpose as it is used to handle the main program initiations and execution sequence. The following pseudo code of the PPASERVICE is taken out from the program to represent a typical module layout:

<module declaration> ::=

MODULE <PPASERVICE> [<module attribute list>]

<type definition list>

<data declaration list>

<routine declaration list>

PROC Home ()

Return to safe position;

ENDPROC

PROC WashDown()

Executed wash down routine;

ENDPROC

```
PROC TestCycle()
    Executed test cycle;
ENDPROC
PROC Homepos()
    Return to home position;
ENDPROC
```

ENDMODULE

As shown in figure 4-8, the data flow for a typical pick and place operation. First of all, PickMaster receives position information from the camera and vision system, which is then sent to the robot's internal buffer. The position information is used to locate the items on the conveyor and also helps synchronizing the robot's movement with the conveyor motion. The RAPID program is built in default with six different movements that can be applicable with different range of robot. The sequence in which the robot moves is given by the pick and place buffers. The robot executes the pick sequence by moving down to the pick position while tracking the conveyor, which means that the TCP will follow the pick target during the pick time. Then the robot grips the item using vacuum suction, rises and moves to the second conveyor with the synchronized speed. The robot repeats the cycle soon after it drops the item on the place position.

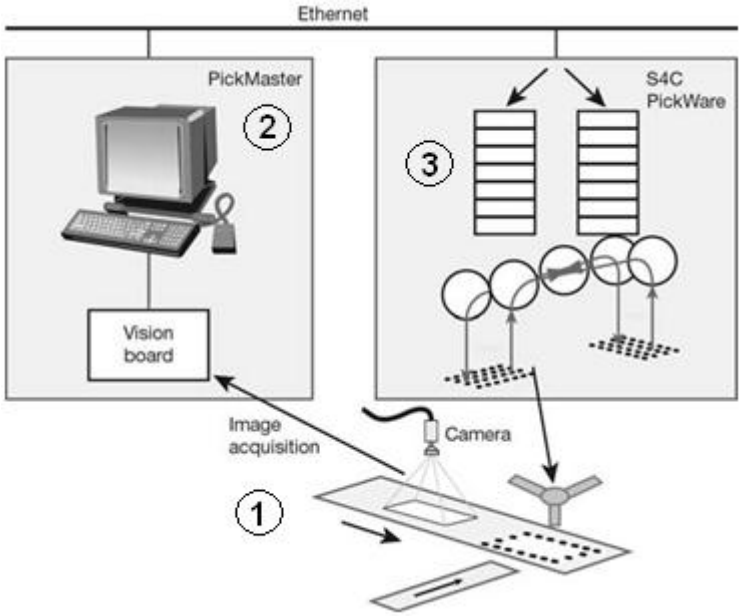


Figure 4-8 PickMaster Work Flow

4.2.4. Remote Integration Services

The Remote Integration Services is designed to interface with external devices through various ranges of protocols such Ethernet, RS232, TCP/IP and Digital I/O. Different plug-ins can be configured to interface with PickMaster commands through the interpretation from a specific command of a certain communication technology. RIS provides the runtime interface as well as the configuration menus to set up the command codes.

The Digital I/O plug-in from Advantech is chosen makes it possible to issue PickMaster commands as digital I/O signals. The plug-in can be used with the Advantech I/O boards PCI-1750 and PCI- 1756. Specifications for these boards are shown in Table 4-1.

	PCI-1750	PCI-1756
Isolated inputs	16	32
Isolated outputs	16	32
Input range	5-50V or dry contacts	10-50V
Output range	5-40V open collector	5-40V
Input current at 24V	N/A	4mA
Sink current (max per channel)	200mA	200mA
Power consumption (typical)	850mA	285mA

Table 4-1 Specifications for the supported Advantech I/O boards

A remote communication session for stopping robot 1 has been fully illustrated in figure 4-9 between any RIS client and the PickMaster RIS server. The PickMaster PC functions as a server and initiated commands from a RIS client, in this case is a PLC. The RIS server monitors all the inputs for control signals and acknowledges back the RIS client if any command status is changed or failed to execute. In the initial state is PickMaster started without a project opened and robot 1 is in auto mode.

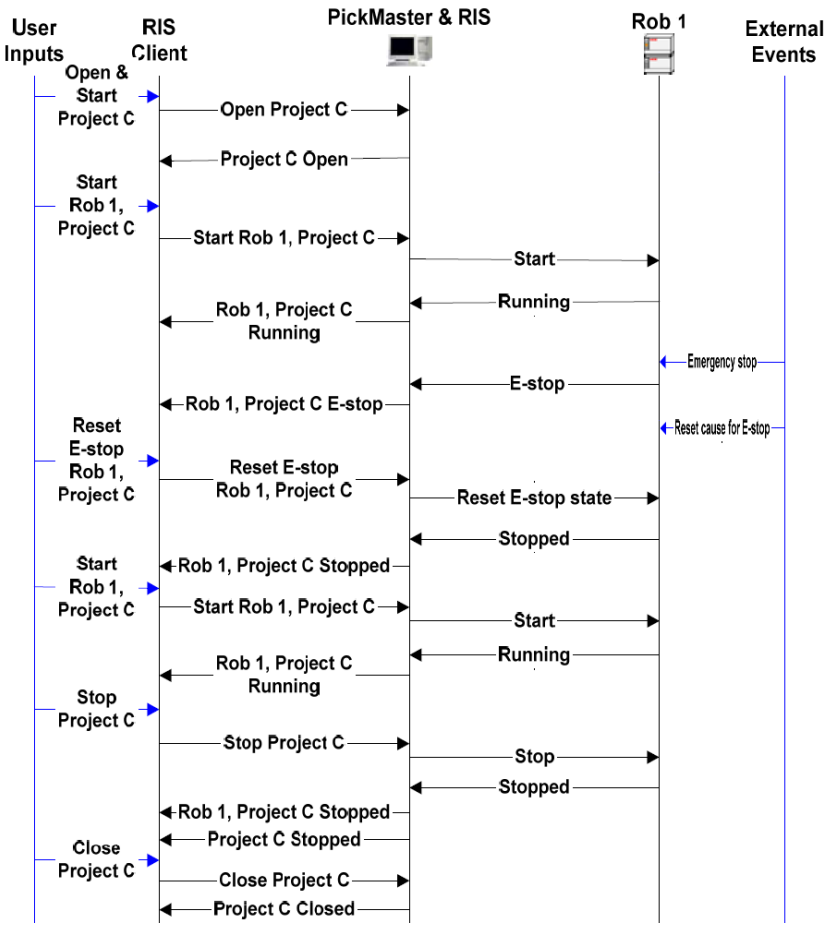


Figure 4-9 RIS & RIS Client Communication Session

In figure 4-10 below, there are four groups of I/O channels required to issue commands to PickMaster, each group of I/O channels are used as binary values such as in four input channels can be used to describe the values 0-15. The timeout and the poll interval are set in the hardware configuration dialog box. Each project, which shall be accessed by the Digital I/O plug-in, must be given a unique number. Furthermore, every robot in a project must also be given a number, which is unique within the project.

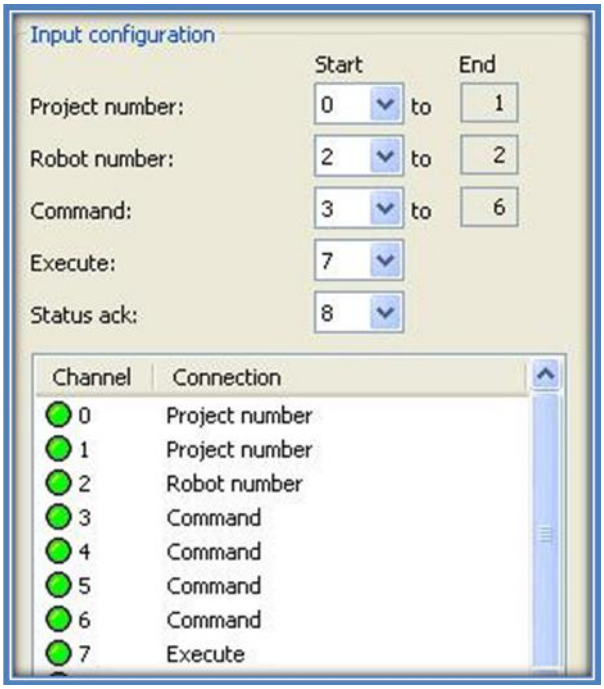


Figure 4-10 Channel Assignment

Figure 4-11 below shows the commands that are available in PickMaster with an example of *Stop* command timing diagram. The commands accepted by the Digital I/O plug-in can be divided into project and robot commands. A specific command is always issued as a binary value on four input channels. For project commands, the *Project number* input must indicate the project for which the command should be executed. For robot commands, both the *Project number* and the *Robot number* inputs must be set to indicate to which robot in what project the command should be sent to. To notify the Digital I/O plug-in that a command should be executed, the *Execute* signal must be set high. The inputs are then read and *Command executed* signal is set high. Before the next command arrived, the *Execute* signal must be set low again before another command can be issued or status messages sent.

	Decimal Value	Binary Value
Project Commands		
Start	3	0011
Stop	4	0100
Robot Commands		
Start	6	0110
Pause	7	0111
Stop	8	1000
Reset Emergency Stop	10	1010

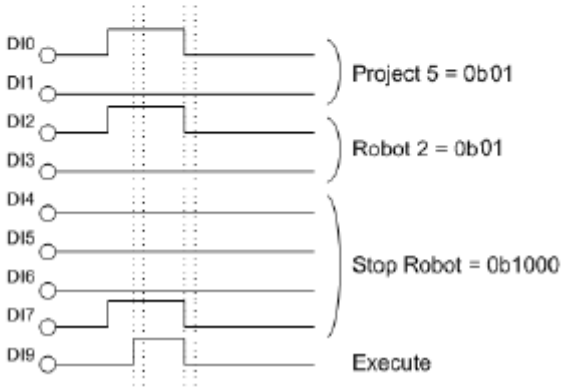


Figure 4-11 Commands for the DI/DO

4.3. Develop Visual Line Tracking for Industrial Robot

Generally speaking, machine vision is an “eye” to a system that can detect target objects. A camera system represents in such a way that it connects to a processing unit for image processing and to control a system. According to (Zuech & Miller, 1989), vision system provides a sensory feedback for industrial robot to intelligently interact with the surrounding environment based on the collected information. In recent years, the world’s industrial inspection activities are increase dramatically and almost 90% of the activities requiring vision will be done with computer vision systems. It is also predictable that the trend will increase more in the next decade.

