2-D LOCATION POINTING SYSTEM FOR INDIVIDUAL COMPONENT ON DEVICE UNDER TEST BY USING LABVIEW (SOFTWARE AND MULTIPURPOSE HOLDER)

CHUNG KA SIEW

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering (Hons.) of Electrical and Electronic Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2012

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : ___________________________

Name : Chung Ka Siew

ID No. : 08UEB04716

Date : ___________________________

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“2-D LOCATION POINTING SYSTEM FOR INDIVIDUAL COMPONENT ON DEVICE UNDER TEST BY USING LABVIEW (SOFTWARE AND MULTIPURPOSE HOLDER)”** was prepared by **CHUNG KA SIEW** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _______________________

Supervisor : Mr. See Yuen Chark

Date : _______________________

Specially dedicated to

my beloved family,

supervisor Mr. See Yuen Chark,

and project partners Ch’ng Khai Chiah and Cheng Xuan Teng.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr. See Yuen Chark for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement in completing my final year project. Their unflinching courage and conviction are much appreciated.

Last but not least, my deepest thank also goes to my project partners, Mr. Ch’ng Khai Chiah and Mr. Cheng Xuan Teng, who have given me the opportunity to work with them and to gain the exposure in the development of research. I am grateful to their sharing of knowledge and support throughout this project.

2-D LOCATION POINTING SYSTEM FOR INDIVIDUAL COMPONENT ON DEVICE UNDER TEST BY USING LABVIEW (SOFTWARE AND MULTIPURPOSE HOLDER)

ABSTRACT

Time is an important factor to determine the market performance of a product, where accelerating the process of test development is one of the solutions. Probing is involved in most of the testing process. Hence, designing a system to assist the engineers in component searching during probing process is critical. In this project, a communication system between the programming tool – LabVIEW, and the PCB drawing tool – EAGLE is developed by using Windows scripting – batch file, as an intermediate link. The user is able to select the desired testing component by clicking its symbol in its schematic, and a laser pointer will be driven to the exact location of the component in real time, with proper calibration of the XY coordinates and motor steps. By pointing the laser pointer to the component, the user can easily observe the component's location on a PCB. In addition, the record of all selected components in the system will be saved and the user is given the option whether to add a comment on each selection. In addition, a multipurpose flexible holder is integrated in this system as an additional feature. It is designed to hold any object and lock its movement in a fixed position, which can be applied in probing or soldering process. Speech recognition is incorporated when the user wants to lock or unlock the movement of this holder in holding an object. It is to counter the problem when the user has no free hand to click on the lock or unlock button. Throughout this project, NI sbRIO platform is used as the central control device to interact with the input/output processing whereas NI LabVIEW software is used as the programming tool to develop the monitoring and control program in this system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

B	Magnetic Field, T
μ_0	Permeability of Free Space, T.m/A
I	Current, A
N	Number of Loops
L	Length of Solenoid, m

CAM	Computer-Aided Manufacturing
CC	Carbon Copy
CSV	Comma-Separated Values
DOF	Degree of Freedom
DUT	Device under Test
EAGLE	Easily Application Graphical Layout Editor
IC	Integrated Circuit
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
PCB	Printed Circuit Board
POP	Post Office Protocol
SMTP	Simple Mail Transfer Protocol
TLS	Transport Layer Security
ULP	User Language Program
VBScript	Visual Basic Script
VI	Virtual Instrument

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Engineers in test development department play an important role in validating the functionality of the products to meet industrial specifications and business needs including scheduling and cost. The key integrity in their working environment is about the effectiveness and efficiency in completing their assigned tasks.

One of the most important tasks is to test the functionality of the semiconductor components, which is commonly mentioned as Devices under Test (DUT). These components are usually combined with other semiconductor components and arranged on a wafer. To perform the testing, they are contact-connected by using a probe, normally in the form of a needle and being fastened to a carrier device such as a probe holder. An oblique position with respect to the surface of the DUT is needed, where an improved flexible probe holder is important in this case. (Kiesewetter, Kreissig, & Kanev, 2009)

In order to achieve greater reliability and performance of the integrated circuitry, the number of semiconductor components packed on a mask is significantly increased and hence its functions will be improved simultaneously. It is because internal chip connections are essentially more reliable than externally wired connections and also the signal speed is greater for internal chip connections. In addition, the lower the number of chips and the higher their density, the more chips can be accommodated on a printed circuit board (PCB). Subsequently, the complexity of a single chip is amplified as circuit densities increase and more functions are packed into the chip. (Wong, 2010) This rapid pace of technology innovation has been predicted by Intel co-founder Gordon Moore nearly 40 years ago, known as “Moore’s Law”, which states that transistor density on integrated circuits (IC) doubles about every eighteen months. (Moore, 2003) Figure 1.1 shows the plot of transistor counts against dates of introduction. The line corresponds to exponential growth with transistor count doubling every eighteen months. ("Moore's Law 40th Anniversary," 2005)

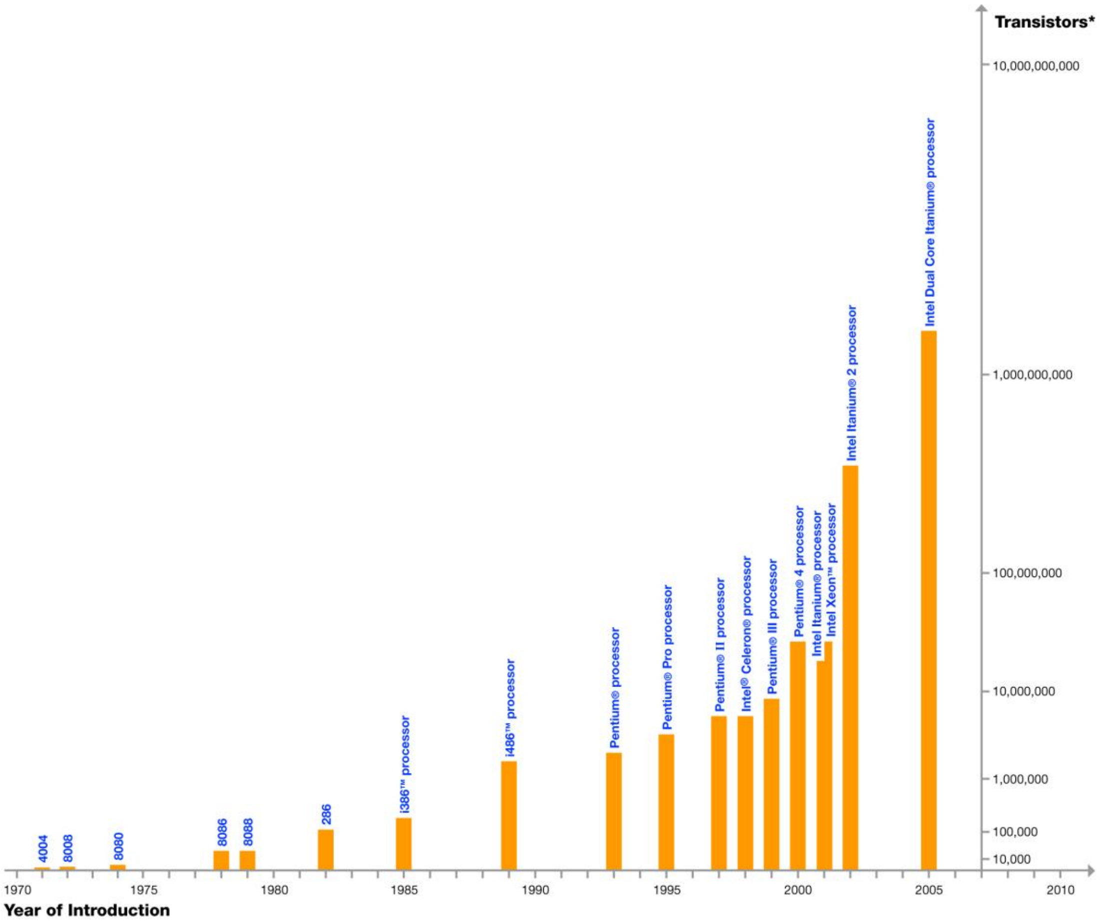


Figure 1.1: Plot of Transistor Counts against Dates of Introduction

1.2 Problem Statement and Motivation

As the complexity of the PCB is greater than before, the process of probing is undeniably time consuming nowadays as the engineers have to be really focused to search for the location of individual component to be probed on the DUT, where this problem is worsened when the DUT is a complicated PCB. Engineers are not supposed to over spend their time in searching for a single component to be tested, but more on the professional problem solving tasks. This problem should not be a reason to double up engineers' work load.

Furthermore, another problem could occur during probing, which is to fix the position of the test probe properly while contacting it to the component. Several probe positioners are produced by Agilent, such as N2784A one-arm probe positioner, N2786A two-leg probe positioner and N2787A 3D probe positioner, where some of them will be discussed in Section 2.4. However, the price of all these devices is not in an affordable range for users. (Agilent, 2009)

All the problems stated have become a motivation to design a system that avoid user from the time-consuming component searching process on the real board, but with only one click on the schematic in personal computer. It is also a motivation to design a multipurpose flexible holder to get users a hand-free helper whenever they are in need to hold something in fixed position, such as test probe or soldering lead. It provides a solution for people, especially engineers, to have a hassle free working environment in their laboratory.

1.3 Project Objectives

There are two sections in this project, where two main objectives are to be achieved. The first section is to create an interface on LabVIEW that when a user clicks on any component from the schematic, the coordinate of the component on the board will be obtained to LabVIEW. This data will be processed to get the true coordinates of the component in a pre-defined area so that the laser pointer will be driven there and the selected component will be pointed out. It is to speed up the process of searching the component especially on a complicated board. To achieve it, an effective communication between LabVIEW and the PCB drawing software is needed. In interest of this objective, some tasks are set to be fulfilled as following:

- To learn to operate LabVIEW as a programming tool
- To study pre-defined functions available in LabVIEW that enable it to connect to external application
- To study the different tools in PCB drawing software, such as its scripting files, user language program (ULP) and also the gerber files that help to export the coordinate and image file to LabVIEW.
- To study the windows scripting which will help to link between LabVIEW and the PCB drawing software, such as batch file, Jscript, PowerShell and VBScript.
- To study configurations of various email servers to be an additional feature for record keeping.

The second section of this project is to build a multipurpose flexible holder, which is not only designed to hold the test probe during probing process, but also to hold other pen-shaped devices when necessary, such as soldering lead during soldering process and etc. The holder is activated and de-activated by using a human sound signal, such as the words “lock” and “unlock”. The tasks need to be achieved in order to meet this objective:

- To study different types of probe holder or positioner available in the market, where their cost is one of the target of research.
- To study the mechanical design to have a fully flexible holder.
- To study the working of speech recognition in LabVIEW

1.4 Project Scopes

Under a limited time frame, the extent of this project needs to be set clearly without compromising the basic requirements needed to be achieved as stated in the project objectives.

This project is targeting for DUT with their schematic files (*.sch) and board files (*.brd) available in a PCB drawing software in computer. Without these two files, the program in LabVIEW is not able to generate the coordinate files of the components on the DUT. In addition, there will be only specific PCB drawing software to be used in this project, which means that only specific PCB drawing software can be communicated with LabVIEW. It is because different PCB drawing software will have their individual scripting language. Different set of command needs to be incorporated in LabVIEW in order to interact with all the software, which will be too vast to be done within the time frame. Furthermore, the size of the DUT to be tested is limited to 18 cm x 30 cm.

For the multipurpose flexible holder, the user will be required to position the holder manually, and then lock it by using voice recognition. There will be only one holder produced in this project; hence only one probe can be hold at a time.

CHAPTER 2

LITERATURE REVIEW

2.1 LabVIEW

Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) is a graphical programming environment used to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart, developed by National Instruments. It is integrated with thousand of hardware devices and it also provides hundreds of built-in libraries for advanced analysis and data visualization. Its platform is scalable across multiple targets and operating systems. ("Product Information: What is NI LabVIEW," 2011) Figure 2.1 shows the screenshot of a simple LabVIEW program.

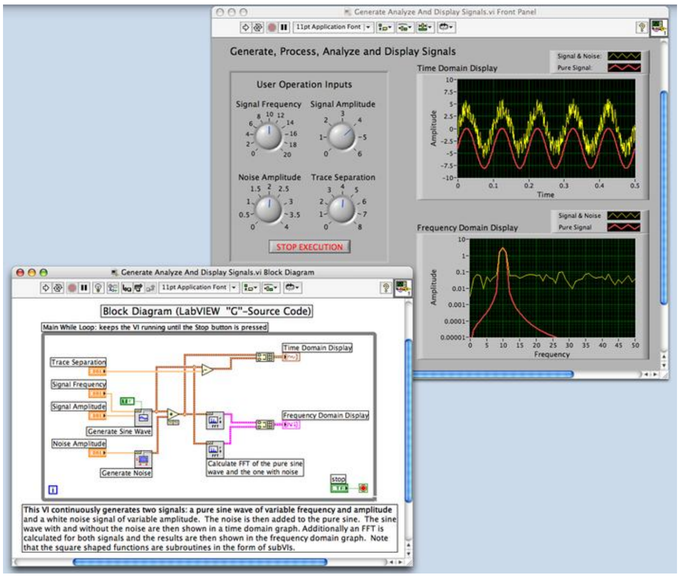


Figure 2.1: Screenshot of a simple LabVIEW program

Some of the common applications of LabVIEW is data acquisition, instrument control and industrial automation on a variety of platforms including Microsoft Windows, various versions of UNIX, Linux and Mac OS X.

LabVIEW programs or subroutines are called virtual instruments (VIs). Each VI has three components, which are a block diagram, a front panel and a connector panel. The front panel is the user interface, and at the same time the controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. Thus, a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the input and outputs for the given node through the connector panel. The connector panel is used to represent the VI in the block diagrams of other, calling VIs. It means that each VI can be easily tested before being embedded as a subroutine into a larger program.

LabVIEW's graphical user interface allows non-programmers to build programs by dragging and dropping virtual representations of lab equipment which they are more familiar with. The programming environment of LabVIEW, with the inclusion of examples and documentation, makes it simple to create small applications. However, it is important that programmers have an extensive knowledge of the special LabVIEW syntax and the topology of its memory management when it comes to complex coding.("Building a Stand-Alone Application," 2009)

2.2 EAGLE

Easily Application Graphical Layout Editor (EAGLE) is flexible and expandable EDA software tools used for designing electronic systems such as PCB and IC. It includes schematic capture, board layout and autorouter. In addition, ULP allows features such as simulation and 3-D visualization. (Cadsoft, 2011b) Figure 2.2 shows the screenshot of a simple EAGLE program.

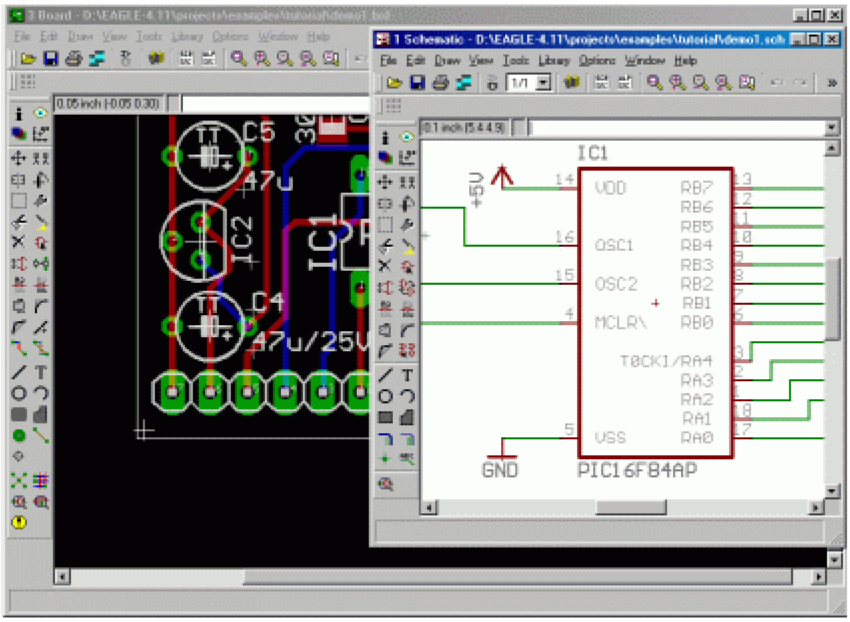


Figure 2.2: Screenshot of a simple EAGLE program

Schematic capture in EAGLE means a schematic editor used to design circuit diagrams, where parts can be placed on many sheets and connected together through ports. (Seidle, 2008)

Board layout is generated from the schematic, allowing back annotation to the schematic and routing of traces according to the users' design.