Modern machine-vision systems typically contain faster and more powerful PC platforms, robust 32- bit operating systems, and easy-to-use integrated software applications, making

machine-vision systems more powerful, easier to program, and less expensive to use than ever before.

The system under control of machine vision could be used in a wide variety of manufacturing operations for repetitive inspection tasks in which accuracy and reliability are important (e.g., verifying date codes on food packaging, inspecting automotive parts for proper assembly, and performing robotic guidance for pick and place operations). One of the most common applications for vision systems is for the identification in pick and place process as shown in Figure 4-12 below.

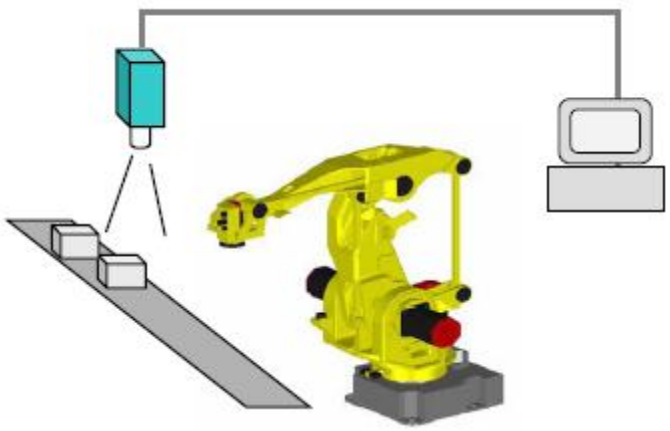


Figure 4-12 Typical System Layout

4.3.1. Vision Hardware

Implementation of the vision system has employed the use of a Logitech 9600 web-cameras, supplying 24 bit ultra-wide RGB colour images at up to 30fps. Capture of the video stream is obtained at a resolution of 320 x 240 for the optimization of both processing speed and image definition.

While there are a number of factors that preclude the use of independent USB cameras in many image processing applications, not least that of independent and automated intensity

and focal adjustment, budget constraints have precluded alternate means of image capture. As a result, synchronization of the conveyor tracking has been achieved through the use of software control in the form of OPENCV's CVCAM library.

4.3.2. Core Machine Vision System

The core vision system - Conveyor Visual Tracking in this project incorporates two primary stages: (1) Target Detection (define as the process of detecting and reacting to the presentation of an object) and (2) Localization, namely the tracking of a given target, which in this case translates to the determination of a relative starting position with regards to the goal pose. According to (Gary Bradski and Adrian Kaehler, 2008), there are several techniques to perform object tracking under the OPENCV environment, such as, Mean-Shift & Cam-shift, Motion Templates, Optical Flow and the use of Condensation Algorithms. Self-organizing maps are too slow to be operated with real-time. Motion template matching requires a prior knowledge of object information to match objects. Unfortunately, due to the required computation load, this method cannot be performed in real time. The method by colour information (Mean-Shift or Cam-shift) covers the former two disadvantages but cannot be used for binary images. The object recognition method using the difference of two images is useful when the environment does not change.

There are four steps each in general for creating object tracking process and localization:

- Step 1: Recognize objects entry by placing the tracking window at the bottom of the frame window.
- Step 2: Perform tracking action based on the colour information (Mean-shift or Cam-shift).
- Step 3: Send the information to the robot by Serial communication.
- Step 4: Reset the tracking window to the bottom of the frame window and wait for

new object to enter.

Object localization:

- Step 1: Initialize vision function.
- Step 2: Create binary image and convert colour space from RGB to HSV.
- Step 3: Specify a tracking window for objects and store objects colour information.
- Step 4: If a specific object matched the select colours, move the tracking window to the object.

During object tracking process, conveyor speed is taken into consideration since the machine vision system is configured to work with a standalone USB camera which is also known as off-line processing system shown in figure 4-13 below. One of advantage is that off-line processing acts as a single host can serve for both camera as well as system control. On the other hand, due to the physical working algorithm of the design, there is a delay in frame-by-frame transmission of video data from the camera, thus it is not suitable for real-time processing for application with fast moving conveyor bell. (Raghavan, 2008) The speed of the conveyor bell needs to be maintained at less than the minimum frame rate of the camera, at around 15 Frame per second (Fps).

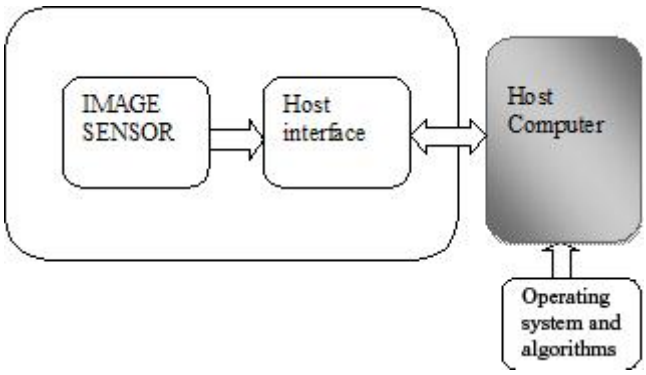


Figure 4-13 Off-Line Processing Overview

The equation for the conveyor belt speed is given by

$$V_c = \frac{X(T^k) - X(T^{k-1})}{T^k}$$

where V_c is the speed of an object (conveyor belt), *X* is the position of an object, T^k is the *k-th* time. The position of the object center is defined to be

$$X = (\bar{x},\bar{y})$$

Based on the above equation, the conveyor speed is calculated and then the new valve is inputted into the robot controller unit.

4.3.3. Mean-Shift Tracking

Mean-Shift algorithm is an effective statistical iterative algorithm by (Fukunaga K, Hostetler L, 1975). They proposed the method of Mean-Shift algorithm for finding cluster center in cluster analysis. By 1995, (Cheng, 1995) improved the algorithm in the Mean-Shift kernel and weight function, and applied to clustering and global optimization in that it expanded the scope of application of the algorithm. From 1997 to 2003, Comaniciu D & Meer P (1999) applied this method into the image feature space analysis, image smoothing and segmentation processing. He also proved that under certain conditions are met; Mean-Shift algorithm can converge to the nearest one-point probability density function of the steady-state. Therefore, Mean-Shift algorithm can be used to detect the existence modes of the probability density function. As the Mean-Shift algorithm is totally dependent on the sample points in feature space analysis, it does not require any a priori knowledge, convergence speed, therefore in recent years it has been widely used in image segmentation, tracking and other areas of computer vision. (Comaniciu D, Ramesh V, Meer P, 2003) In this thesis, because mean-shift

itself is a fairly deep topic, the goal here is aimed mainly at developing intuition for the design of application.

4.3.3.1. Mean-Shift Algorithm

Mean-shift algorithm is the derivation from the gradient of a density function of nonparametric estimation, and nonparametric estimation started from the sample set to estimate the density function. It does not require any a priori knowledge and the distribution of arbitrary shape are valid. One of the most commonly used is the kernel density estimation as it is based on kernel function $K_{(x)}$ by calculating the sample set to get the density function.

Fukunaga K & Hostetler L (1975) in the article defined Mean-Shift as: Given a set $\{x_i\}_{i=1,\dots N}$ of N points in the d-dimensional space R_d , the basic form of Mean-Shift vector at the x point is:

$$M_h(x) = \frac{1}{k} \sum_{x_i \in S_h} (x_i - x)$$

where S_h is region of a high-dimensional spherical with radius of h,

$$S_h(x) \equiv \{y: (y-x)^T (y-x) \leq h^2\}$$

where in the n-sample points x_i, there are k points fall within the S_h region.

The base form have been improve by (Cheng, 1995) later in the following three aspects (1) non-flat kernels are allowed; (2)points in data can be weighed; (3)shift can be performed on any subset of X, while the data set S stay the same. The basic Mean-Shift form became:

$$M(x) \equiv \frac{\sum_{i=1}^{n} G_H(x_i - x)w(x_i)(x_i - x)}{\sum_{i=1}^{n} G_H(x_i - x)w(x_i)}$$

where $G_H(x_i - x) = |H|^{-1/2} G(H^{-1/2}(x_i - x))$; G(x) as a unit kernel function; H as a positive definite symmetric d x d matrix, commonly known as bandwidth matrix; $w(x_i) \geq 0$ for an assigned weight of sample point x_i.

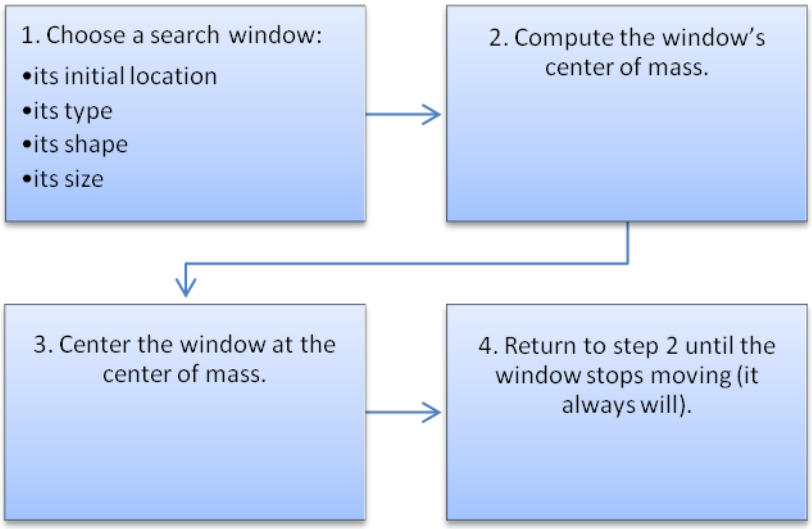


Figure 4-14 Mean-Shift Algorithm Logic Flow

Based on the above figure, the designer have to first choose the feature distribution to represent an object (e.g., color + texture), then start the mean-shift window over the feature distribution generated by the object, and finally compute the chosen feature distribution over the next video frame. Starting from the current window location, the mean-shift algorithm will find the new peak or mode of the feature distribution, which presumably is centered over the object that produced the color and texture in the first place. In this way, the mean-shift window tracks the movement of the object frame by frame.

4.3.3.2. Mean-Shift Target Tracking

As presented in (Gary Bradski and Adrian Kaehler, 2008), the mean shift iterations are employed to find the target candidate that is the most similar to a given target model, with the similarity being expressed by a metric based on the Bhattacharyya coefficient.

Comaniciu D & Meer P (2002) proposed the used of Bhattacharyya coefficient to define the similarity measure. In generally it use the distribution of objects' gray or color to describe this object, assuming that the object center located at x_0, then the object can be expressed as:

$$\hat{q}_u = C\sum_{i=1}^{n} k(||\frac{x_i^2 - x_0}{h}||^2)\delta[b(x_i^s) - u]$$

Candidate objects in the y can be described as:

$$\hat{p}_u(y) = C_h\sum_{i=1}^{n} k(||\frac{x_i^2 - y}{h}||^2)\delta[b(x_i^s) - u]$$

Therefore, object tracking can be simplified to find the optimal y, makes the $\hat{p}_u(y)$ and $\hat{q}_u$ most similar. $\hat{p}_u(y)$ and $\hat{q}_u$ similarity with the Bhattacharyya coefficient of $\hat{p}_u(y)$ to measure the distribution, that is,

$$\text{Bhattacharyya} = \sum_{u=1}^{m}\sqrt{p_u(y)\hat{q}_u}$$

Substitute the equation $\hat{p}_u(y)$ into the Bhattacharyya coefficient and simplify as follows,

$$\text{Bhattacharyya} \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{p(y_0)q_u} + \frac{C_h}{2}\sum_{i=1}^{m} w_i k(||\frac{y - x_i}{h}||^2)$$

4.3.3.3. Mean-Shift Tracking Cons and Improvement

Mean shift tracking uses kernel histogram to modeling targets and then tracking the matched one. The algorithm required less parameters, very robustness and fast implementation of real-time model calculations. However, due to their own characteristics Mean shift specified in Fashing M &Tomasi C (2005), which of the following disadvantages:

(1) The kernel function histogram in describing the target characteristics is relatively weak, especially in tracking target in the gray-scale or less texture information image, you cannot get good tracking results;

(2) As the Mean shift is a kind of climb hilling algorithm, so a common drawback with this algorithm is easy to fall into local maxima and low convergence speed;

(3) As kernel function bandwidth matrix has been simplified, throughout the tracking process, the window width of the kernel function remain unchanged, which means that it cannot adapt to target with multi-DOF in some circumstances; for example, target scaling, rotation and other changes;

(4) It does not have a good solution in target blocking problems.

To overcome the shortcomings of Mean-Shift algorithm, people conducted a lot of research and come up with corresponding improvement methods and experiments to prove the effectiveness of these methods.

4.3.4. Tracking Processing Logic Flow

The algorithm that has been employed utilizes the continually adaptive mean shift algorithm as presented by Bradski (1998). Though never intended as a tracking algorithem this method

has been selected based on its functionality, in terms of, noise exlcusion, speed (30fps), efficiancy and ability to cope with dynamically changing colour probability distributions.

The CAMSHIFT algorithem is fundamentaly an extension of the appearance based obstacle detection algorithm presented in (Iwan Ulrich,Illah Nourbakhsh , 2000). where object feature identification is performed through the appraisal of probability distributions (1D saturation and intensity histograms) within the HSV colour space.

The discrepancy arises in the application of a dynammic search window which defines a singular Area of Interest (AOI) within the image. Thus, rather than the backprojection of the reference histograms against the full image, the adapative near shift algorithem continually adapts the location of the search window to optimise the recovery of the tracked feature between consecutive frames. The algorithem, as presented by Bradski (1998), is outlined in the following figure.

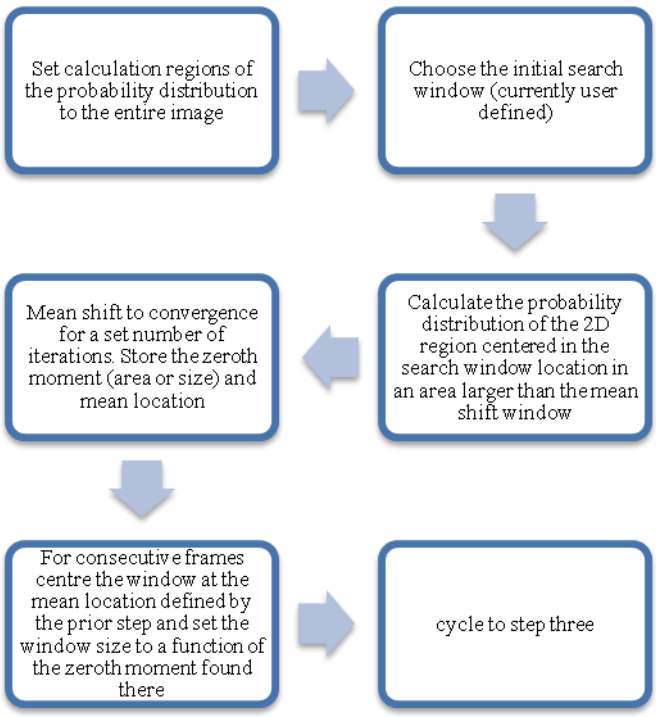


Figure 4-15 Adaptive Mean Shift Tracking Algorithem

A short segment of video sequence is presented in figure 4-16, illustrating the succesful implementation of the algorithm, in the tracking of a specified feature in real-time (30fps).

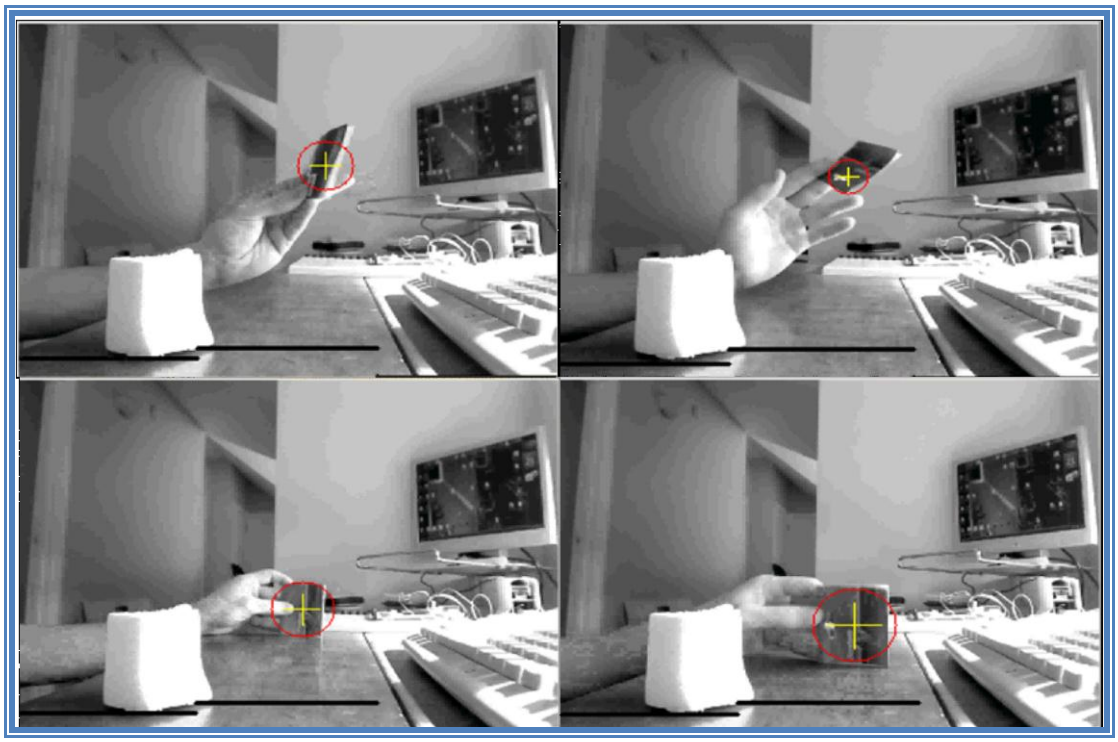


Figure 4-16 video sequence exhibiting both the camshaft and ABOD applications

While the algorithem presents an extreamly effective and suitable means of feature colour tracking, it is noted that further improvements (i.e. increased robustness to drift and a reduction in the overhead) may be attained through the use of a more principled means of probability evaluation, such as that of Bayes rule (Hewitt, 2007), as opposed to the heuristic that has been employed here. This area is therefore one of ongoing development.

4.4. Hardware Design

4.4.1. Vision Trigger Board Design

The camera using in this project is a normal USB webcam, it does not have a built in I/O system, which means it cannot trigger directly from the robot controller. Therefore the trigger

strobe board here is responsible to trigger the Flexpicker when the camera indicates the presence of objects on the conveyor. This switch should be present so that it provides a reliable and repeatable signal sending out from the host PC to the robot.

Vision triggering is one of the most important parts of the PickMaster concept functions that equip with S4Cplus controller. It enables very precise synchronization between the robot controller and image requirement. Once the position of the object is triggered, the robot will be programmed to approach the moving object and maintain the TCP speed close to the work object speed.

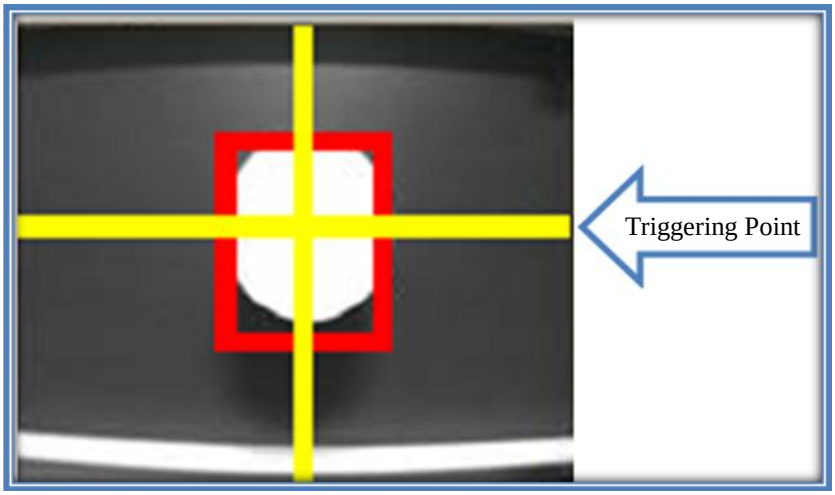


Figure 4-17 Object Tracking with Position Triggering

The screenshot of vision tracking is presented in figure 4-17 demonstrates that during the tracking process, the trigger strobe works by waiting for the incoming signal from the PC serial port when objects passed the triggering point; a microcontroller is then used to convert the pulse string signal into digital logic high/low, since the S4Cplus controller equip with digital I/O function for external devices interfacing. In order to mechanically switch a relatively high Current/Voltage ON/OFF, a relay driver circuit is designed to provide the necessary current (typically 25 to 70ma) to energize the relay coil.