Autorouter allows automatic connections of traces based on the definition in the schematic. It saves users time to route the connections by themselves but it usually results in inefficient board layout.

Furthermore, EAGLE is integrated with other powerful functions, such as CAM Processor, User Language Program (ULP) and Scripting files, where users can manage them through buttons as circled in the screenshot of EAGLE in Figure 2.3. The details of these functions will be discussed in the following sub sections.

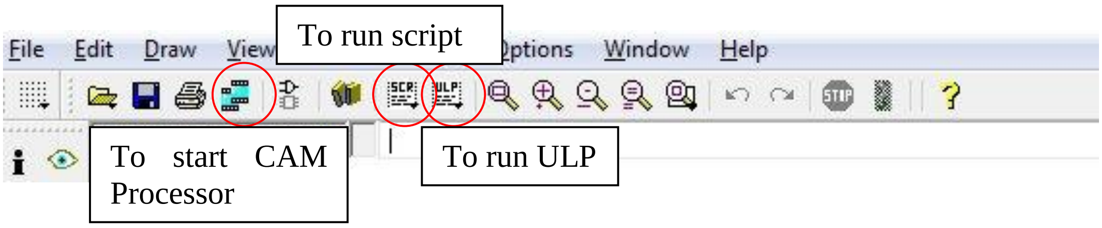


Figure 2.3: Screenshot of Location to Manage CAM Processor, ULP and Script

2.2.1 CAM Processor

The Computer-aided manufacturing (CAM) Processor allows users to output any combination of layers to a device or file. It enables users to combine several sets of parameter settings to form a CAM Processor Job, which can be used to produce a complete set of output files with a single click of a button. The EAGLE CAM Processor can be used to write Gerber files, which are the file used by PCB industry to describe the images of a PCB (copper layers, solder mask, legend, drill holes and etc.).

Gerber files are typically produced by PCB designers and sent to PCB fabricators where they are loaded into a CAM system to prepare data for each step of the PCB production process. They transfer the layer information from CAD to CAM in this workflow. They are used to specify drilled hole information that can be viewed as image layers too. In addition, those files are also used to drive quality control machines such as automated optical inspection. ("PCB Layout Data,")

There are two versions of Gerber format, which are RS-274X (the extended version) and RS-274-D (the standard version). RS-274X is a 2D bi-level vector image description format, formed by human readable ASCII format. It consists of a sequence of commands and coordinates. Its imaging primitives are line draw, flash predefined shapes at a given location and outline fill. Positive and negative objects can be combined. An example of this file provided in EAGLE Manual Version 6 (Cadsoft, 2011a) is as following:

```
G04 Film Name:    paste_top*
G04 Origin Date:  Thu Sep 20 15:54:22 2007*
G04 Layer:  PIN/PASTEMASK_TOP*
%FSLAX55Y55*MOIN*%
%IR0*IPPOS*OFA0.00000B0.00000*MIA0B0*SFA1.00000B1.00000*%
%ADD28R,.11X.043*%
%ADD39O,.07X.022*%
%AMMACRO19*
21,1,.0512,.0512,0.0,0.0,45.*%
%ADD19MACRO19*%
%LPD*%
G75*
G54D10*
X176250Y117500D03*
Y130000D03*
Y163750D03*
G54D39*
X496250Y142500D03*
Y137500D03*
Y132500D03*
Y127500D03*
M02*
```

RS-247-D is used to drive vector photo plotters, which were also 2D NC machines. It is formed by simple ASCII format consisting of commands and the X-Y coordinates. This file on its own is not an image description as it does not contain all information, i.e. the coordinate unit and the definitions of the apertures are not defined in this file, and instead they are set manually by the plotter operator. An example of this file provided in EAGLE Manual Version 6 (Cadsoft, 2011a) is as following:

```
D11*
X1785250Y2173980D02*
X1796650Y2177730D01*
X1785250Y2181480D01*
X1796650Y2184580D02*
D12*
X3421095Y1407208D02*
X3422388Y1406150D01*
M02*
```

There are a total of seven output files generated through the EAGLE CAM Processor to a board layout, which are:

*.drl = drill list, import this first in gc-prevue (file type will be unrecognized, select 'drill rack')
*.drd = drill data, location of the drill holes
*.cmp = component side metal
*.sol = solder side metal
*.stc = mask stop for component side
*.sts = mask stop for solder side
*.plc = silkscreen for component side.

2.2.2 ULP File

A User Language Program (ULP) is used by EAGLE users to access to virtually all data either in EAGLE or external files as well as to create any file type and generate any data format used by other software or hardware. It is written in a plain text file with C-like syntax. It uses the extension .ulp, which can be created with any text editor, such as notepad, provided that is does not insert any additional control characters into the file.

A ULP generally consists of two major items, which are definitions and statements. Definitions are used to define constants, variables and functions which will be used by statements. An example of a simple ULP provided in EAGLE Manual Version 6 (Cadsoft, 2011a) is as following:

```
#usage "Add the characters in the word 'Hello'\n"
       "Usage: RUN sample.ulp"
// Definitions:
string hello = "Hello";
int count(string s)
{
  int c = 0;
  for (int i = 0; s[i]; ++i)
```

```
    c += s[i];
  return c;
}
// Statements:
output("sample") {
  printf("Count is: %d\n", count(hello));
  }
```

The presence of the directive “#usage” means that its value will be used in the Control Panel to display a description of the program.

To execute a ULP in EAGLE, the command “Run” from the editor window’s command line is used. It can return information on whether it has run successfully or not. The function “exit ()” is used to set the return value where “0” means the ULP is ended successfully whereas other return value indicates an abnormal termination of program.

2.2.3 Script File

Script file contains any EAGLE command which is written in text file, with the extension .scr. It enables the users to implement their own functions and thus brings to a flexible input interface defined by the EAGLE command syntax. Some of the examples of long sequences of commands are the specifications of specific colours and fill-patterns of layers. Since every EAGLE operation can be carried out with the help of text commands, the user can import data or configure EAGLE with the aid of script files as well.

One of the script file used in the EAGLE is eagle.scr, which is executed each time a new drawing is loaded into an editor window or when the drawing type is changed in a binary. It is used to customize the way EAGLE works. (Cadsoft, 2011a)

2.3 Windows Scripting

Windows is famous as graphical user interface and many computer users do not know that there are also very useful scripting functions in Windows. In some cases, it is better than just pointing and clicking in Windows, and instead the keyboard and the script can be substantial adjuncts to the mouse and icons. Two basic features involved in Windows scripting is the entry “Run” and the command prompt window. (Laurie, 2009c)

Several scripting languages commonly used include batch file, Jscript, Powershell and VBScript which will be discussed in the following sub-sections.

2.3.1 Batch File

Batch files are simple text files containing some lines with commands that get executed in sequence, one after the other. These files use the extension .bat or .cmd, which are recognized and executed through an interface provided by a system file called the command interpreter. (Laurie, 2009a)

A batch file is written in any text editor such as Notepad, where the commands used are often quite simple that a user is not necessary to learn any programming language. These files are normally used on saving time and effort for some routine stuff like system housekeeping and simple file management. However, branching and looping will be needed if users wish to explore the intricacies.

Batch files can be run by simply clicking on it or in a command prompt or the Start-Run line, which is as shown in Figure 2.4. In the second case, the full path name must be used unless the file’s path is in the path environment.

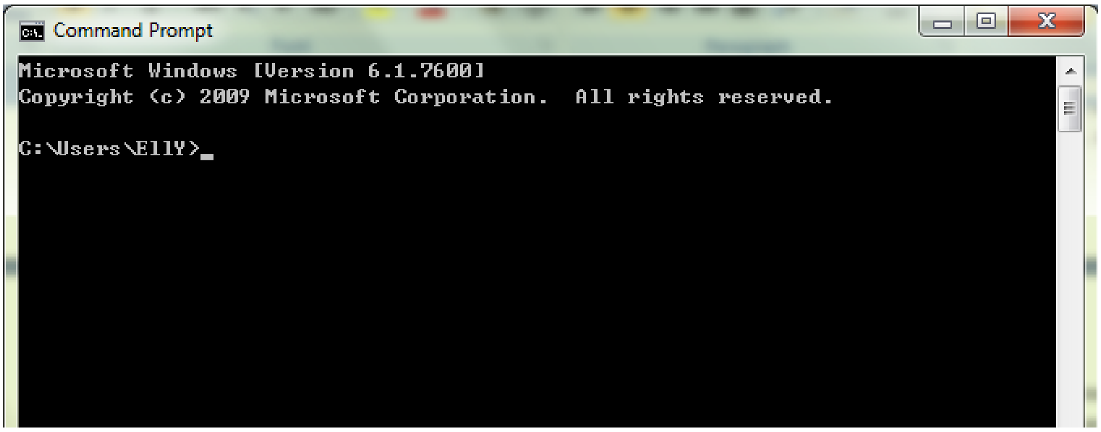


Figure 2.4: Screenshot of Command Prompt

2.3.2 JScript

Jscript is the Microsoft implementation of the JavaScript scripting language. It can be “plugged in” to any application that supports Windows Script, such as Internet Explorer, Active Server Pages and Windows Script Host. It also means that any application supporting Window Script can use multiple languages including Jscript, VBScript and others. ("What is WSH," 2009)

Jscript is an interpreted object-based scripting language, but with fewer capabilities than full-fledged object-oriented languages like C++ and Java. It is not a simplification of any other languages, where it is only distantly and indirectly related to Java. One of its limitations is that user is not able to write standalone applications in it; for example, it has little capability for reading and writing files. Furthermore, it can only run in the presence of an interpreter, either in a web server or a web browser.

To use Jscript, users do not have to declare the data types of variables explicitly. The script can perform conversion automatically when they are needed. For example, when a number is added to an item consisting of text, this number will be converted to text as well. ("What is JScript," 1997)

2.3.3 PowerShell

PowerShell is a new command line interface developed by Microsoft, which was first made available as a stand-alone application for Windows XP and then Vista and Windows 7. The older command interpreter cmd.exe is still present however. PowerShell is quite different from the previous command line interface but is considerably more powerful because of its making use of more sophisticated techniques and objects and the requirement of .NET Framework 2.0. It has new functions for system and network administration and is aimed at IT professionals. (Laurie, 2009b)

Different from the normal Windows command line which is executed by command interpreter, PowerShell has a new approach that makes use of what Microsoft calls “cmdlets”. The definition of “cmdlets” by Microsoft:

A cmdlet (pronounced "command-let") is a single-feature command that manipulates objects in Windows PowerShell. You can recognize cmdlets by their name format -- a verb and noun separated by a dash (-), such as Get-Help, Get-Process, and Start-Service

PowerShell is still able to recognize the commands from the older command shell, although the command is an alias for a PowerShell cmdlet in most cases.

PowerShell is the basis for a scripting language that makes administrative tasks easier and seems likely to supplant VBScript in the future. This script uses the extension .ps1. Many security features are built into the scripting engine and the default setting is to prevent scripts from running where a feature called “Execution Policy” controls the permission to run.

2.3.4 VBScript

Visual Basic Scripting (VBScript) is an active scripting language developed by Microsoft which is the light version of its programming language Visual Basic, with a fast interpreter for the use in a wide variety of Micorsoft environments. It uses the Component Object Model to access elements of the environment within which it is running; for example, the FileSystemObject (FSO) is used to create, read, update and delete files. It is installed by default in every desktop release of Microsoft Windows since Windows 98.

A VBScript must be executed within a host environment, such as Windows Script Host (WSH), Internet Explorer (IE) and Internet Information Services (IIS). In addition, this hosting environment is embeddable in other programs through technologies such as Microsoft Script Control (msscript.ocx).

When a VBScript is inserted into an HTML document, IE browser will read the HTML and interpret the VBScript. The script can be executed immediately, or at a later event. Other web browsers such as Firefox and Opera do not have built-in support for VBScript, which means that developers will choose Jscript over VBScript when client-side scripting and cross-browser compatibility are required.

Although VBScript is a general-purpose scripting language, it is more preferable in some particular areas, such as among system administrators in Microsoft environment. Besides, it is the scripting language for Quick Test Professional, a test automation tool. VBScript is also adopted as the internal scripting language for some embedded applications, such as industrial operator interfaces and human machine interfaces.

The relatively widespread use of VBScript mainly because the royalties need not be paid to Microsoft by implementers as long as the VBScript trademark is acknowledged. Undeniably, there are several useful features in the full Visual Basic, which include strong typing, extended error trapping and ability to pass a variable number of parameters to a subroutine have been removed. (Clinick, 2000)

2.4 Configurations of Mail Server

Several specific terms used in the settings of most mail servers need to be digested before the configurations are started. Table 2.1 shows the definitions of some of the most commonly-used terms. ("Configuring other mail clients," 2011)

Table 2.1: Definition of Terms in Email Configurations

Post Office Protocol (POP)	It is a one way download of the messages which allows user to access the mail with a mail program like Outlook Express. It only offers one-way communication, which means that the action user takes in mail client will not be synced to the mail server.
Domain	It is a name for web address.
Transport Layer Security (TLS)	It is a way of changing data such as the username and password into code as it travels across the Internet, so that the server will be secure and private. It begins with an unsecured connection to mail servers and then upgrades to a secure connection once the information is sent in a mail delivery.
Simple Mail Transfer Protocol (SMTP)	It is a set of standard Internet procedures where two email providers transfer email messages to one another’s mail servers.

In order to properly use the local email client software such as Outlook Express, Microsoft Outlook or the function provided in LabVIEW, the email software needs to be configured with the incoming and outgoing mail servers of the email provider. Table 2.2 shows some of the mail server settings.

Table 2.2: Mail Server Settings

Email Provider	Settings
Hotmail	
Hotmail Incoming Mail Server (POP3)	pop3.live.com (logon using Secure Password Authentification - SPA, mail server port: 995)

Hotmail Outgoing Mail Server (SMTP)	smtp.live.com (TLS enabled, port 587)
Yahoo! Mail	
Yahoo Incoming Mail Server (POP3)	pop.mail.yahoo.com (SSL enabled, port 465)
Yahoo Outgoing Mail Server (SMTP)	smtp.mail.yahoo.com (SSL enabled, port 995)
Google Gmail	
Google Gmail Incoming Mail Server (POP3)	pop.gmail.com (SSL enabled, port 995)
Outgoing Mail Server	smtp.gmail.com (TLS enabled, port 587)
MSN Mail Settings	
MSN Incoming Mail Server (POP3)	pop3.email.msn.com (port 110, using Secure Password Authentication - SPA)
MSN Outgoing Mail Server	smtp.email.msn.com (select "My outgoing server requires authentication")

2.5 Probe Holder

A probe holder facilitates the operation of holding and sliding the probe over a DUT, which is normally configured to allow a flexible adjustment of the probe so that the probe can be fixed at a location having intimate contact with the DUT. A probe holder can be malfunctioned, which is not only limited to holding a probe, but also other equipment to offer a hands-free work to the engineers.

Some of the existing probe holders available in market nowadays are investigated in the following sub-sections especially on their features and applications.

2.5.1 Agilent N2786A Two-Leg Probe Positioner

Agilent has come out with various types of probe positioner, where one of them is N2786A. It is a low-cost two-legged positioned, which has no controls to position it in place and requires the user to position it manually instead. It has three different sized apertures that allow three different sizes of probes to be placed, and these cover most of the passive probes in the market. (Agilent, 2009) The price of this probe positioner is RM237, quoted from the official website of Agilent. Figure 2.5 shows the outlook of this device.