4.4.2. Circuit Design & Microcontroller Programming

Figure 4-18 shows the circuitry design for the trigger strobe. The top-right portion is a level converter uses a Max232 and five capacitors to adapt the RS-232 signal voltage levels to TTL logic. It is a very popular method of interfacing serial port to microcontroller. The bottom-right portion is a basic relay driver circuit. As can be seen an NPN transistor is being used to control the relay. The transistor will turn ON the relay when a LOGIC 1 is written on the microcontroller PORT pin. The relay is turned OFF on the other hand. A diode is connected across the relay coil to protect the transistor from damage due to the back EMF generated in the relay's inductive coil when the transistor is turned OFF. (Shah, 2008)

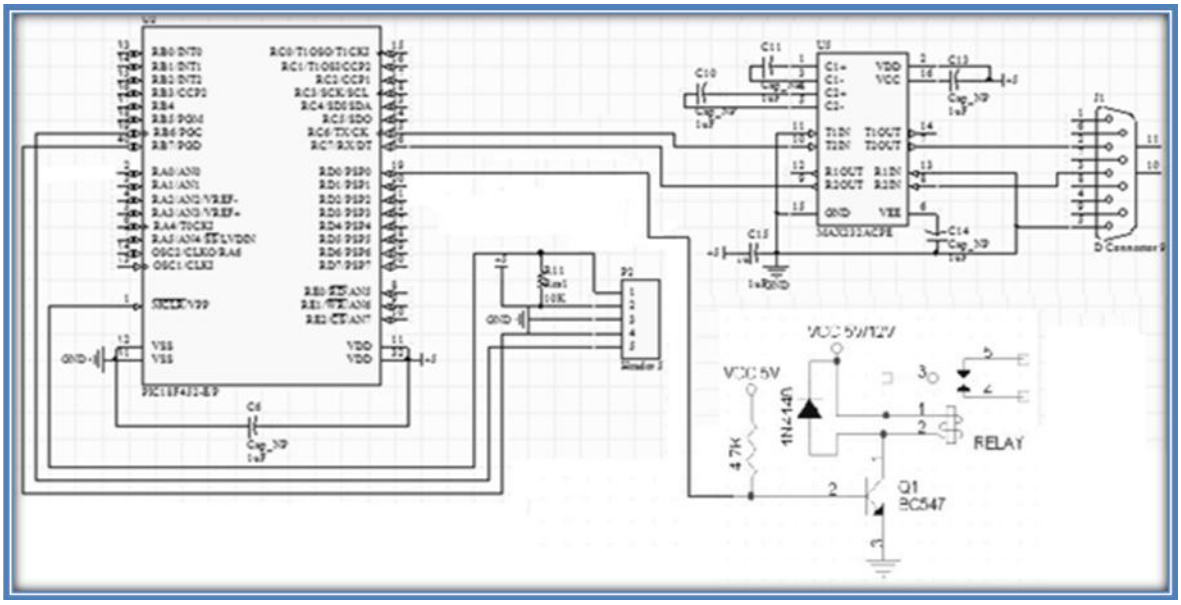


Figure 4-18 Circuit design for trigger strobe

Choice of the embedded controller was based on the need to read and write to non-volatile memory and write data to a serial port. The microcontroller used in this project was a PIC18F4520, since the selection of the PIC microcontroller was primarily attributed to familiarity and prior experience with Microchip Microcontrollers. Before the program start, the SCI (serial communication interface) communication requires initialization. The speed at

which the communication will take place must be written to the baud register. The default setting of the serial communication port is 9600 baud rate. Putting a special value into the baud rate control register will ensure communications at the appropriate speed. That value can be calculated by the equation,

$$BUAD = F_c /16 \text{ x } BRR,$$

where f_{ck} is the frequency of the clock, in and BRR is the value written to the register. The

SCI was configured for 8 data bits, no parity, and 1 start bit. Using the equation for BRR, the value of nine was loaded into the baud rate register. The receiver interrupt was enabled but the transmitter empty interrupt was not. It was important for the receiver interrupt and the transmitter interrupt to be mutually exclusive with regard to being enabled. This was because the SCI was not full duplex and sent and received data may clash, causing garbled messages.

The figure below shows the Serial Communications Control Register. Here the receiver and transmitter (bits two and three) are enabled. Bits six and five enable the transmitter or receiver interrupt respectively when set.

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Figure 4-19 UART setting up the register

Once the communication between the systems is established, the PIC18F4520 will wait for the triggering signal from the camera and switch on the relay to acknowledge the robot perform tasks.

Chapter 5 - System Integration and Testing

5.1. Testing of Individual System Components

5.1.1. Web-based SCADA Application Testing

There are three stages involved in the testing process. The first stage was to test the server by accessing the service through a remote web browser. The WinCC Web Navigator is used for place a WinCC project online for a user on the remote end. The purpose of testing in this stage is to make sure that network communication between client and server can be correctly established and installed all the necessary software. This allows some problems in the server software or capability issues to be worked out before more complicated interactions with the client application compounded any error such as firewall blocking, administration right or network configuration. Figure 5-1 pictures the test conducted on the client's PC by just simply starting the IE browser, then type in the address of the server and login with the registered account. Once the user is logged in, the user will be prompted to install all the required software (Figure 5-2) and after installation, the project running on the server site will display on the browser windows.

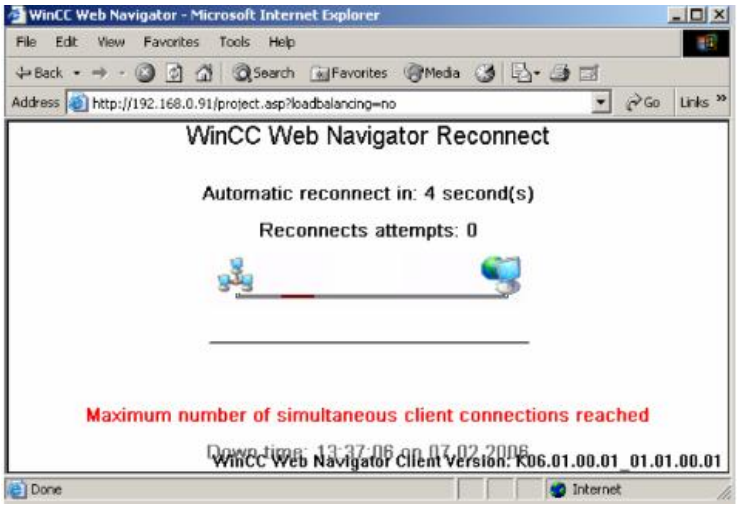


Figure 5-1 Connecting to the Server

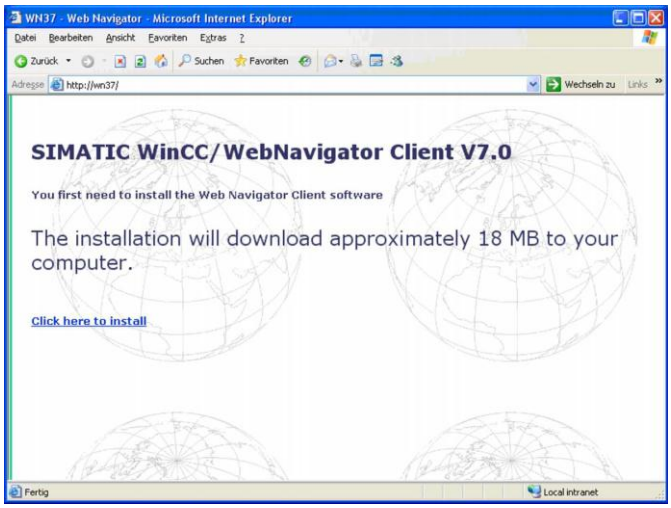


Figure 5-2 Web Navigator Client Installation

To qualify the application's performance

The second stage is mainly testing the project on the server machine, as well as running the client application part on the same machine. This test can demonstrate that the client application and server project could communicate correctly without any issue caused by WinCC configuration. The test began by starting the project on the server, and then typed in the server address into the web browser. Once it is logged in, we can see the process pictures being loaded, then changing the state of the button or values in the process pictures and we can see the response of the robot from the surveillance camera as the real-time data is sent to the process tag in binary format, thus exchanged data between the WinCC and PLC system. When certain statues and changes in the process, the WinCC message system generates messages and output them as tables in Runtime, which helps in identifying critical situation early so that the system information can be monitored efficiently.

The last step is to repeat stage 1 and 2 again but across a network. The test was conducted on a Windows XP machine initially, and then later a Windows Vista machine was used. Throughout the test we found that the communication was established effectively between the client and the server application without any compatibility or network issues.

5.1.2. PickMaster's Application Testing

Testing of the end application consisted of ensuring it was able to deliver the pick and place tasks and accept remotely initiated commands from a RIS client over PCI-1750 DI/DO card, as well as be interfaced with the vision system.

When all configurations are set properly in the PickMaster environment, implementation of the project and testing can be done through maintenance dialogue. In the maintenance dialogue, there are various setting for obtaining the best result of pick and place operation. During the testing and implementation, there are 24 groups of configuration setting that have been done to obtain an accurate operation orientation for a selected item, which listed as below:

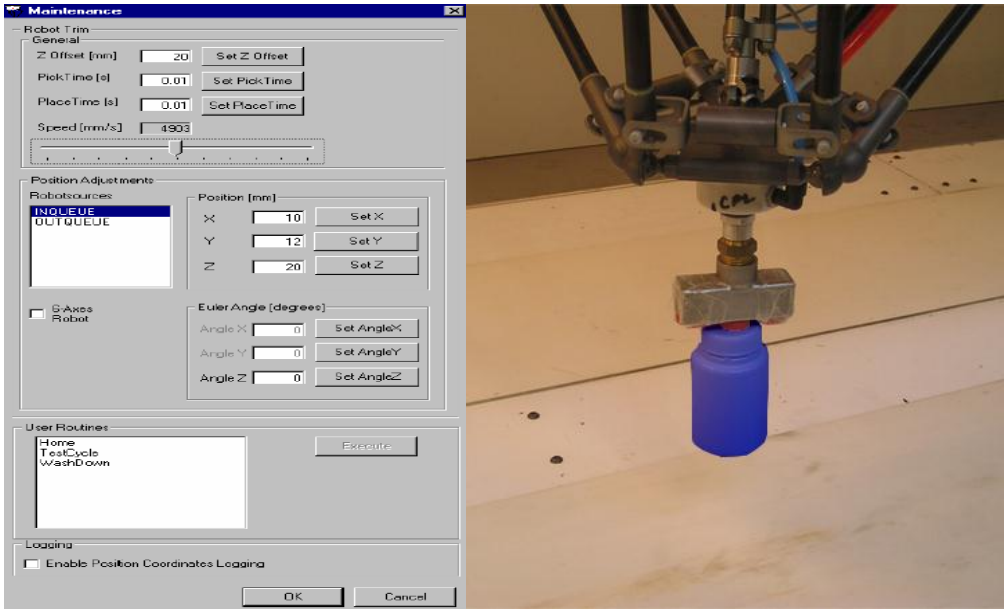


Figure 5-3 Maintenance Setting

NO.1	Z offset(mm)	Pick time (s)	Place time(s)	Speed(mm/s)	Place: Position X-Y-Z (mm)	Pick: Position X-Y-Z (mm)
1	20	0.01	0.01	**3803**	10-10-10	10-10-10
2	20	0.01	0.01	**1500**	10-10-10	10-10-10
3	20	0.01	0.01	**750**	10-10-10	10-10-10
4	20	0.01	0.01	**350**	10-10-10	10-10-10
5	20	0.01	0.01	**500**	10-10-10	10-10-10
6	20	0.01	0.01	**600**	10-10-10	10-10-10
7	**20**	0.5	0.5	600	10-10-10	10-10-10
8	**30**	0.5	0.5	600	10-10-10	10-10-10
9	**40**	0.5	0.5	600	10-10-10	10-10-10
10	**50**	0.5	0.5	600	10-10-10	10-10-10
11	**60**	0.5	0.5	600	10-10-10	10-10-10
12	60	0.5	0.5	600	10-10-10	10-10-10
13	60	0.5	0.5	600	**8-5-5**	10-10-10
14	60	0.5	0.5	600	**7-3-5**	10-10-10
15	60	0.5	0.5	600	**2-3-5**	10-10-10
16	60	0.5	0.5	600	**5-2-5**	10-10-10
17	60	0.5	0.5	600	**5-2-10**	10-10-10
18	60	0.5	0.5	600	5-2-15	**5-3-15**
19	60	0.5	0.5	600	5-2-15	**5-3-15**
20	60	0.5	0.5	600	5-2-15	**5-3-15**
21	60	0.5	0.5	600	**5-2-10**	10-10-10
22	60	0.5	0.5	505	**3-2-20**	**5-3-15**
23	60	0.5	0.5	505	**3-2-20**	**5-3-15**
24	60	0.5	0.5	505	**3-2-20**	**5-3-15**

Table 5-1 Orientation Settings

After testing for many times, the basic pick and place operation was successful, which the bottles could be picked and placed into the predefined position accuracy at limited speed. During the testing, as the speed increased, the bottle would be thrown off by robot as the acceleration was high and needed a stronger force to hold the bottle.

5.1.3. Vision System and Trig Board Testing

In order to test the functionality of the vision application, a prototyped trig board was connected to the server PC's serial port. By configuring the COM port parameters, the communication link between the PC and the device can be established at the baud rate of 9600.

The application started by initializing the camera device using the "CvCapture* capture = 0;"

argument and "! capture" to determine initialization failed. After the initialization, the program will make handshake with COM port, and for the purpose of easy debugging the application, the void Serial_Comm_init() function was created to feedback any port errors to the screen such as invalid handle value or failed to connect etc.

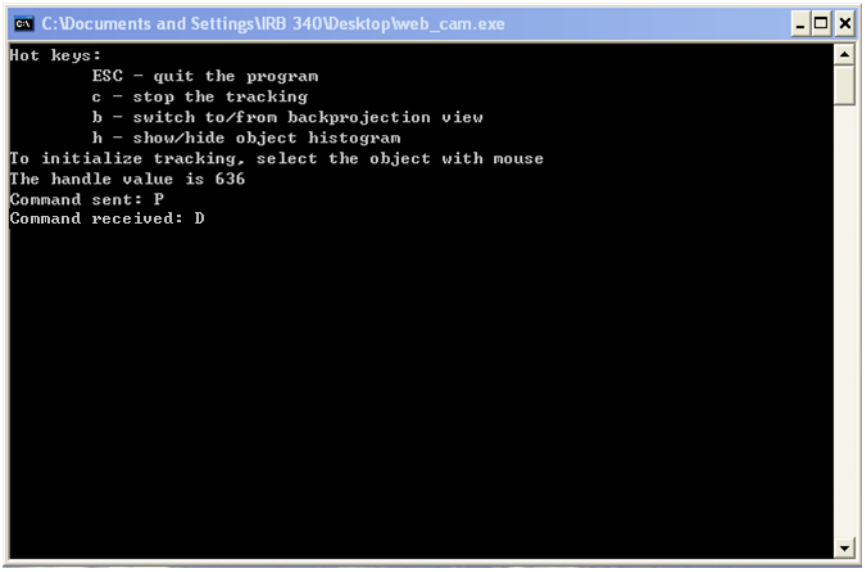


Figure 5-4 Vision Application with complete initialization

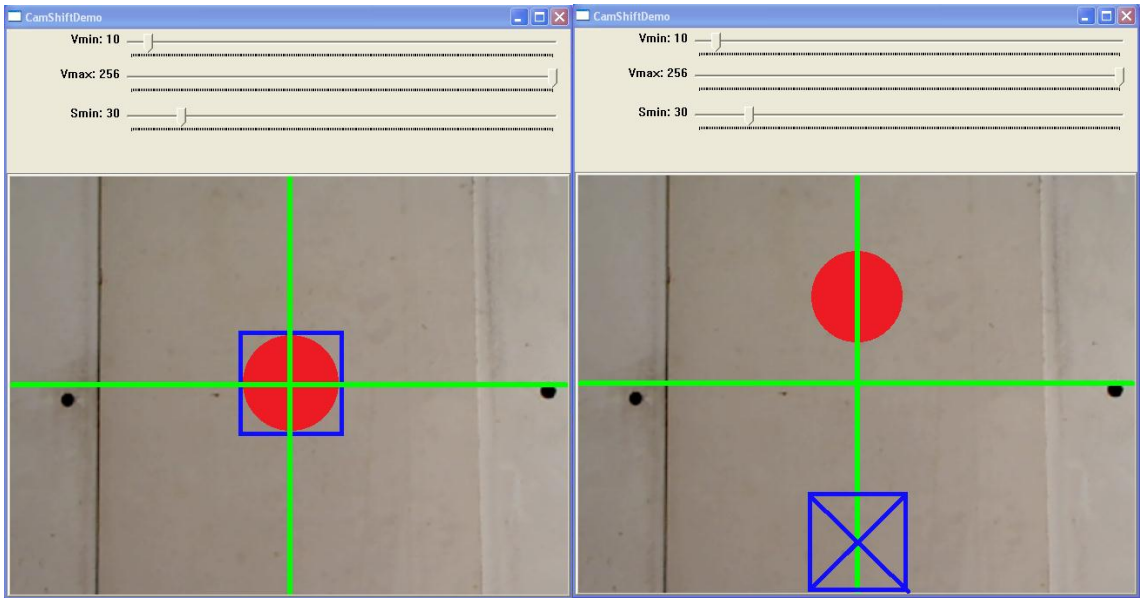


Figure 5-5 Tracking Process

The figure 5-5 demonstrated the testing of target tracking using the MeanShift technique.

This test ensure the application was able to detect a target on the moving conveyor with a rectangle window based on the pre-define sample color. During the test, one with completely different from background color was chosen for the items to avoid distraction. When the rectangle window passed the triggering line (the horizontal green line), it would send a command to trigger the robot to perform tasks and meanwhile returned the rectangle window to the mid-bottom of the screen. If the command was send successfully, it would be acknowledged with the string: “Command sent: P”. (Figure 5-4) As expected, soon as the trig board received the command from the camera, it would echo back the end application and we could see the string on the screen: “Command received: D”. This stage of testing is to ensure the end application was able to write and read the serial port correctly, as well as be interfaced with the trig board’s embedded program.

5.1.4. System Integration and Testing

Having validated the operation of the web-based SCADA application and PLC remote control operation, Flxipicker application, machine vision system, embedded circuit and application, video broadcasting, the individual components were now integrated to perform the complete experiment. In order to test the combined system, the server was booted with the WinCC application, vision application, PickMaster project, OPC server and video broadcasting server loaded. Follow then, sampled the color of the items on the conveyor by using the mouse and draw a rectangle over the item on the captured image. Up to this step, the preparation for the server side was completed. The next step was to boot up the Flexpicker system and set the S4Cplus robot controller to MOTOR ON and AUTOMATIC mode.

Once the embedded circuit was connected to the S4Cplus DI/DO board and the PLC program is loaded, the system is ready to go. The client can now login to the server with the created account and install the software followed by the instructions given on the page. During the

test, we found that the PickMaster may not recognize the command sometimes. This may cause due to the very short delay between the logic 0 and 1, thus, the DI/DO plug-in did not notify the changes of signal.

Figure 5-4 demonstrates the combined system's functionality, demonstrating the remote control output for a PickMaster project.