Figure 2.5: N2786A two-leg probe positioner

2.5.2 Agilent N2784A One-Arm Probe Positioned

Another probe positioner produced by Agilent is N2784A which provide a quick and stable X-Y positioning to PCB and devices that require hands-free probing. It does not require multiple adjustments to lock the probe in position, but needs only a "lift and drop" motion. It applies a weight stabilization technique to keeps constant pressure at the probing point so the probe tip is able to stay stably in a fixed position even when the target board is bumped. As for N2786A, these N2784A probe positioners are compatible with most of the probes available in market. (Agilent, 2009) Its price as quoted from Agilent's official website is RM4597. Figure 2.6 shows the outlook of this probe positioner.

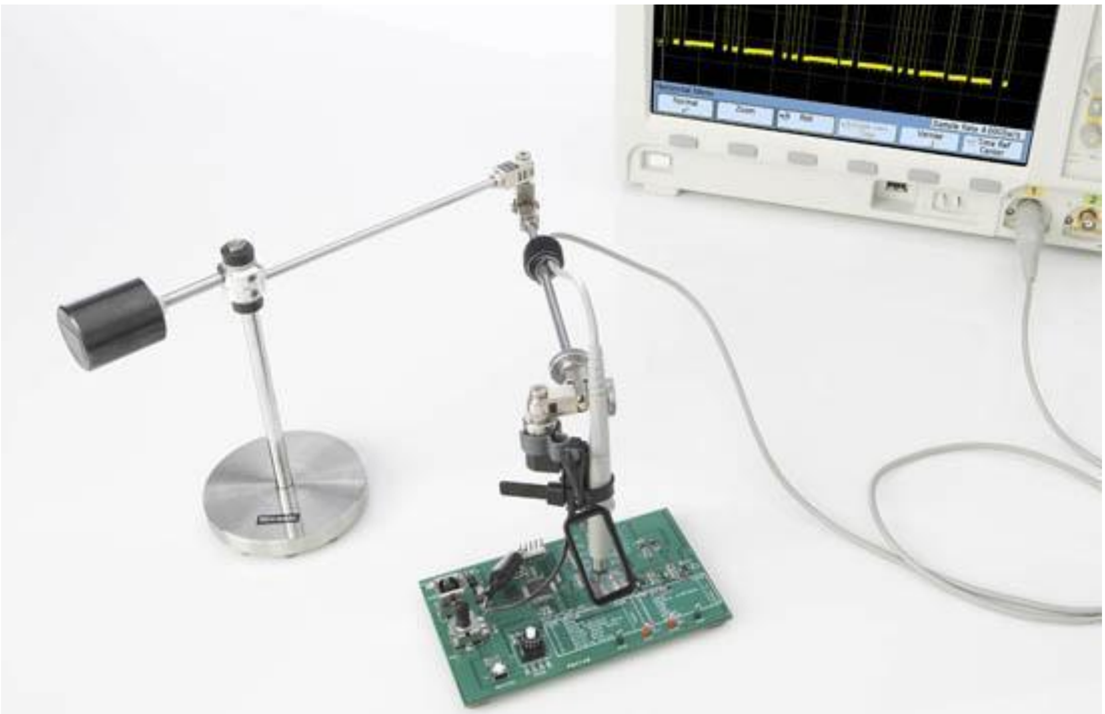


Figure 2.6: N2784A one-arm probe positioner

2.5.3 MAGNA-VUE MV-36 Probe Positioner

The MAGNA-VUE MV-36 probe positioner provide a hand-free probing without taking up a lot of bench space or unneeded complexity. Its two swivel joints are fully adjustable to allow motion in all six degrees of freedom, and yet able to be locked at rigid position. A knob on a spring loaded lever at the base allows fine control of the probing pressure. The users can attach various brands and types of probes using common wire ties with this v-groove probe holder. The price quoted for this device is at $895 from its website.(Magna-Vue, 2010) Figure 2.7 shows its outlook.

Figure 2.7: Magna-VUE MV-36 Probe Positioner

CHAPTER 3

METHODOLOGY

3.1 Overview

The block diagram in Figure 3.1 shows the general design of the whole 2-D location pointing system for individual component on device under test. The focused parts include software – getting component coordinate from PCB drawing software and using SMTP to link up Gmail with the program, and also the multipurpose flexible holder – hardware design and speech recognition. Figure 3.2 shows the flowchart of how the overall process runs, where this project is focusing on the two circled parts in the figure.

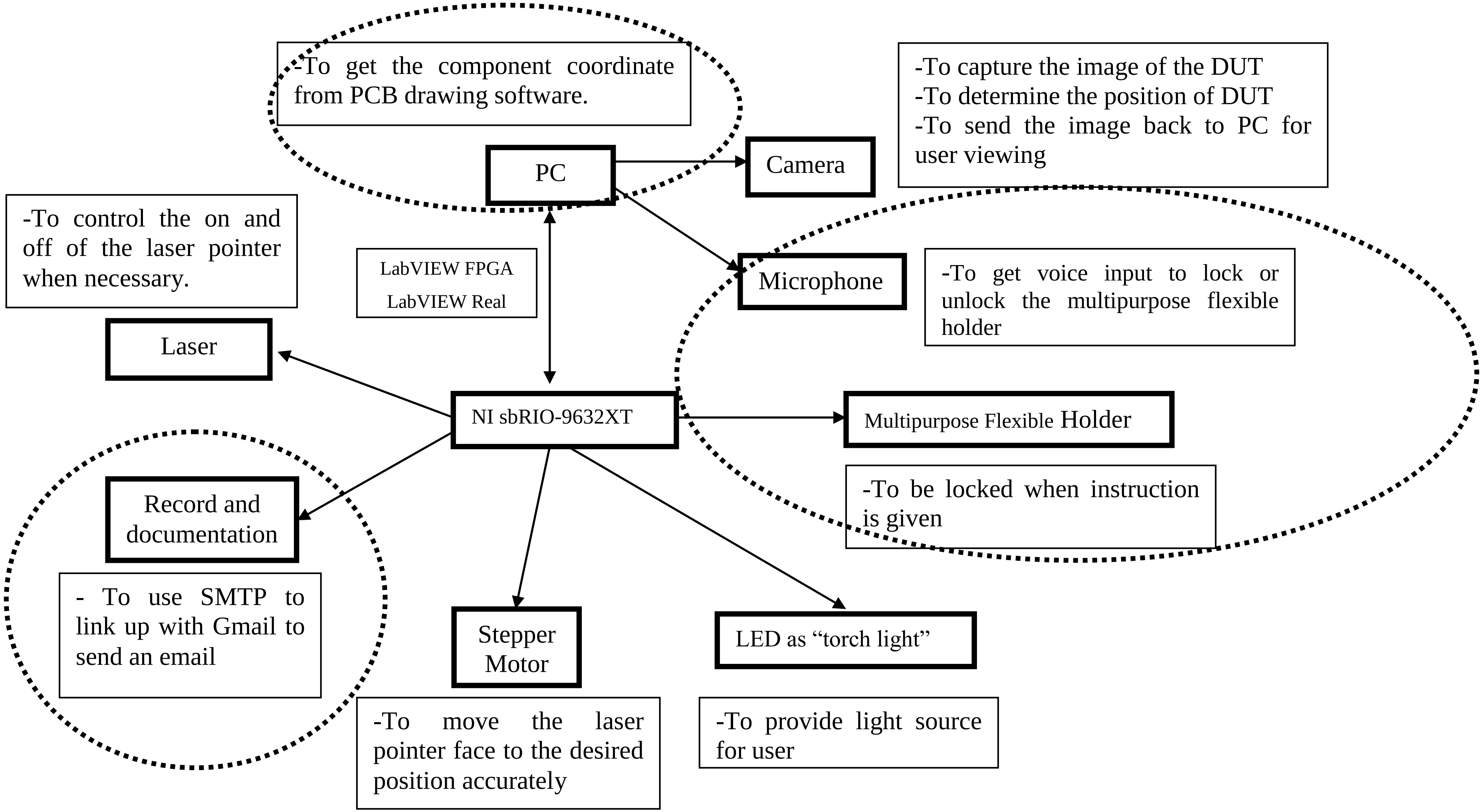


Figure 3.1: Overall Block Diagram of the Location Pointer System

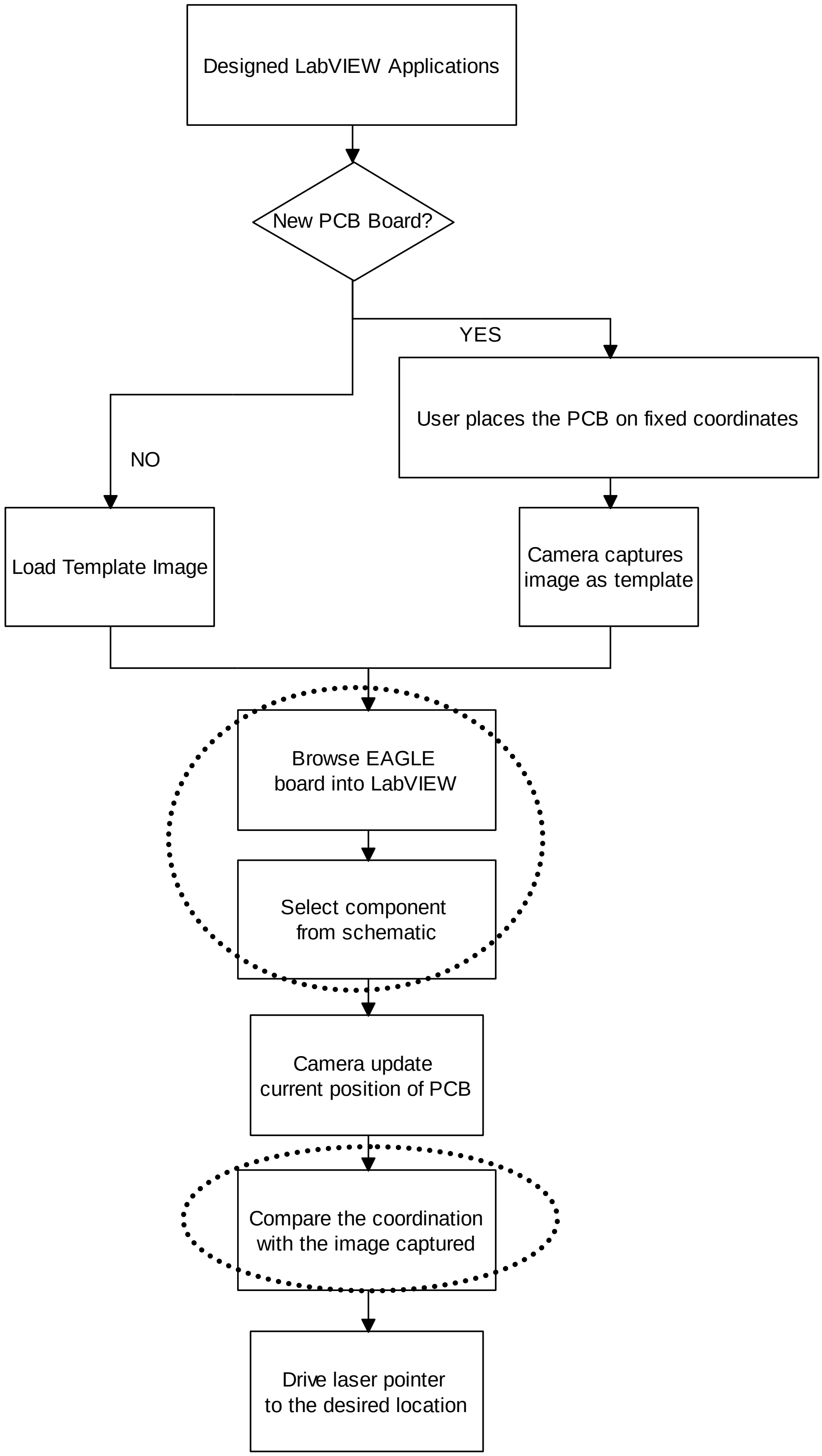


Figure 3.2 Overall Flowchart of the Location Pointer System

3.2 Communication between PCB Drawing Software and LabVIEW

When the user selects a component in the schematic of the DUT, the laser pointer has to be able to point to this exact location. Hence, it is important to get the coordinates of the component on the board from the PCB drawing software. The PCB drawing software used in this project is EAGLE, because it provides a lot of scripting freedom for the software development that allows users to control it by simple scripting.

The implementation of the communication is basically divided into two sections, which are hardware implementation and software implementation, as shown in Figure 3.3. The whole process will be completed in a personal computer, because all the software used- LabVIEW, window scripting and EAGLE, are installed in the personal computer.

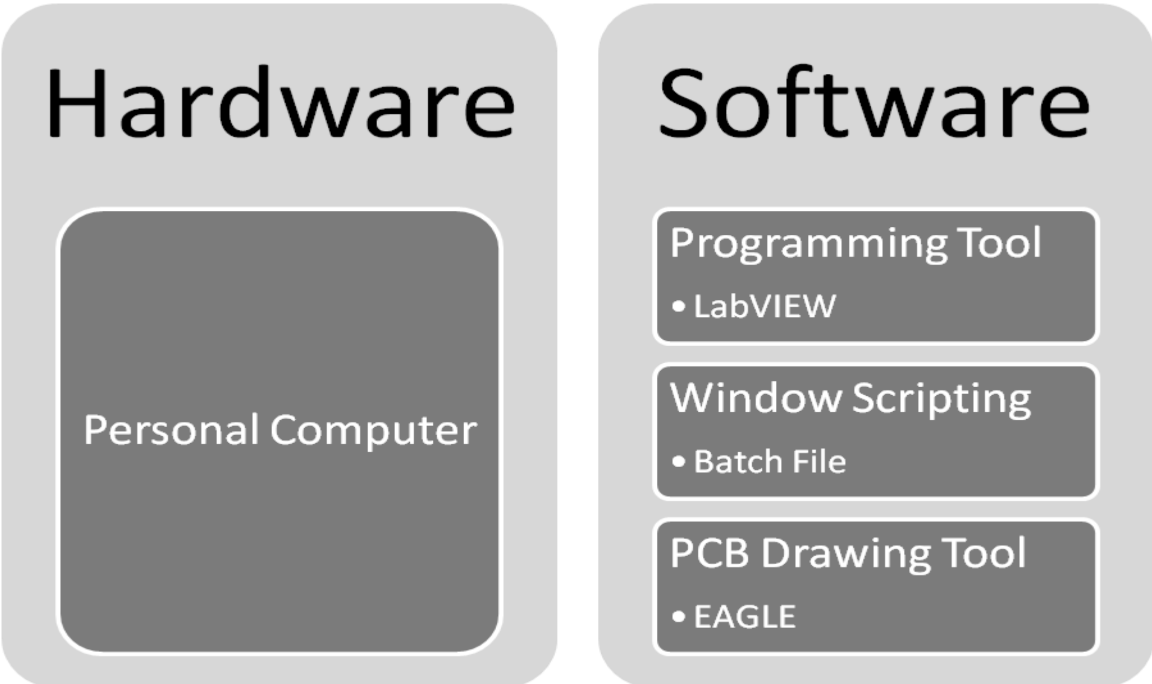


Figure 3.3: Implementation of Hardware and Software for the Communication between the PCB Drawing Software and LabVIEW

LabVIEW is not able to access the external applications in the computer, i.e. EAGLE, but it can make use of its function- System Exec VI to execute or launch

Window applications or Linux command-line applications. Hence, a window scripting is needed to be the intermediate between LabVIEW and EAGLE, i.e. LabVIEW accesses to the script, and the script will access to EAGLE, as shown in Figure 3.4.