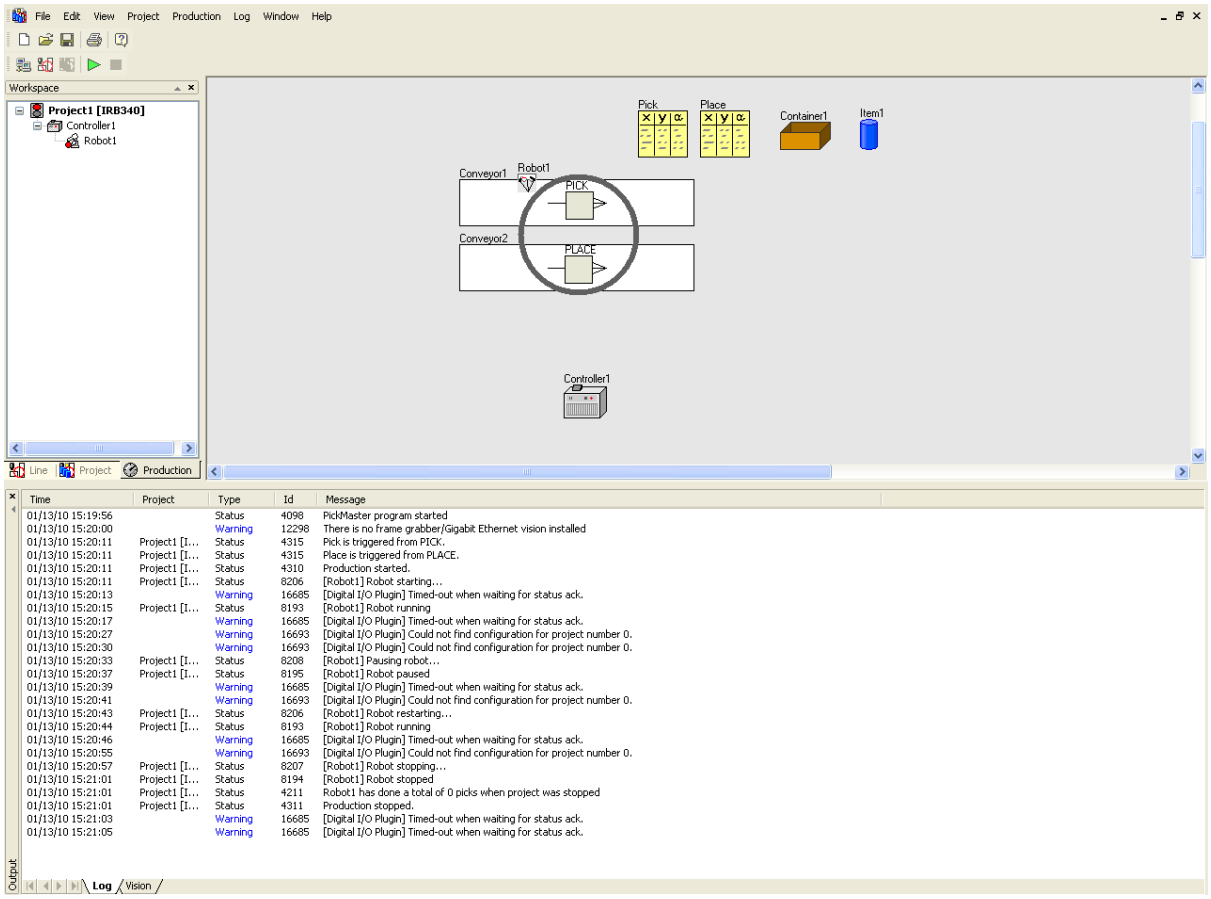


Figure 5-6 PickMaster Demonstrating Reception of the Remote Control Event

Chapter 6 - Conclusion

We have implemented a integrate system for automatic picking and placing using ABB Flexpicker system with the ability to detect moving objects present on the conveyor via the vision system and achieved SCADA system control via a graphical user interface. In the thesis we have presented integration of recent technologies such as robotics, computer network technology, remote control technology, machine vision technology and SCADA system that applied in the pick and place manufacturing cell. This kind of industrial control is type of a distributed network architecture called SNRP (Simple Network Robot Protocol) that simplifies a lot the interaction between the different components of the system such as robots, cameras, conveyor belt, etc. The technique in this thesis demonstrated a very appropriate and low cost method for education, research and even industrial applications.

The developed system was able to demonstrate the capability of achieving supervisory control, networked data communication, data collection, presentation and machine vision system. The main server consisted of WinCC configuration software, Web Navigator server, surveillance broadcasting server, OPC server, and vision program. Each program has its own tasks and was able to manage communication through network protocol. When startup the WinCC program, it will first initialize all the runtime environments, Web Navigator server, OPC drivers, tag variables and process pictures. The WinCC module provided a platform to interface with lower hardware while the Navigator server broadcasting all the process pictures on to the web. The end system can browse the control process and perform operation on the homepage with real-time video stream from the surveillance server. The Flexpicker system on the other hand is controlled by the integrated RIS service. The RIS service realized PLC remote control of PickMaster through PCI 1750 digital I/O card. Once the command is

received, the PickMaster will open and start the corresponding project in the server. The C++ vision end application scans for the pre-define color in each frame, and drew a rectangle on the matched target. The rectangle is able to track the moving target and once the target passed the trigging point, it will activate the IRB340 to perform tasks, meanwhile, reset the position of the rectangle to the starting point for the next target.

Based on the experience gained from the development of the entire system, there are number of improvements that could be made in future work: further develop of real time monitoring and data acquisition for the control system should be performed in order to develop a more flexible, efficiency and low cost manufacturing plant; logging, alarm archiving system and report outputting system for process pictures need to be set up to provide detail information of the running system; develop registration module on the Homepage to allow creating new user account; expand the S7-200 PLC with a network module so that it can communicate with the Flexpicker system through TCP/IP protocol to provide more reliable control, low cost and easy to setup than the digital I/O PCI 1750 card. Feature research could also investigate ways of advancing the vision system to achieve shape and position recognition, multi-target detection; video stream record and download feature to provide trace-back function for dialogistic; another feature could be developed is the closed-loop conveyor system with robot arm to achieve automatic loading and looping assignment.

Chapter 7 - Bibliography

A. Sayouti, F. Qrichi Aniba, H. Medromi. (2008, 11). Remote Control Architecture over Internet Based on Multi agents Systems. *International Review on Computers and Software (I.RE.CO.S) , 3,* pp. 666-671.

AB, A. (2004). IRB 340 Product specification. SWEDEN.

ABB AB, Robotics. (2008-2009). PickMaster User's Guide. Sweden.

ABB. (2004-2007). Technical reference manual - RAPID kernel.

Berry, B. (2008, 4 1). SCADA Tutorial: A Quick, Easy, Comprehensive Guide. Fresno, CA.

Bonev, I. (2001, May). *Delta Parallel Robot — the Story of Success*. Retrieved from ParalleMic: http://www.parallemic.org/Reviews/Review002.html

Bradski, G. R. (1998). Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Jounal* , 1-15.

Cheng, Y. (1995). Mean shift, Mode Seeking, and Clustering. *IEEE Transactions on PAMI , 17* (8), 790-799.

Chou Wusheng , Wang Tianmiao. (2003). Internet Based Remote Robotic Experimental System. *Chinese High Technology Letters , 13* (08).

Comaniciu D, Meer P. (1999). Distribution Free Decomposition of Multivariate Data [J]. *Pattern Analysis & Applications , 2* (1), 22-30.

Comaniciu D, Meer P. (1999). Mean Shift Analysis and Applications [C]. *The Proceedings of the Seventh IEEE International Conference on Computer Vision* , 1197-1203.

Comaniciu D, Ramesh V, Meer P. (2003). Kernel-Based Object Tracking [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 25* (5), 564-575.

Comaniciu D,Meer P. (2002). Mean Shift: A Robust Approach toward Feature Space Analysis [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 24* (5), 603-619.

Darkwet Network/Moonware Studios. (n.d.). *WebcamXP*. Retrieved from WebcamXP:

http://www.webcamxp.com/home.aspx

Fan, R. (2006). Internet based SCADA: A new approach using JAVA and XML. *ELEKTRON JOURNAL- SOUTH AFRICAN INSTITUTE OF ELECTRICAL ENGINEERS* , 43-45.

Fashing M, Tomasi C. (2005). Mean Shift is a Bound Optimization [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 27* (3), 471-474.

Fukunaga K, Hostetler L. (1975). The Estimation of the Gradient of A Density Function, with Applications In Pattern Recognition. *Information Theory,IEEE Transaction on , 21* (1), 32-40.

Gary Bradski and Adrian Kaehler. (2008). *Learning OpenCV.* United States of America: O'Reilly Media, Inc.

Guglielmetti, P. (1994). Model-based control of fast parallel robots: A global approach in operational space. *Ph.D. thesis* . Swiss Federal Institute of Technology Lausanne (EPFL), 1228.

Iwan Ulrich,Illah Nourbakhsh . (2000, 7). Appearance-Based Obstacle Detection with Monocular Color Vision . *Proceedings of the National Conference on Artificial Intelligence* , pp. 886-871.

J.M.Sebastian, R.Saltaren, R.Aracil,J. Sanpedro. (2005). RoboTenis: optimal design of a parallel robot with high performance. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on* (pp. 2134- 2139). Colombia: Facultad de Ingenieria Electronica.

Krut, S., Nabat, V., Company, O., Pierrot, F. (2004). A high-speed parallel robot for Scara motions. *Robotics and Automation, 2004. Proceedings. ICRA '04.2004 IEEE International Conference on, 4*, pp. 4109- 4115.

Mahn, C. (2005). Open Vs. Closed Communication Protocols Why Open Protocols are the Way of the Future. Gurnee, IL.

Prof. Dr.-Ing.Gruhler Gerhard. (2003). *Remote Control of CAN-based Industrial Equipment Using Internet Technologies.* University of Applied Sciences Reutlingen. Germany: Steinbeis Technology Transfer Center Automation.

Quanyu Wang; Siyin Liu; Zhe Wang. (2006). A New Internet Architecture for Robot Remote Control. *Intelligent Robots and Systems,2006 IEEE/RSJ International Conference on*, (pp. 4989-4993).

R, H. (2007, 6 29). *Bayesian Mean Shift Face Tracker*. Retrieved from http://www.robinhewitt.com/research/track/index.html

R.Clavel. (1988). DELTA:A fast robot with parallel geometry. *18th International Symposium on Industrial Robot* , 91-100.

Raghavan, V. (2008, 12 19). *Camera design for machine vision.* Retrieved from Video Imaging DesignLine : http://www.videsignline.com/showArticle.jhtml;jsessionid=UTNTPXKB3IIIZQE1GHPSKH4ATMY32JVN?articleID=212501328&queryText=Camera+design+for+machine+vision

Ronen Ben-Horin,Moshe Shoham,Shlomo Djerassi. (n.d.). *Kinematics, Dynamics and construction of a planarly actuated parallel robot*. Retrieved from Kinematics, Dynamics and construction of a planarly actuated parallel robot: http://robotics.technion.ac.il/projects/ronen_project.html

Shah, A. (2008, 11). *Serial Communication Basic*. Retrieved from DNA TECHNOLOGY: http://www.dnatechindia.com/index.php/Tutorials/8051-Tutorial/Serial-Communication-Basic.html

Siemens AG. (2008, 09). SIMATIC WinCC System Description. GERMANY.

SIEMENS. (n.d.). SIMATIC MicroComputing User Manual.

Staicu St. & Carp-Ciocardia D.C. (2003). Dynamic Analysis of Clavel's DELTA Parallel Robot. 1-6.

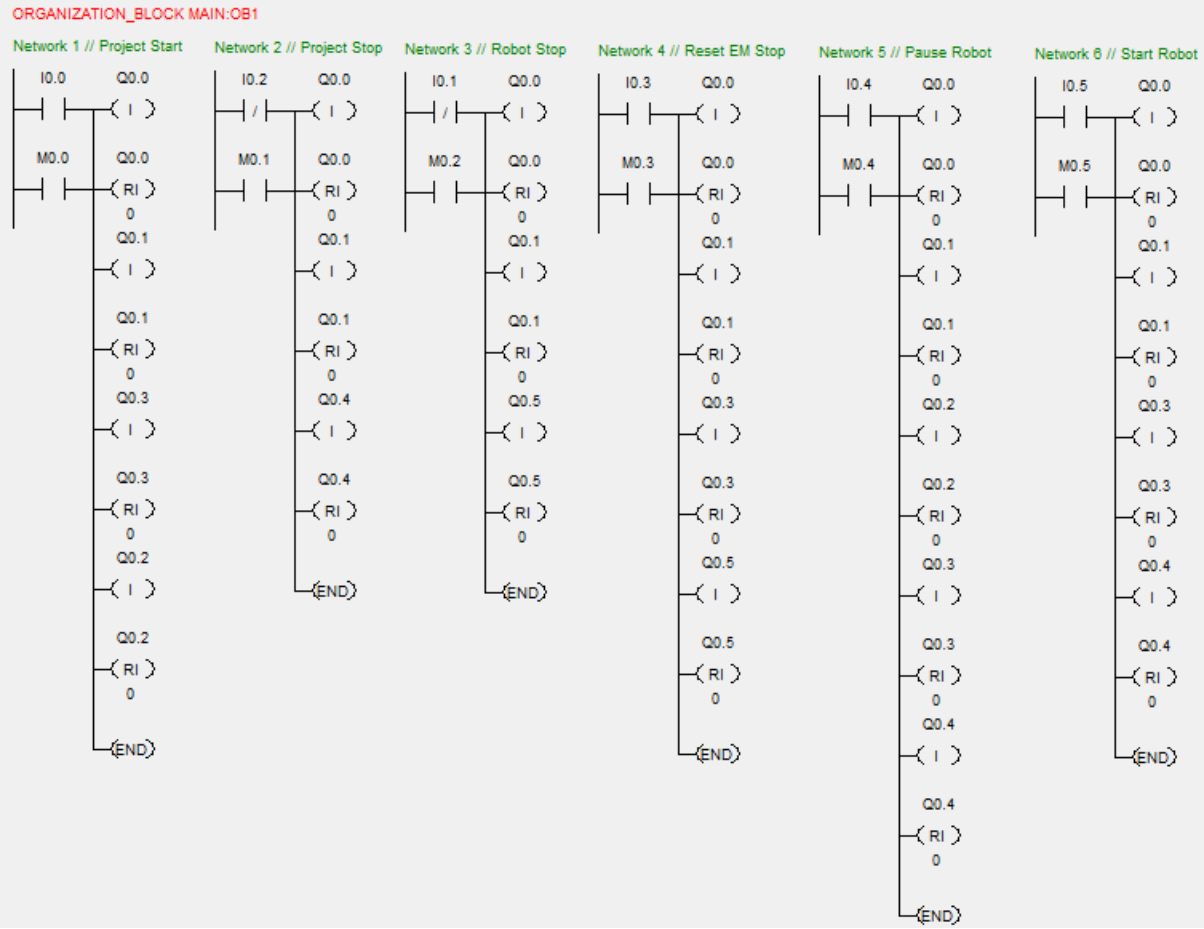
Zhou, X. (2004). Research of communication between S7-200 series PLC and supervision computer. *CONTROL AND AUTOMATION* , 5-7 .

Zhou,Xiaoping Jiang,Jianfang Su,Shaoyu Chen,Xun. (2004). Research of communication between S7-200 series PLC and supervision computer. *CONTROL AND AUTOMATION* , pp. 5-7.

Zuech, N., & Miller, R. K. (1989). In *Machine Vision* (p. 89). Lilburn, GA: Fairmont Press.