Figure 3.4: Link between LabVIEW and EAGLE – Window Scripting

The window scripting used in this project is a batch file, which can be easily run by LabVIEW by using System Exec VI. The command in the script will access into EAGLE to bring out the actions needed, which are as following:

- To export the image of the schematic
- To generate the coordinate files of both the schematic and board, by running a ULP file

After obtaining the desired outputs from EAGLE, some conversions and linking will be done in LabVIEW in order the get the component's coordinate on board when the user is clicking the component in schematic. The overall flow of this process is shown in the flowchart in Figure 3.5.

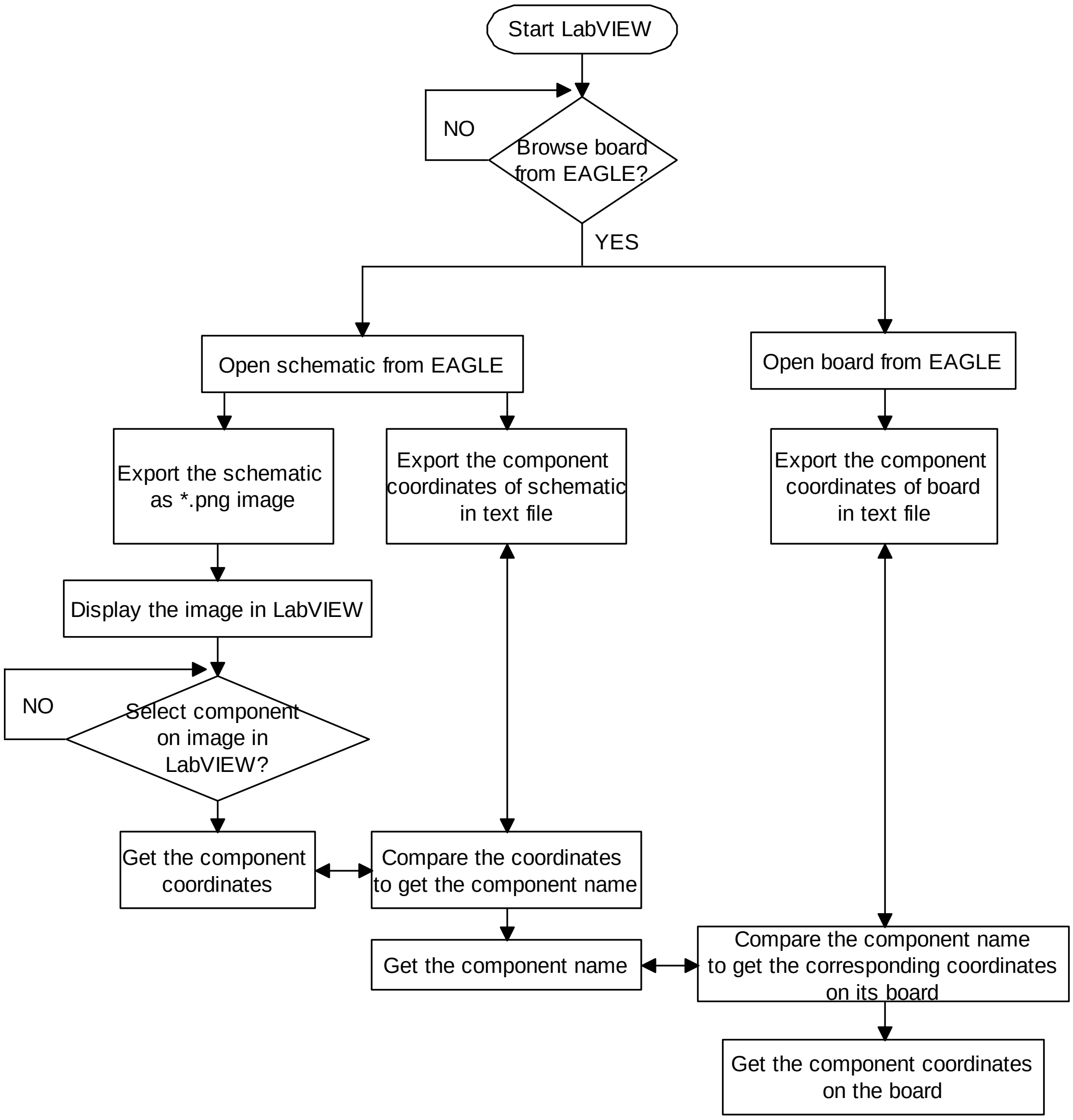


Figure 3.5: Flowchart of Process to Get Component Coordinate from EAGLE

3.3 Process Documentation

The documentation of the overall process is vital to have a reference for the future use. The history of the testing process of a DUT needs to be recorded to ease the tracking back when errors happen. Thus, the tested components, the coordinates, the date of testing and etc are to be recorded in this project. Two types of record will be kept, which are in a Comma-Separated Values (CSV) file saved in local drive and also sent to assigned email.

3.3.1 Record Keeping in CSV File

The data to be kept in the CSV file includes date, time, names of components, and xy coordinates of components. User is given an option to add comment after each testing process. The file will be saved by using the same name as the selected PCB. The format of the record keeping is shown in Table 3.1.

Table 3.1: Format of Record Keeping in CSV File

Date	Component's name	X coordinate	Y coordinate	Comment (optional)
Time1	Name1	X1	Y1	Comment1
Time2	Name2	X2	Y2	-

3.3.2 Record Keeping in Email

The data may need to be accessed in different location or by different person; hence sending the CSV file to email will be necessary sometimes. There are a lot of different mail servers nowadays, where different of them require different settings configuration. For this project, one of the servers will be chosen to ease the development of the software. It will only limit the sender's email account, but not the recipient's email account.

The Google Gmail service is chosen in this project. The outgoing mail server of Gmail uses the Simple Mail Transfer Protocol (SMTP) mail server address provided by smtp.gmail.com, with Transfer Layer Security (TLS) enabled and by using port 587. ("Incoming and Outgoing Mail Server Settings for Hotmail, Yahoo! Mail, GMail, MSN, AOL and more," 2011)

3.4 Multipurpose Flexible Holder

In this project, the user will be able to have hand-free holding of any pen-shaped devices when they are doing their works. It is not only targeting to hold the probe, but also other devices such as holding leads during the soldering process. In addition, the designed holder does not require multiple adjustments to be locked at position, but with only speaking out the word “lock”, and the “unlock” to free the holder from that position. The implementation of hardware and software is shown in Figure 3.6.

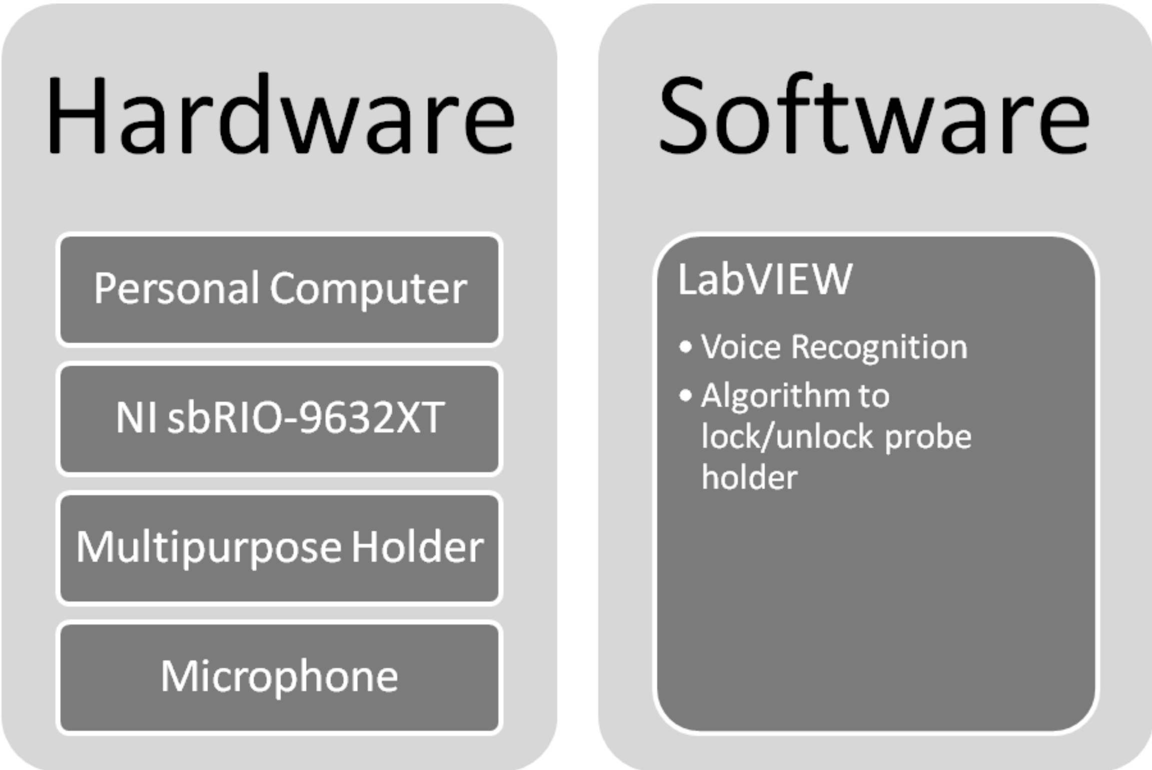


Figure 3.6: Implementation of Hardware and Software for Design of Multipurpose Flexible Holder

A personal computer is used to process the voice recognition of the words “lock” and “unlock” obtained from the microphone attached whereas the NI sbRIO-9632XT board is used to control the input and output of the holder. The software involved is the voice recognition algorithm in LabVIEW and also the output of signal to lock/unlock the holder. The development of the mechanical design of the holder as well as the voice recognition will be discussed in the following sub-sections.

3.4.1 Mechanical Design

In order to implement a full flexibility to the multipurpose holder in this project, some degrees of freedom (DOF) have to be adopted into the design. The DOF is a joint on the arm, a place where it can bend, rotate or translate. A real life example of joints with multiple DOFs is human shoulder, as shown in Figure 3.7, which is often a reference for building robotic arm. Figure 3.8 shows a 5 DOF robotic arm, by referring to its number of actuators.

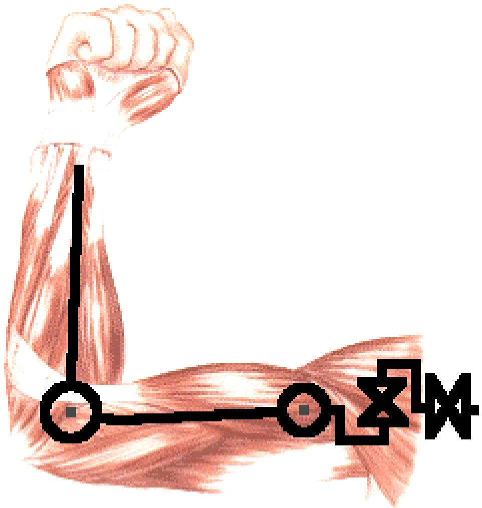


Figure 3.7: DOF in Human Shoulder

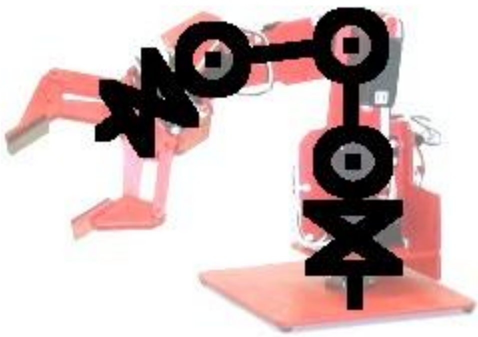


Figure 3.8: DOF of Robotic Arm

The design of the multipurpose flexible holder drawn in SolidWorks is as shown in Figure 3.9 (its outer look) and Figure 3.10 (its inner look). There will be three DOF taken on the holder, the centre part is connected by a sphere holder which is able to make a wide angle rotation.

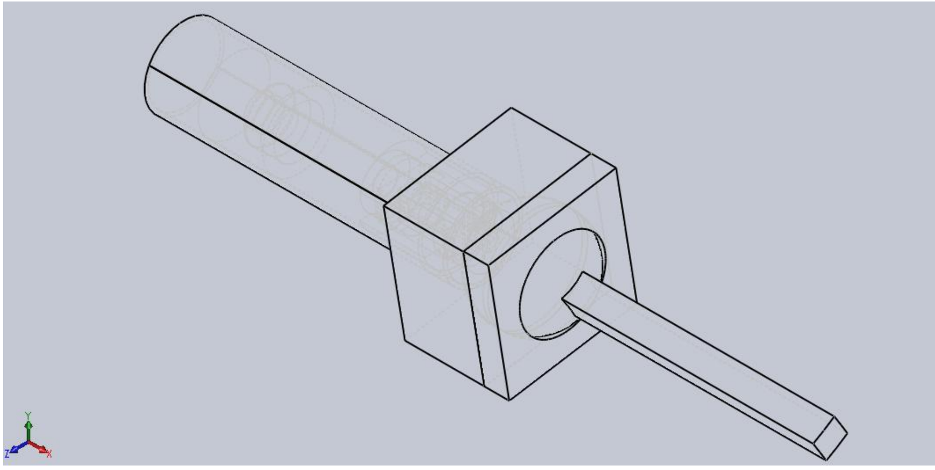


Figure 3.9: Outer View of Holder Design

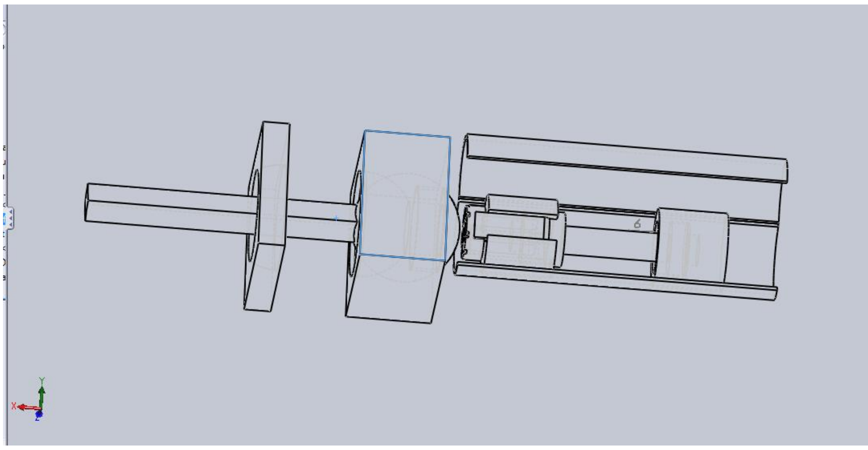


Figure 3.10: Inner View of Holder Design

Two small cylindrical magnets will be fitted inside the holder to ‘lock’ or ‘unlock’ the movement of the holder when instruction is given. To lock the holder’s movement, the electromagnet is activated in certain direction in the core which will attract the magnet to it, which subsequently push a stopper to create a friction to ‘lock’ the sphere in position and hence the holder is locked. To unlock it, the electromagnet is activated in the opposite direction and the stopper will be repelled back to its original position. The production of the electromagnet is due to a current carrying solenoid, where a solenoid is a long coil of wire consisting many loops of wire. A solenoid acts like a magnet with one end is considered as north pole and the other the south pole. By placing a piece of iron inside a solenoid, the magnetic field is increased greatly because the iron becomes magnet, as shown in Figure 3.11. (Giancoli, 2005) Figure 3.12 shows the magnets to be fitted with the iron core with coil of copper wire in this project.