Chapter 8 - Appendix

8.1. PLC Code



8.2. Mean-Shift Code

Main.CPP

```
#include "stdafx.h"

#include <cv.h>

#include <cxcore.h>

#include <atlstr.h>

#include <highgui.h>
```

```
#include <iostream>

#include <windows.h>

#include <ctype.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

#include <string>

//=================================================================

IplImage *image = 0, *hsv = 0, *hue = 0, *mask = 0, *backproject = 0, *histimg = 0;

CvHistogram *hist = 0;

int backproject_mode = 0;

int select_object = 0;

int track_object = 0;

int show_hist = 1;

CvPoint origin;

CvRect selection;

CvRect track_window;

CvBox2D track_box;

CvConnectedComp track_comp;

int hdims = 16;

float hranges_arr[] = {0,180};
```

```
float* hranges = hranges_arr;

int vmin = 10, vmax = 256, smin = 30;

//======================== Function Declerations ========================

void Serial_Comm_init(void);

void SerialPutc(HANDLE *m_hCom);

void Comm (HANDLE *ptr);

// ====================== Serial Port Declerations ========================

char txString[10]="P";

CString m_sComPort;

HANDLE  m_hCom;

COMMTIMEOUTS m_CommTimeouts;

DCB m_dcb;

void SerialPutc(HANDLE *hCom){// ----- write byte to serial -----

        BOOL Write = false;

        int numBytes=0, len = 0;

        len = strlen(txString) + 1;

        Write = WriteFile(*hCom, &txString, len, (LPDWORD)&numBytes, NULL); //
writefileA required to prevent the default to WriteFileW

        if (Write){

                printf("Command sent: %s\n", txString);

        }else{

                printf("failed to pass command");
```

```
        }

        return;             // something to do with unicode etc

}

void Comm (HANDLE *ptr) {

        SerialPutc(ptr);

}

void Serial_Comm_init(){ // ------------ SETS UP COM4 --------------------

        BOOL m_bPortReady = TRUE;

        m_sComPort = "COM4";

        m_hCom = CreateFile(m_sComPort,  //         constructs a handle for the port

        GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING,  0,
//FILE_FLAG_NO_BUFFERING,    //0,   // no overlapping I/O  ?? 0);

        if (m_hCom == INVALID_HANDLE_VALUE){

                printf("failure to connect to COM4\n failed to create handle \n - exit(1)\n");

                getchar();

                m_bPortReady = FALSE;}

                if(m_bPortReady){

                m_bPortReady = SetupComm(m_hCom, 128, 128);  // sets the buffer sizes

                if(!m_bPortReady){ printf("failed to setup buffers");

                }}

        if(m_bPortReady){

                m_bPortReady = GetCommState(m_hCom, &m_dcb); // identifies the current
settings - allocating the defaults
```

```
        if(!m_bPortReady){ printf("get comm state failed");

        }}

    if(m_bPortReady){

        m_dcb.BaudRate = 9600;

        m_dcb.ByteSize = 8;    // ? only sending eight bits as deined within the
hyperterminal settings

        m_dcb.Parity   = NOPARITY;

        m_dcb.StopBits = ONESTOPBIT;

        m_dcb.fAbortOnError = TRUE;

        m_bPortReady = SetCommState(m_hCom, &m_dcb);

        if(!m_bPortReady){

            printf("setcommState failed");

        }}

        if(m_bPortReady){

        m_bPortReady = GetCommTimeouts (m_hCom, &m_CommTimeouts);

        if(!m_bPortReady){ printf("GetCommTimeouts failed");

            m_bPortReady = FALSE;

        }}

    if(m_bPortReady){

        m_CommTimeouts.ReadIntervalTimeout = 50;

        m_CommTimeouts.ReadTotalTimeoutConstant = 50;

        m_CommTimeouts.ReadTotalTimeoutMultiplier = 10;
```

```
                //m_commTimeouts.WriteTotalTimeoutMultipler = 10;
                m_CommTimeouts.WriteTotalTimeoutConstant = 50;
                m_CommTimeouts.WriteTotalTimeoutMultiplier = 10;
                m_bPortReady = SetCommTimeouts (m_hCom, &m_CommTimeouts);
                if(!m_bPortReady){ printf("failed to SetCommTimeout");
                }}// setup of serial port comm complete
        printf("The handle value is %d \n", m_hCom);
}
//========================================================================
void on_mouse( int event, int x, int y, int flags, void* param )
{
   if( !image )
      return;
   if( image->origin )
      y = image->height - y;
   if( select_object )
   {
      selection.x = MIN(x,origin.x);
      selection.y = MIN(y,origin.y);
      selection.width = selection.x + CV_IABS(x - origin.x);
      selection.height = selection.y + CV_IABS(y - origin.y);
      selection.x = MAX( selection.x, 0 );
```

```
        selection.y = MAX( selection.y, 0 );
        selection.width = MIN( selection.width, image->width );
        selection.height = MIN( selection.height, image->height );
        selection.width -= selection.x;
        selection.height -= selection.y;
    }
    switch( event )
    {
    case CV_EVENT_LBUTTONDOWN:
        origin = cvPoint(x,y);
        selection = cvRect(x,y,0,0);
        select_object = 1;
        break;
    case CV_EVENT_LBUTTONUP:
        select_object = 0;
        if( selection.width > 0 && selection.height > 0 )
            track_object = -1;
        break;
    }}
CvScalar hsv2rgb( float hue )
{
    int rgb[3], p, sector;
```

```
    static const int sector_data[][3] = {{0,2,1}, {1,2,0}, {1,0,2}, {2,0,1}, {2,1,0}, {0,1,2}};
    hue *= 0.033333333333333333333333333333333f;
    sector = cvFloor(hue);
    p = cvRound(255*(hue - sector));
    p ^= sector & 1 ? 255 : 0;
    rgb[sector_data[sector][0]] = 255;
    rgb[sector_data[sector][1]] = 0;
    rgb[sector_data[sector][2]] = p;
    return cvScalar(rgb[2], rgb[1], rgb[0],0);
}
int main( int argc, char** argv ) {
    CvCapture* capture = 0;
    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );
    if( !capture ) {
        fprintf(stderr,"Could not initialize capturing...\n");
        return -1;
    }
    printf( "Hot keys: \n"
        "\tESC - quit the program\n"
```

```
        "\tc - stop the tracking\n"
        "\tb - switch to/from backprojection view\n"
        "\th - show/hide object histogram\n"
        "To initialize tracking, select the object with mouse\n" );
//          cvNamedWindow( "Histogram", 1 );
            cvNamedWindow( "CamShiftDemo", 1 );
            cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 640); //640
            cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 480);//480
            cvResizeWindow("CamShiftDemo", 640,480);
            cvSetMouseCallback( "CamShiftDemo", on_mouse, 0 );
            cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
            cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
            cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );
            HANDLE *COM4_ptr;
            COM4_ptr = &m_hCom;
            Serial_Comm_init();
    for(;;) {
        IplImage* frame = 0;
        int i, bin_w, c;
        frame = cvQueryFrame( capture );
        if( !frame )
            break;
```

```
if( !image ) {
    image = cvCreateImage( cvGetSize(frame), 8, 3 );
    image->origin = frame->origin;
    hsv = cvCreateImage( cvGetSize(frame), 8, 3 );
    hue = cvCreateImage( cvGetSize(frame), 8, 1 );
    mask = cvCreateImage( cvGetSize(frame), 8, 1 );
    backproject = cvCreateImage( cvGetSize(frame), 8, 1 );
    hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 );
    histimg = cvCreateImage( cvSize(320,200), 8, 3 );
    cvZero( histimg );
}
cvCopy( frame, image, 0 );
cvCvtColor( image, hsv, CV_BGR2HSV );
cvLine(image, cvPoint(0,240), cvPoint(640,240), cvScalar(0,255,0), 5);
cvLine(image, cvPoint(320,0), cvPoint(320,480), cvScalar(0,255,0), 5);
if( track_object ) {
    int _vmin = vmin, _vmax = vmax;
    cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
            cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
    cvSplit( hsv, hue, 0, 0, 0 );
    if( track_object < 0 )
    {
```

```
    float max_val = 0.f;
    cvSetImageROI( hue, selection );
    cvSetImageROI( mask, selection );
    cvCalcHist( &hue, hist, 0, mask );
    cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
    cvConvertScale( hist->bins, hist->bins, max_val ? 255. / max_val : 0., 0 );
    cvResetImageROI( hue );
    cvResetImageROI( mask );
    track_window = selection;
    track_object = 1;
    cvZero( histimg );
    bin_w = histimg->width / hdims;
    for( i = 0; i < hdims; i++ ) {
       int val = cvRound( cvGetReal1D(hist->bins,i)*histimg->height/255 );
       CvScalar color = hsv2rgb(i*180.f/hdims);
       cvRectangle( histimg, cvPoint(i*bin_w,histimg->height),
       cvPoint((i+1)*bin_w,histimg->height - val), color, -1, 8, 0 );
    }}
cvCalcBackProject( &hue, backproject, hist );
cvAnd( backproject, mask, backproject, 0 );
cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
          &track_comp, &track_box );*/
```

```
        cvMeanShift( backproject, track_window,
        cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
                                    &track_comp);
        track_window = track_comp.rect;
        if( backproject_mode )
           cvCvtColor( backproject, image, CV_GRAY2BGR );
        if( image->origin )
           track_box.angle = -track_box.angle;
cvRectangle(image,cvPoint(track_window.x,track_window.y),cvPoint(track_window.x+track
_window.width,track_window.y+track_window.height) , CV_RGB(0,0,255),2, CV_AA , 0 );
cvLine( image,cvPoint(track_window.x,track_window.y),cvPoint(track_window.x+track_win
dow.width,track_window.y+track_window.height),CV_RGB(0,0,255),2, CV_AA , 0);
cvLine( image,cvPoint(track_window.x+track_window.width,track_window.y),cvPoint(track
_window.x,track_window.y+track_window.height),CV_RGB(0,0,255),2, CV_AA , 0);
cvPoint(track_window.x+track_window.width/2,track_window.y+track_window.height/2) }
          if(track_window.y+track_window.height/2 >= image->origin+240){
                    Comm(COM4_ptr);
                    track_window.y = 0;
                    track_window.x = image->width/2;
                    int i = track_window.x + track_window.width/2;
                    i = image->width/2;
               }
     if( select_object && selection.width > 0 && selection.height > 0 ) {
```

```
        cvSetImageROI( image, selection );
        cvXorS( image, cvScalarAll(255), image, 0 );
        cvResetImageROI( image );
    }
    cvShowImage( "CamShiftDemo", image );
    cvShowImage( "Histogram", histimg );
    c = cvWaitKey(10);
    if( (char) c == 27 )
        break;
    switch( (char) c ) {
    case 'b':
        backproject_mode ^= 1;
        break;
    case 'c':
        track_object = 0;
        cvZero( histimg );
        break;
    case 'h':
        show_hist ^= 1;
        if( !show_hist )
            cvDestroyWindow( "Histogram" );
        else
```

```
            cvNamedWindow( "Histogram", 1 );
          break;
       default: ;
       }}
   cvReleaseCapture( &capture );
   cvDestroyWindow("CamShiftDemo");
   CloseHandle(m_hCom);
   return 0;
}
#ifdef _EiC
main(1,"camshiftdemo.c");
#endif
```

8.3. Webcam Interface HTML Code

```
<html><head></head><body topmargin=0 leftmargin=0><center>
<table cellspacing cellpadding width="320" height="240"><tr><td width="320" height="240">
<img src="http://130.123.255.138:8080/loading.jpg" class="webcam" id="webcam1" onmousedown="PTZMouseDown1(event)" width="320" height="240" alt="Live Stream" />
<script type="text/javascript"><!--
currentCamera1= 1;
errorimg1= 0;
document.images.webcam1.onload = DoIt1;
```

```
document.images.webcam1.onerror = ErrorImage1;

function LoadImage1(){

    uniq1 = Math.random();

    document.images.webcam1.src = "http://130.123.255.138:8080/cam_" + currentCamera1 + ".jpg?uniq="+uniq1;

    document.images.webcam1.onload = DoIt1;}

function PTZMouseDown1(event) {

    var IE = document.all?true:false;

    var x,y;

    var myx,myy;

    var myifr = document.getElementById("_iframe-ptz");

    tp = getElPos1();

    myx = tp[0];

    myy = tp[1];

    if(IE){

    var scrollX = document.documentElement.scrollLeft ? document.documentElement.scrollLeft : document.body.scrollLeft;

    var scrollY = document.documentElement.scrollTop ? document.documentElement.scrollTop : document.body.scrollTop;

    x = event.clientX - myx;

    y = event.clientY - myy;

    } else {

    x = e.pageX - myx;
```

```
    y = e.pageY - myy;

    }

    if ((width_array[currentCamera1] != null) && (width_array[currentCamera1] > 0)) x =
Math.round((x * 400) / width_array[currentCamera1]);

    if ((height_array[currentCamera1] != null) && (height_array[currentCamera1] > 0)) y =
Math.round((y * 300) / height_array[currentCamera1]);

    if (x > 400) x = 400;

    if (y > 300) y = 300;

    if (myifr != null) myifr.src = "http://130.123.255.138:8080/ptz?src=" + currentCamera1
+ "&moveto_x=" + x + "&moveto_y=" + y +"";

    return true;}

function getElPos1(){

        el = document.images.webcam1;

        x = el.offsetLeft;

        y = el.offsetTop;

        elp = el.offsetParent;

        while(elp!=null) { x+=elp.offsetLeft;

            y+=elp.offsetTop;

            elp=elp.offsetParent;

          }

        return new Array(x,y);

}

function ErrorImage1(){
```

```
        errorimg1++;
        if (errorimg1>3){
            document.images.webcam1.onload = "";
            document.images.webcam1.onerror = "";
            document.images.webcam1.src = "offline.jpg";
            }else{
              uniq1 = Math.random();
          document.images.webcam1.src = "http://130.123.255.138:8080/cam_" +
currentCamera1 + ".jpg?uniq="+uniq1;}}
function DoIt1(){
        errorimg1=0;
        window.setTimeout("LoadImage1();", 40);}
//--></script></td></tr></table></center></body></html>
```

8.4. PIC18F4520 Code

```
#include <p18f4520.h>
#include <usart.h>
#include <delays.h>
void configure_pins(void);
void main(void) {
        configure_pins();
        OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_HIGH, 51);
```

```
        ADRESH = 0; //Resetting the ADRES value register
        ADRESL = 0;
        while(1) {
                unsigned char c;
                while(BusyUSART());
                c = ReadUSART();
                if (c=='P'){
                while(BusyUSART());
                putcUSART('d');
                LATDbits.LATD0 =1;
                Delay10KTCYx(5); // 0.1s or 100ms
                LATDbits.LATD0 =0;
                }
   }
        CloseUSART();
}
void configure_pins() {
         OSCCONbits.IRCF2 = 1;
         OSCCONbits.IRCF1 = 1;
         OSCCONbits.IRCF0 = 1;
        LATD = 0; // Clear port D data latch
        TRISD = 0b11110000; // This binary value sets: RD4-7 inputs, RD0-3 outputs
```

```
        TRISC = 0b11111111;             // RX & TX setup

        RCSTA = 0b10010000;             // 0b10010000 SPEN = 1 CREN = 1

}
```