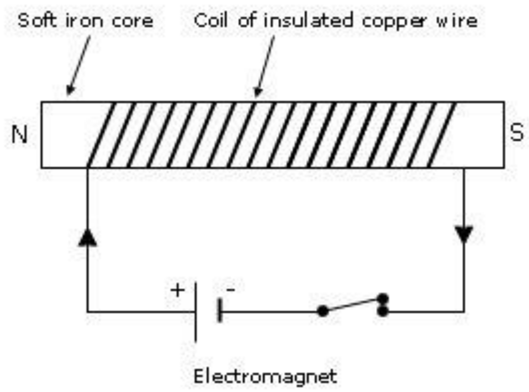


Figure 3.11: Mechanism to Produce Electromagnetism

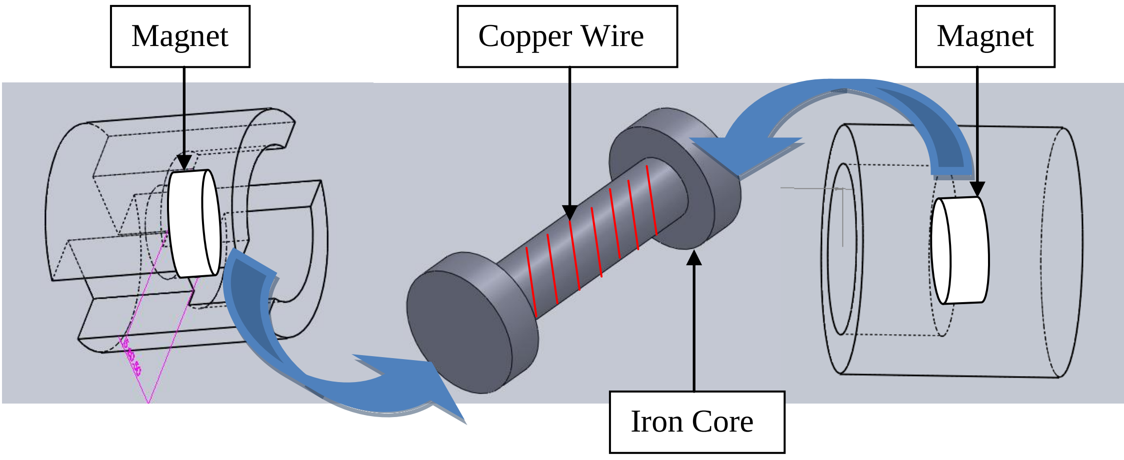


Figure 3.12: Magnet Fitted Inside the Holder

The magnitude of the magnetic field produced in this mechanism is directly proportional to the number of turns in the solenoid as well as to the strength of current provided to the solenoid. In addition, it depends on the nature of core material used in making the solenoid, whereby using soft iron rod as core in a solenoid will produce the strongest magnetism. (Jezek, 2006)

3.4.2 Design for Speech Recognition

A microphone is needed in order to obtain the speech “lock” or “unlock” from the user to operate this multipurpose holder, hence a Logitech Webcam C170 with built-in microphone is selected for this project since the camera is needed in another part of this project. With its plug-and-play setup, this microphone can be easily connected to the personal computer and start functioning. The clear voice can be collected without the background noise such as the humming sound as the ambient noise has been isolated and more deftly been cancelled out with a signal below the threshold of human hearing. The outlook of C170 is as shown in Figure 3.13.

Figure 3.13: Logitech C170 with Built-in Microphone

The implementation of voice recognition in LabVIEW can be completed by using some predefined VI such as speech setup.vi to set the initializations required for the voice recognition. With the speech setup.vi integrated in the overall program, the words “lock” and “unlock” can be identified and the subsequent actions can be taken to the holder. The flowchart of the process is as shown in Figure 3.14.

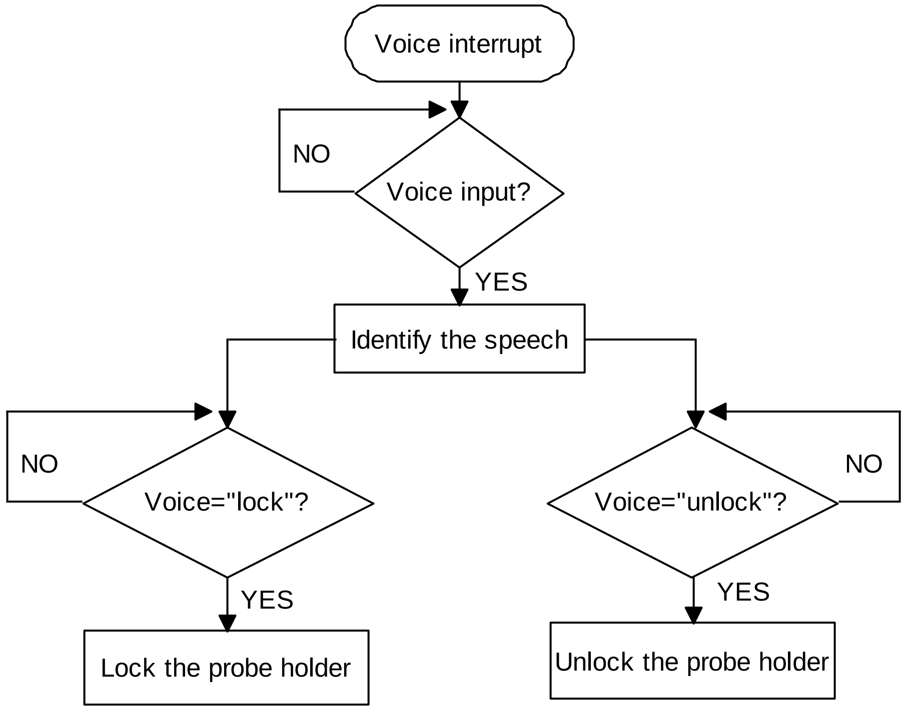


Figure 3.14: Flowchart of Process to Lock or Unlock the Holder

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Software Design Architecture

The main program VI linked all the sub-VIs of different functions in order to achieve the overall project objectives. However, this project focused on certain tasks, which included EAGLE_Path.vi, Layout_Select.vi, Get_Coordinate.vi, XY_Coord_Calibration.vi and Email_Module.vi. The software flow chart of the main.vi is shown in Figure 4.1, whereas other sub-VIs are discussed one by one in the following sub-sections.

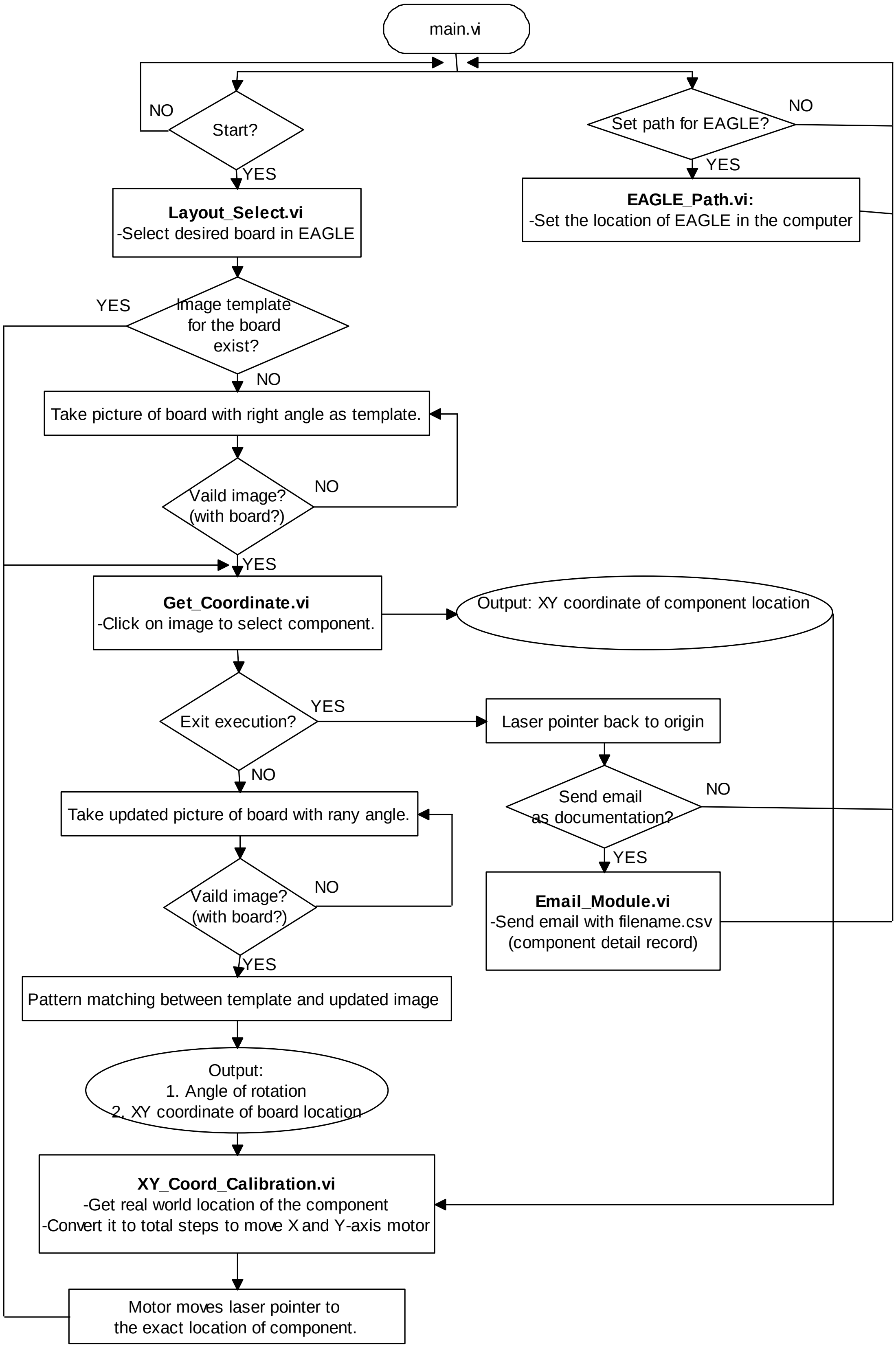


Figure 4.1: Flowchart of main.vi

Figure 4.1 shows that the overall program starts only after the user click the start button. User was prompted to select the board in EAGLE whose PCB was going to be tested. The program then checked whether the image template of the PCB had already existed in the collection, and it took a picture of the PCB if it did not exist. The PCB needed to be placed at right angle for template taking, but it was tolerable with any angle of XY rotation after having the template. The program loaded the image of the schematic for the user to select component to be pointed. Calibration of the XY coordinate of selected component to the exact coordinate of the component on real board was done and the coordinate was converted into the motor steps that drove the laser pointer to the desired location. When the user clicked on the button to exit the execution, the motor drove the pointer to back to its origin. User was given option whether to send the testing record to email.

4.1.1 Development of EAGLE_Path.vi

EAGLE_Path.vi was needed during the first use of this program in a computer. Every computer needed to have the EAGLE installed in its own location. If the path of the bin folder that contains EAGLE.exe was not set correctly, the predefined EAGLE script in this program was not able to execute. LabVIEW failed to access to EAGLE. The flowchart of this sub-VI is shown in Figure 4.2.

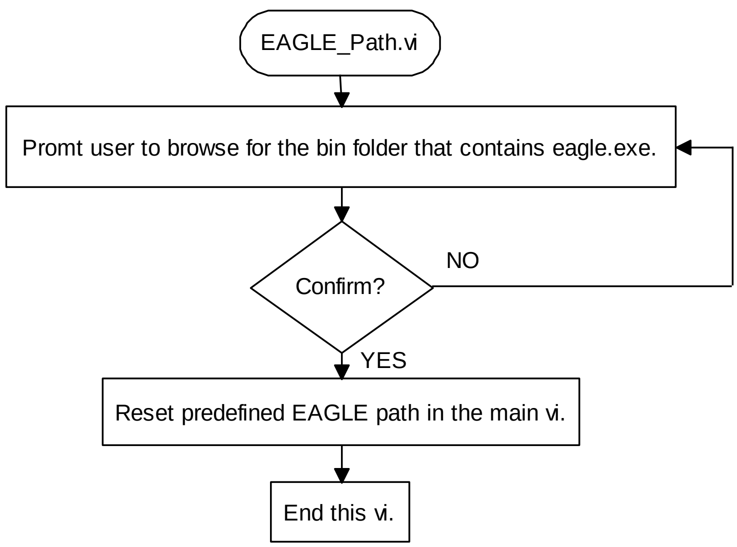


Figure 4.2: Flowchart of EAGLE_Path.vi

4.1.2 Development of Layout_Select.vi

The main function of Layout_Select.vi was to generate the required output files: image of schematic (SchImage.png – Figure 4.3), coordinate file of schematic (sch_coord.mnt – Figure 4.4), and also coordinate file of board (brd_coord.mnt – Figure 4.5). The batch files (generate_sch_coord.bat – Figure 4.6 and generate_board_coord.bat – Figure 4.7) were the window scripting used to run the internal script of EAGLE to execute the ULP files (SchCoordGeneration.ulp – Figure 4.8 and BoardCoordGeneration.ulp Figure 4.9) as well as to export the image of schematic. Those executed ULP files ultimately produced the coordinate files of both the schematic and board. The complete flowchart of this sub-VI is shown in Figure 4.10.

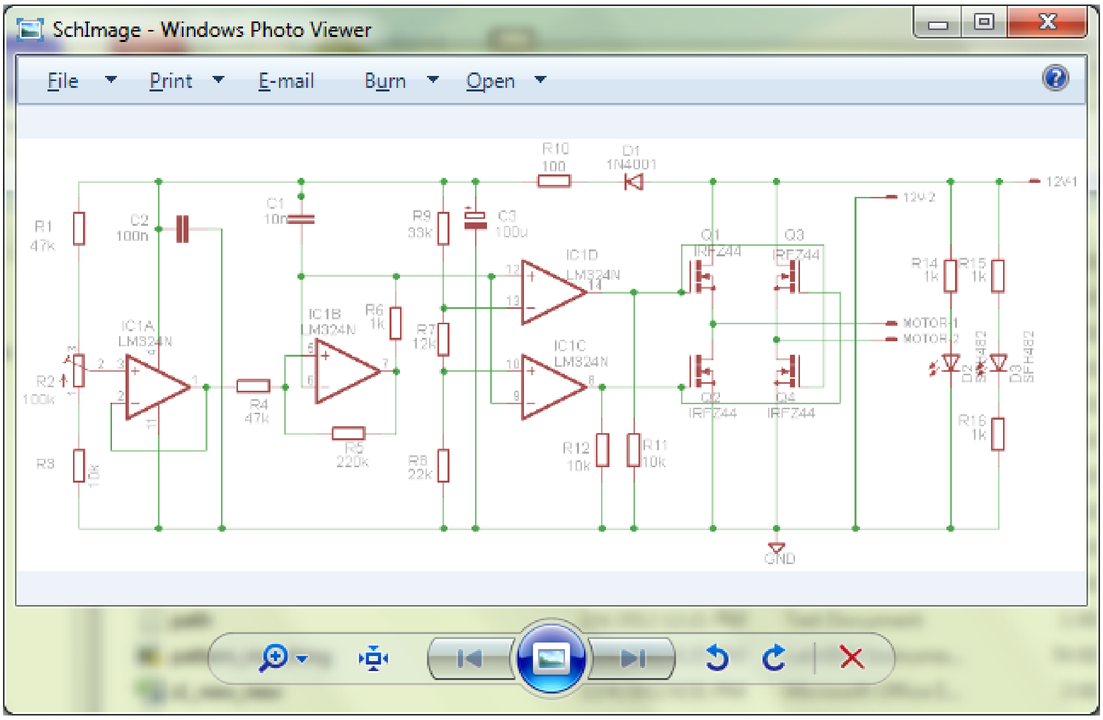


Figure 4.3 Screenshot of SchImage.png

sch_coord - Notepad

File Edit Format View Help

C1	70.3580	27.1780
C2	70.3580	50.0380
C3	70.3580	72.8980
C4	93.2180	17.0180
C5	93.2180	39.8780
C6	93.2180	62.7380
DIS2	29.7180	47.4980
H1	6.8580	100.8380
H2	50.0380	100.8380
JP1	32.2580	17.0180
JP2	9.3980	55.1180
JP3	118.6180	44.9580
R1	70.3580	19.5580
R2	70.3580	42.4180
R3	70.3580	65.2780
R4	93.2180	9.3980
R5	93.2180	32.2580
R6	93.2180	55.1180
S1	62.7380	24.6380
S2	62.7380	47.4980
S3	62.7380	70.3580
S5	88.1380	17.0180
S6	88.1380	39.8780
S7	88.1380	62.7380
SUPPLY1	57.6580	80.5180

Figure 4.4: Screenshot of sch_coord.mnt

board_coord - Notepad

File Edit Format View Help

C1	46.9900	12.6900
C2	41.9100	12.6900
C3	36.8300	12.6900
C4	31.7500	12.6900
C5	26.6700	12.6900
C6	21.5900	12.6900
DIS2	104.1400	53.3300
H1	9.1600	8.8800
H2	110.2200	8.8800
JP1	70.8700	22.8500
JP2	98.7900	22.8500
JP3	40.6400	22.8500
R1	44.4500	11.4200
R2	39.3700	11.4200
R3	34.2300	11.4000
R4	29.3300	11.3000
R5	24.1300	11.4200
R6	19.0500	11.4200
S1	54.6100	35.5500
S2	45.7200	35.5500
S3	36.8300	35.5500
S5	26.6700	36.1850
S6	17.7800	36.1850
S7	8.8900	36.1850

Figure 4.5: Screenshot of board_coord.mnt

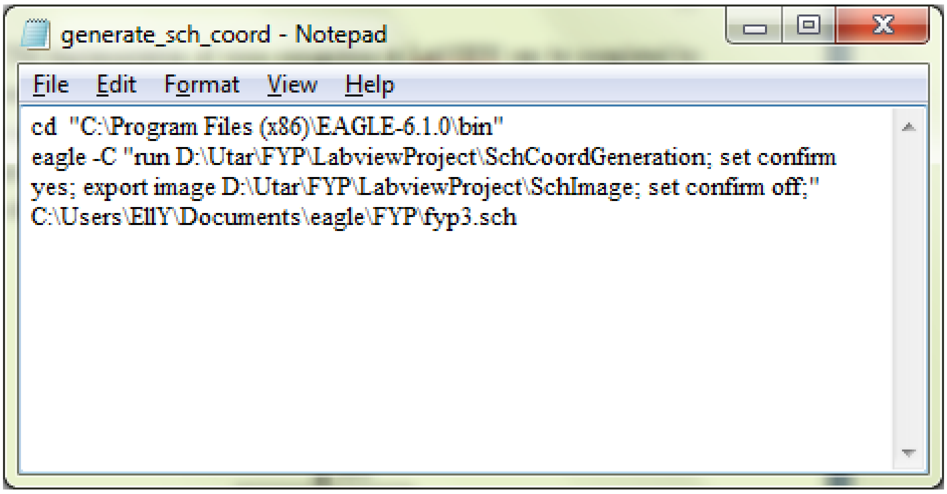


Figure 4.6: Screenshot of generate_sch_coord.bat

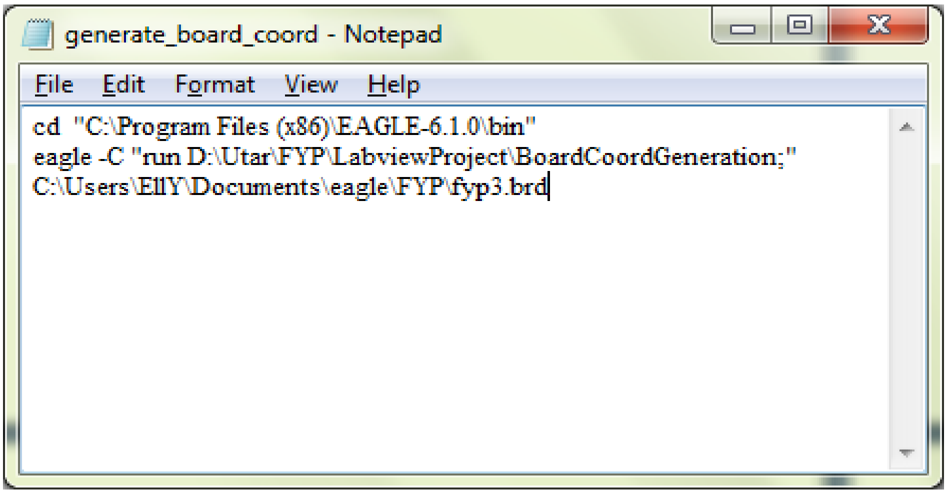
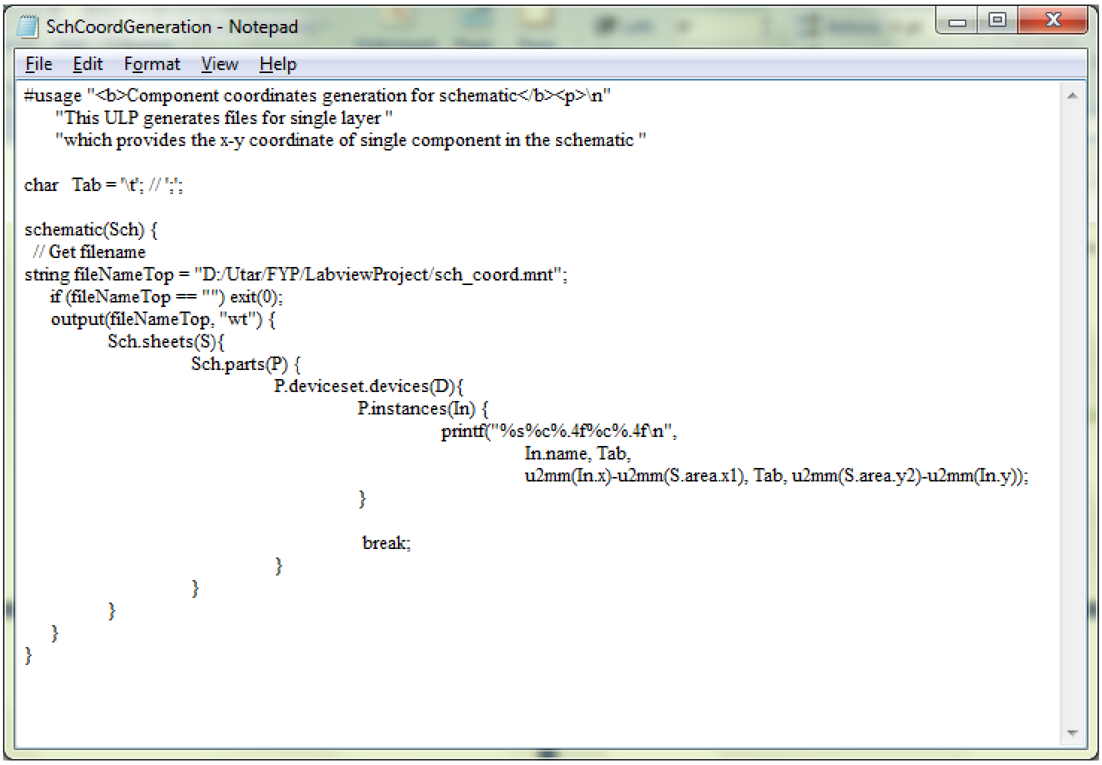


Figure 4.7: Screenshot of generate_board_coord.bat



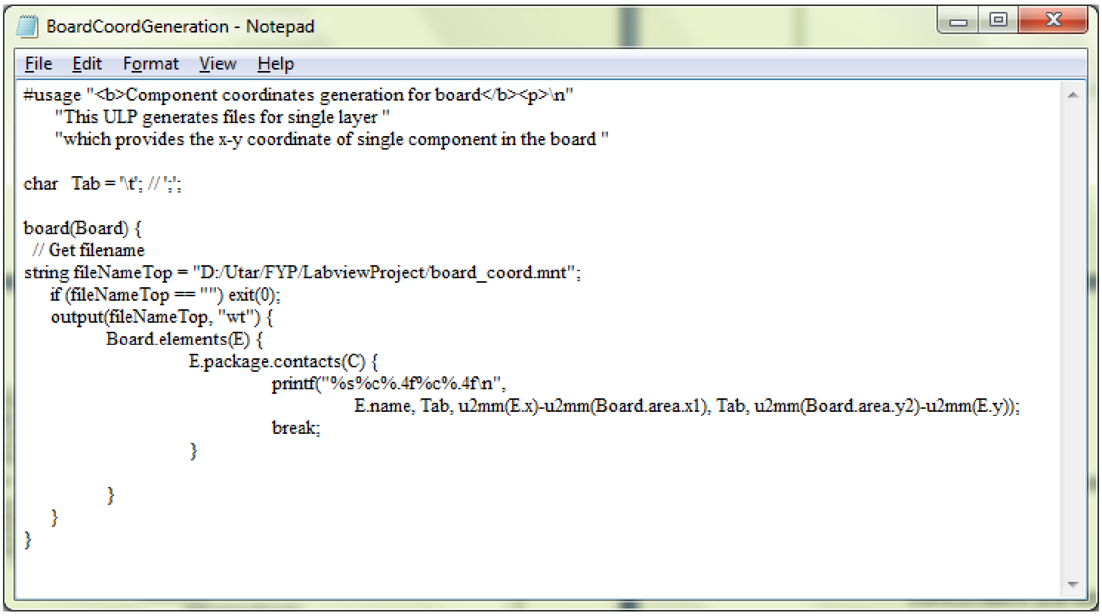

```
#usage "<b>Component coordinates generation for schematic</b><p>\n"
    "This ULP generates files for single layer "
    "which provides the x-y coordinate of single component in the schematic "

char  Tab = '\t'; // ';';

schematic(Sch) {
 // Get filename
string fileNameTop = "D:/Utar/FYP/LabviewProject/sch_coord.mnt";
   if (fileNameTop == "") exit(0);
   output(fileNameTop, "wt") {
          Sch.sheets(S){
                    Sch.parts(P) {
                              P.deviceset.devices(D){
                                        P.instances(In) {
                                                  printf("%s%c%.4f%c%.4f\n",
                                                            In.name, Tab,
                                                            u2mm(In.x)-u2mm(S.area.x1), Tab, u2mm(S.area.y2)-u2mm(In.y));
                                        }

                                        break;
                              }
                    }
          }
   }
}
```

Figure 4.8: Screenshot of SchCoordGeneration.ulp




```
#usage "<b>Component coordinates generation for board</b><p>\n"
    "This ULP generates files for single layer "
    "which provides the x-y coordinate of single component in the board "

char  Tab = '\t'; // ';';

board(Board) {
 // Get filename
string fileNameTop = "D:/Utar/FYP/LabviewProject/board_coord.mnt";
   if (fileNameTop == "") exit(0);
   output(fileNameTop, "wt") {
          Board.elements(E) {
                    E.package.contacts(C) {
                              printf("%s%c%.4f%c%.4f\n",
                                        E.name, Tab, u2mm(E.x)-u2mm(Board.area.x1), Tab, u2mm(Board.area.y2)-u2mm(E.y));
                              break;
                    }

          }
   }
}
```

Figure 4.9: Screenshot of BoardCoordGeneration.ulp

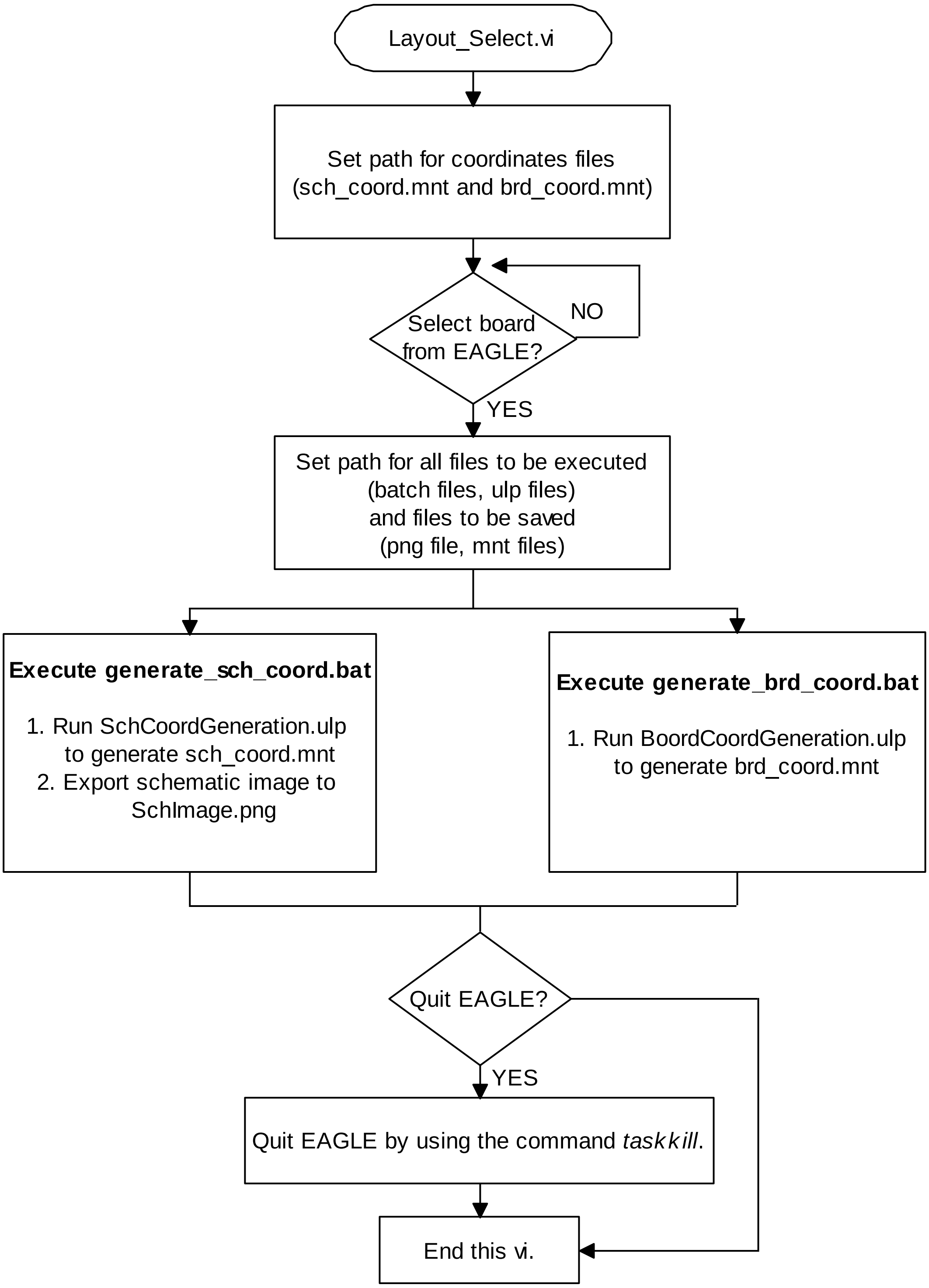


Figure 4.10: Flowchart of Layout_Select.vi

This sub-VI incorporated a lot of external executable files, and hence all the paths of the files needed to be set carefully. In this case all the paths were set relatively to this sub-VI, i.e. in the same folder.

4.1.3 Development of Get_Coordinate.vi

Get_Coordinate.vi began at prompting the user to select a component by clicking a point in SchImage.png. The coordinates of the points were compared with the coordinate of the component in the schematic. The component of closest distance with that point was the component selected by the user; however it was defined as no component selected if the closest distance was more than 5mm. This limit was set to counter the situation when user clicked on area without any component. After identifying the selected component's name, the program matched the name with the component list generated from the board and gets its coordinate. The coordinate was then export to the next VI – XY_Coord_Calibration.vi. Before ending this VI, it saved all the history data of the selected component into a CSV file, and provided user an option to whether add a comment on this testing. The complete flow of this sub-VI is shown in Figure 4.11.

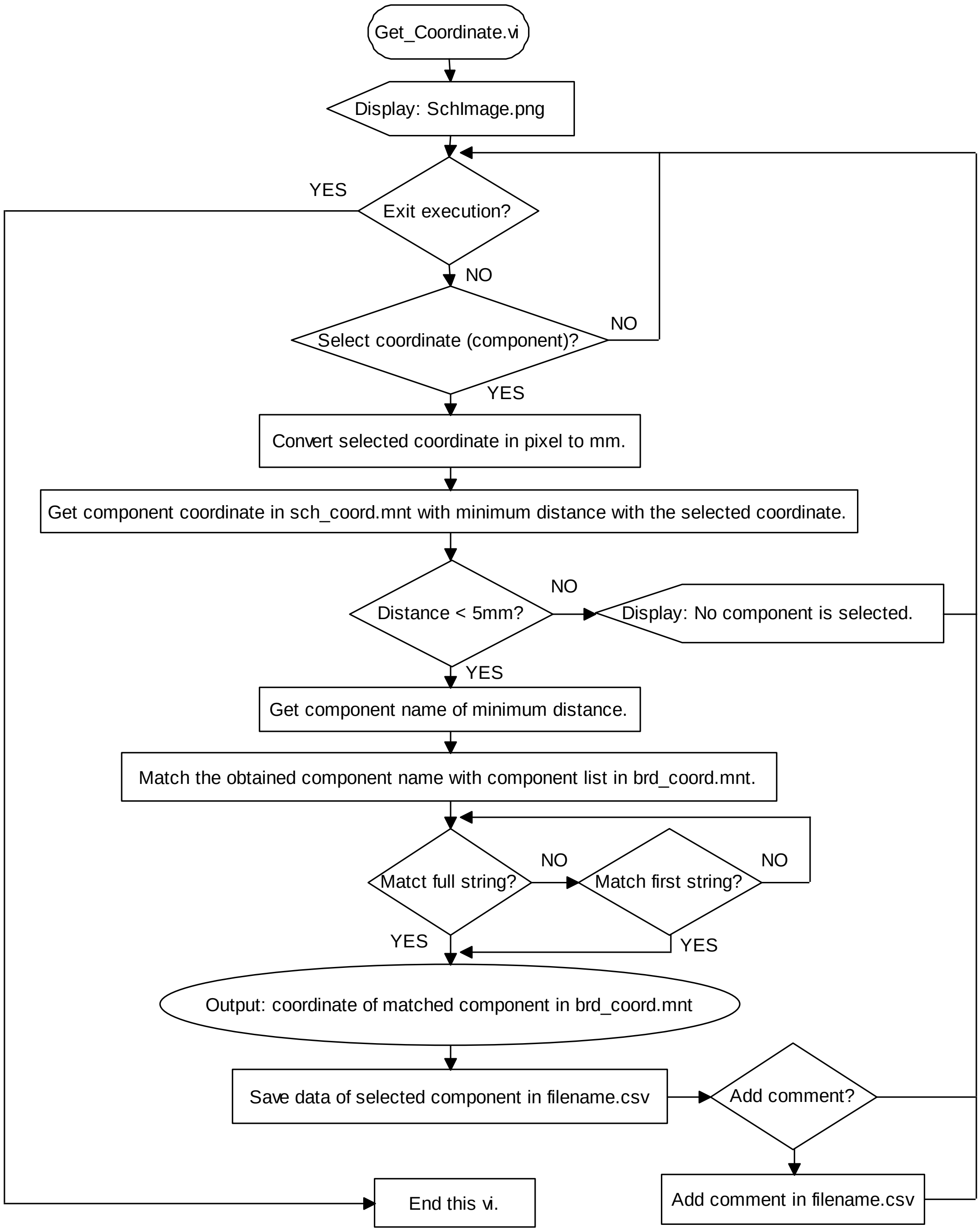


Figure 4.11: Flowchart of Get_Coordinate.vi

There were two concerns in this sub-VI. The first one was about the unit conversion between pixels and millimetres. The coordinate obtained when the user clicked on the image is in pixels, but the calibration of getting exact XY location of the component was done in millimetres. Hence, conversion was needed by using Equation 4.1.

$$
\begin{aligned}
Distance(mm) &= Distance\ (pixel) /(resolution\ (dpi) * conversion\ (inch/mm)) \\
&= Distance\ (pixel) / (150dpi * (1/25.4)) \\
&= Distance\ (pixel) / 5.91 \qquad (4.1)
\end{aligned}
$$

The resolution (dpi) of an image was obtained by viewing its properties, as shown in Figure 4.12.

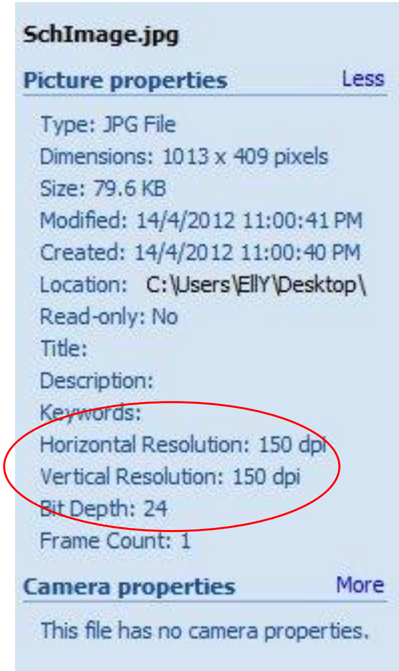
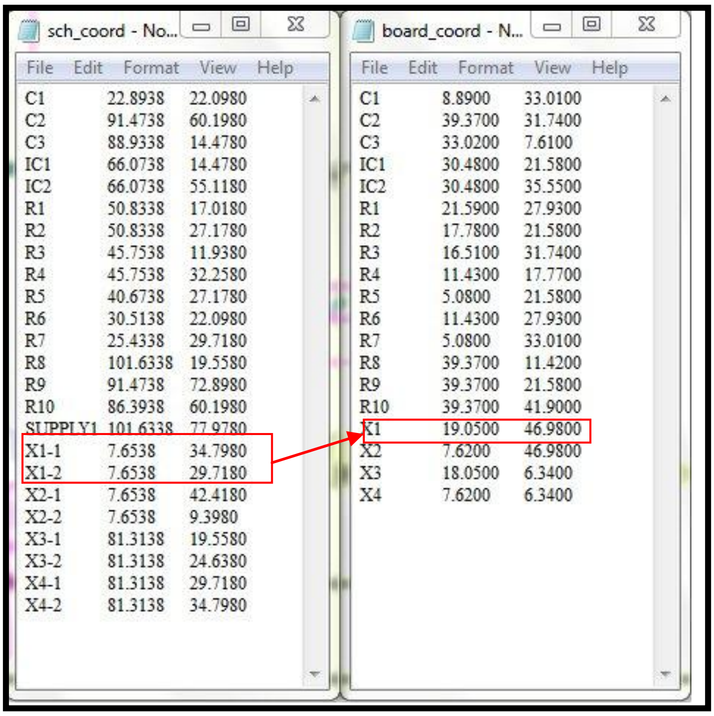


Figure 4.12: Image Properties of SchImage Showing Its Resolution

Another concern was that some of the component's names in schematic might differ from the names in board, as shown in Figure 4.13. This problem needed to be countered by matching for the full string in the board_coord.mnt first, and matching its first string if no full string was matched.



sch_coord - No...

File Edit Format View Help

C1	22.8938	22.0980
C2	91.4738	60.1980
C3	88.9338	14.4780
IC1	66.0738	14.4780
IC2	66.0738	55.1180
R1	50.8338	17.0180
R2	50.8338	27.1780
R3	45.7538	11.9380
R4	45.7538	32.2580
R5	40.6738	27.1780
R6	30.5138	22.0980
R7	25.4338	29.7180
R8	101.6338	19.5580
R9	91.4738	72.8980
R10	86.3938	60.1980
SUPPLY1	101.6338	77.9780
X1-1	7.6538	34.7980
X1-2	7.6538	29.7180
X2-1	7.6538	42.4180
X2-2	7.6538	9.3980
X3-1	81.3138	19.5580
X3-2	81.3138	24.6380
X4-1	81.3138	29.7180
X4-2	81.3138	34.7980

board_coord - N...

File Edit Format View Help

C1	8.8900	33.0100
C2	39.3700	31.7400
C3	33.0200	7.6100
IC1	30.4800	21.5800
IC2	30.4800	35.5500
R1	21.5900	27.9300
R2	17.7800	21.5800
R3	16.5100	31.7400
R4	11.4300	17.7700
R5	5.0800	21.5800
R6	11.4300	27.9300
R7	5.0800	33.0100
R8	39.3700	11.4200
R9	39.3700	21.5800
R10	39.3700	41.9000
X1	19.0500	46.9800
X2	7.6200	46.9800
X3	18.0500	6.3400
X4	7.6200	6.3400

Figure 4.13: Difference between Component's Name in Schematic and Board

4.1.4 Development of XY_Coord_Calibration.vi

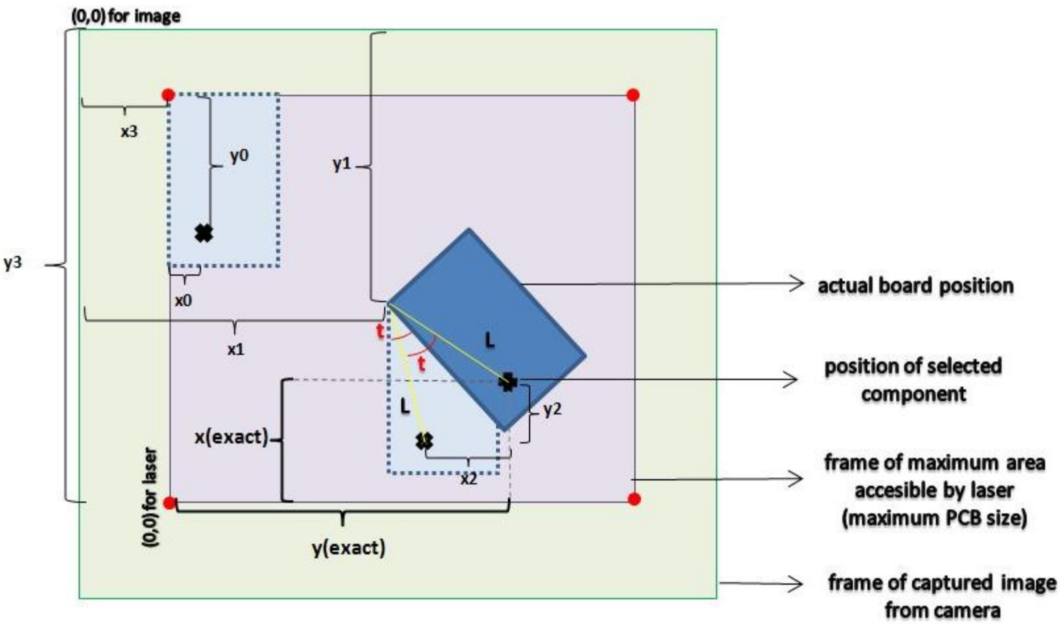


Figure 4.14: Overview of Sets of XY Coordinates

As shown in Figure 4.14, there were several sets of XY coordinates to be obtained and calculated before the exact coordinate of the selected component from the origin (0,0) was located. It was noted that the origin (0, 0) for the image and for the laser was different, and hence it affected the x-axis and y-axis of both image and laser. It needed to be taken care of when doing the calculation. All the calculations, done in unit of millimetre, are shown in Table 4.1.

Table 4.1: Calculations to Get Exact XY Coordinate of Component

Image Dimension	The dimension was 1024 pixels * 768 pixels. Conversion from pixel to millimetre was done by using Equation 4.2: *Distance(mm) = Distance(pixel)/(resolution(dpi)*conversion(inch/mm))* *= Distance (pixel) / (96 dpi * (1/25.4))* *= Distance (pixel) / 3.78* (4.2) The resolution in dpi of the image was obtained by viewing its properties, shown in Figure 4.15.
(x0, y0)	It was extracted out from brd_coord.mnt in Get_Coordinate.vi, as shown in Figure 4.4 and Figure 4.5 in Section 4.1.2.
(x1, y1)	It was obtained from the image captured by the camera, after image calibration. However, the obtained coordinate was in pixel which was converted into millimetre by Equation 4.2.
(x2, y2)	It was calculated by treating the movement of the component as a circle with length = L, calculated by Equation 4.3: $L = \sqrt{x0^2 + y0^2}$ (4.3) The circle trigonometric is shown in Figure 4.16. It is thereby followed by Equation 4.4, 4.5 and 4.6. $t = t1 - \tan^{-1}\frac{y0}{x0}$ (4.4) $x2 = L\cos t$ (4.5) $y2 = L\sin t$ (4.6)
(x3, y3)	It was obtained by setting the laser at its origin, and getting the total pixels from the origin of the image to the origin of the laser, which is (46, 740) as shown in Figure 4.17. The unit was in pixel as well, and the

	conversion was done by using Equation 4.2.
x (exact)	y3 – y1 – y0 + y2
y (exact)	x1 – x3 +x0 + x2

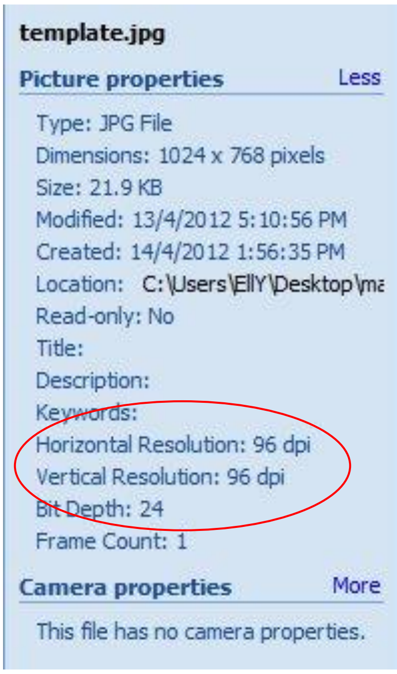

template.jpg
Picture properties Less
Type: JPG File
Dimensions: 1024 x 768 pixels
Size: 21.9 KB
Modified: 13/4/2012 5:10:56 PM
Created: 14/4/2012 1:56:35 PM
Location: C:\Users\EllY\Desktop\ma
Read-only: No
Title:
Description:
Keywords:
Horizontal Resolution: 96 dpi
Vertical Resolution: 96 dpi
Bit Depth: 24
Frame Count: 1
Camera properties More
This file has no camera properties.

Figure 4.15: Image Properties of template.jpg Showing Its Resolution

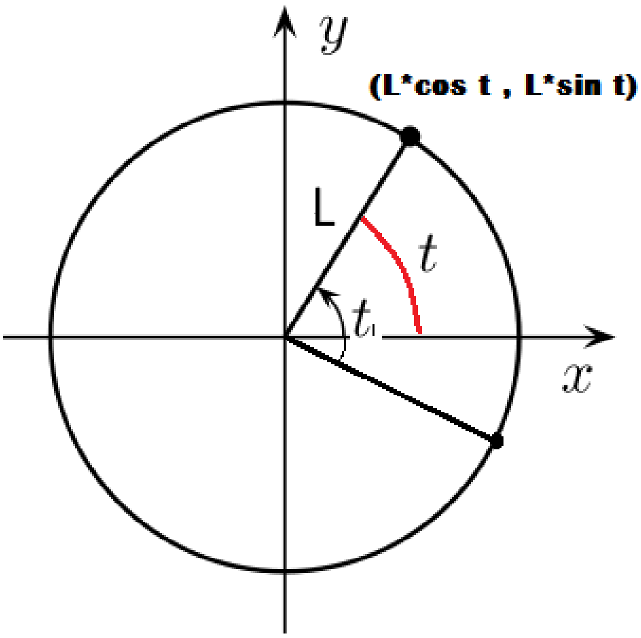


Figure 4.16: Circle Trigonometric

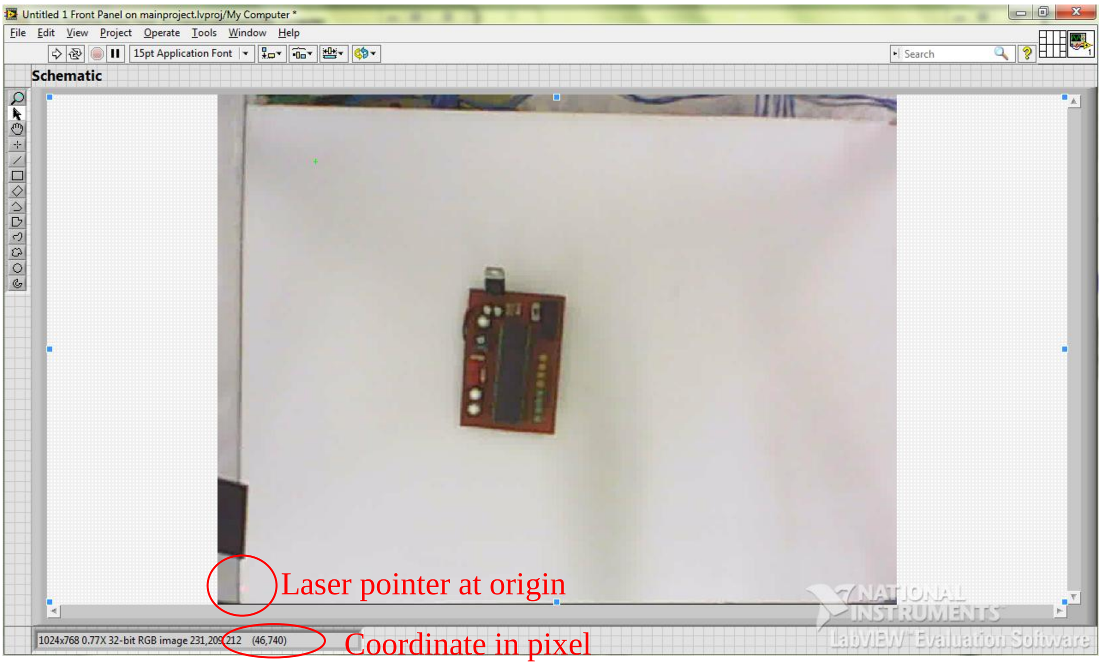


Figure 4.17: Origin of Laser Pointer

In order for the motor to drive the laser to point to the exact location of the component, the motor steps needed to be identified relatively to distance. By obtaining the total number of pixels between two points in image when the pointer was moved by a certain motor steps, the relationship of motor steps and distance could be identified, which is shown in Table 4.2.

Table 4.2: Relationship of Motor Steps and Point Distance

	Motor Steps	**Point Distance (pixel)**	**Motor Steps / Pixel**
X-axis	1000	1087	0.92
Y-axis	2000	123	16.3

4.1.5 Development of Email_Module.vi

When the user chose to send the CSV file through email, Email_Module.vi was executed. User needed to enter the recipient's name and email address, subject and body, and optionally email address in carbon copy (CC). The CSV file was automatically attached to the email. All the configurations of Gmail were set internally. However, sometime the email was not sent successfully due to several reasons, such as invalid email address, proxy not supported and etc. If it happened, a message of "Email is not sent" was displayed to the user, else it showed "Email is sent". The complete flowchart of this module is shown in Figure 4.18.

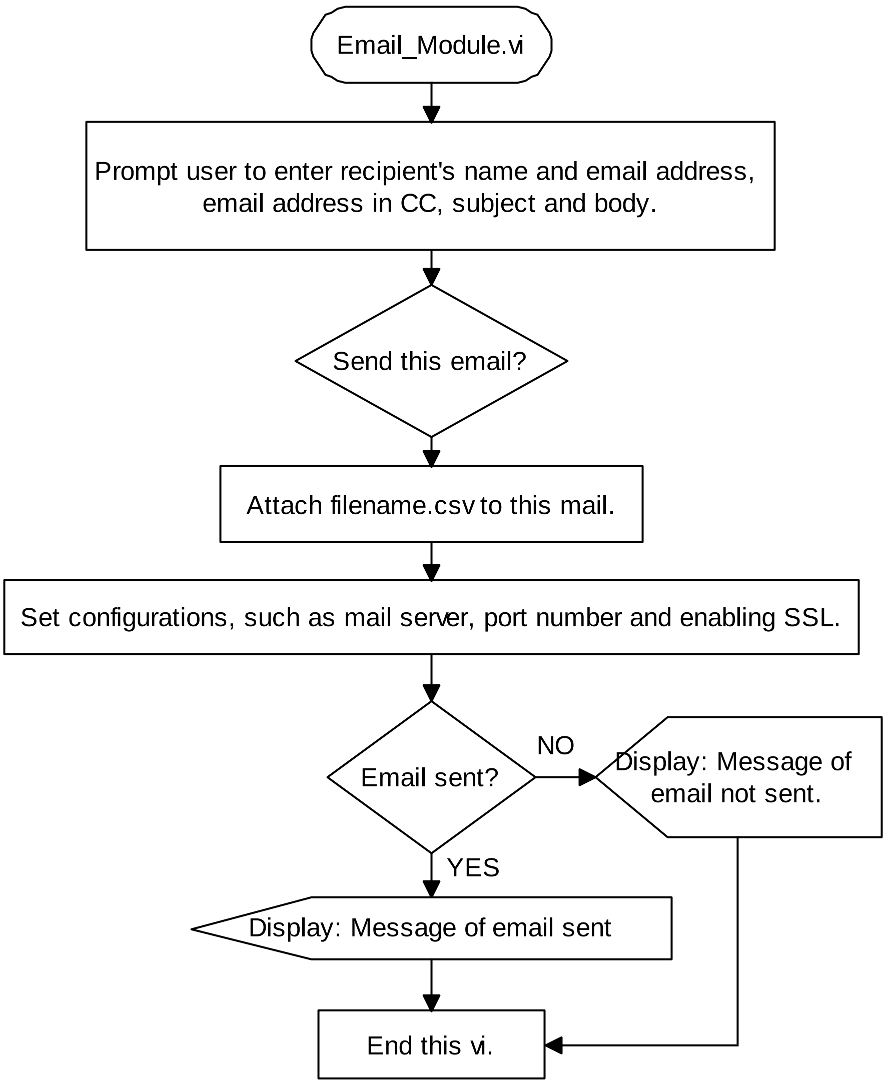


Figure 4.18: Flowchart of Email_Module.vi

4.2 Integration of Hardware and Software

There were two buttons in the main menu for the user to select: Set path for EAGLE and Start. For the first use in a computer, the user was required to set the path for the bin folder that contained EAGLE.exe, the screenshot of the process of changing the path is shown in Figure 4.19.

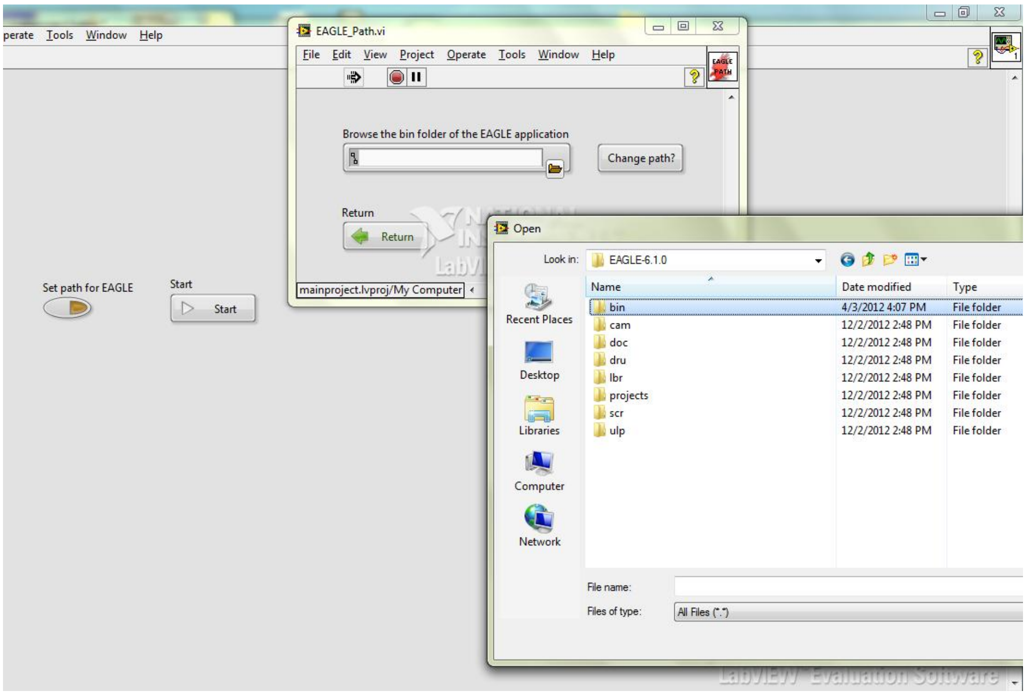


Figure 4.19: Screenshot of Changing EAGLE Path

After setting the EAGLE path, user clicked on the “Start” button to start browsing the desired board in EAGLE, as shown in Figure 4.20. It was a one-step process as the user was only required to select the board file, and the program was automatically linked to its schematic file and image template.

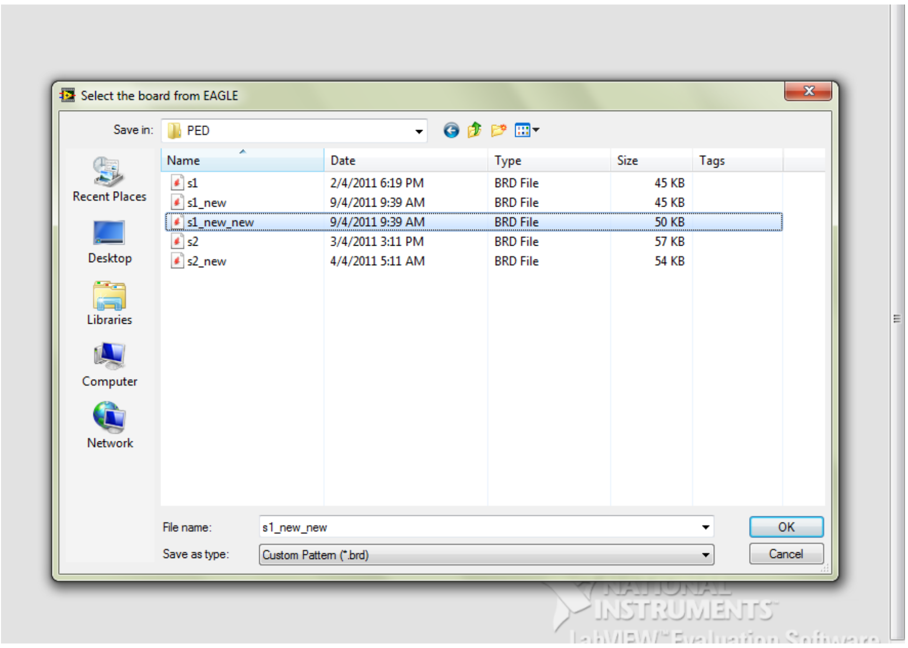


Figure 4.20: Screenshot of Selecting Board File

Both board file and schematic file were opened during the execution of the EAGLE script. User was given option whether to keep those files open or to quit them, as shown in Figure 4.21.

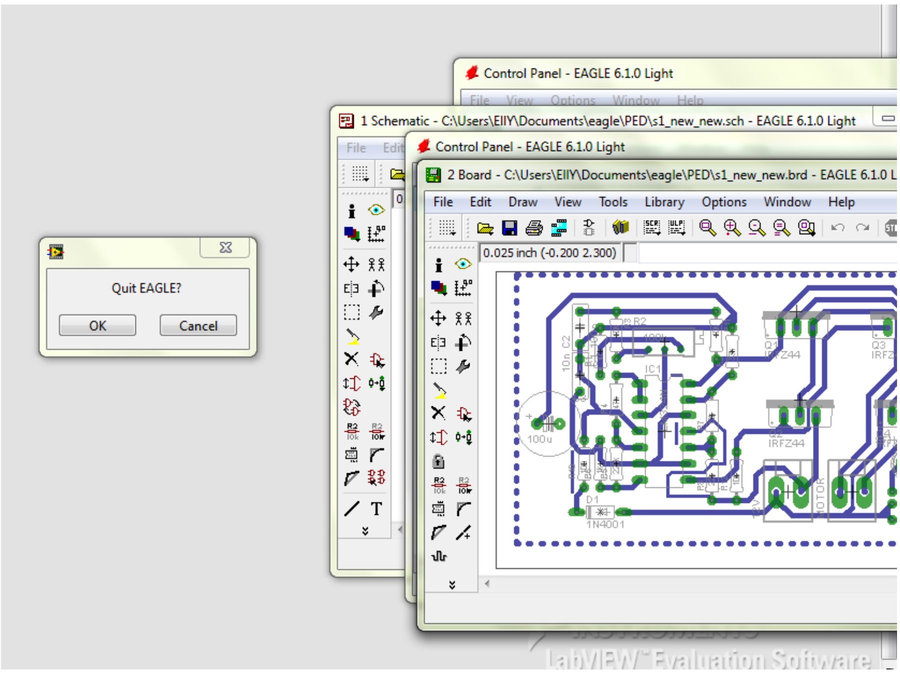


Figure 4.21: Screenshot of EAGLE processing

At this moment, the front panel of the Get_Coordinate.vi appeared, displaying the image of the schematic. It prompted the user to click on the schematic to select the component, as shown in Figure 4.22.

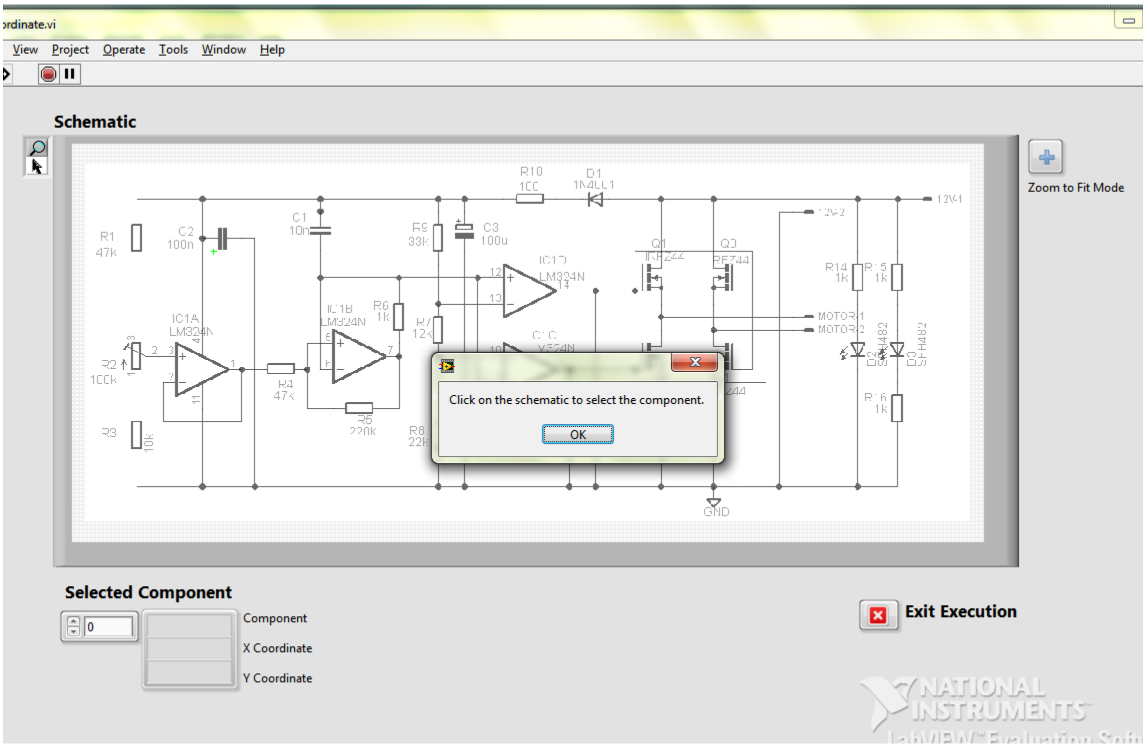


Figure 4.22: Screenshot of Component Selection

After the user selecting a component (which is C2), calibration was done to get the coordinate of the component in real world. The motors drove the laser pointer to the desired location for the user to probe. In the front panel, a dialogue box was displayed to ask whether the user intends to add a comment on this probing, as shown in Figure 4.23. All the data was saved in a CSV file.

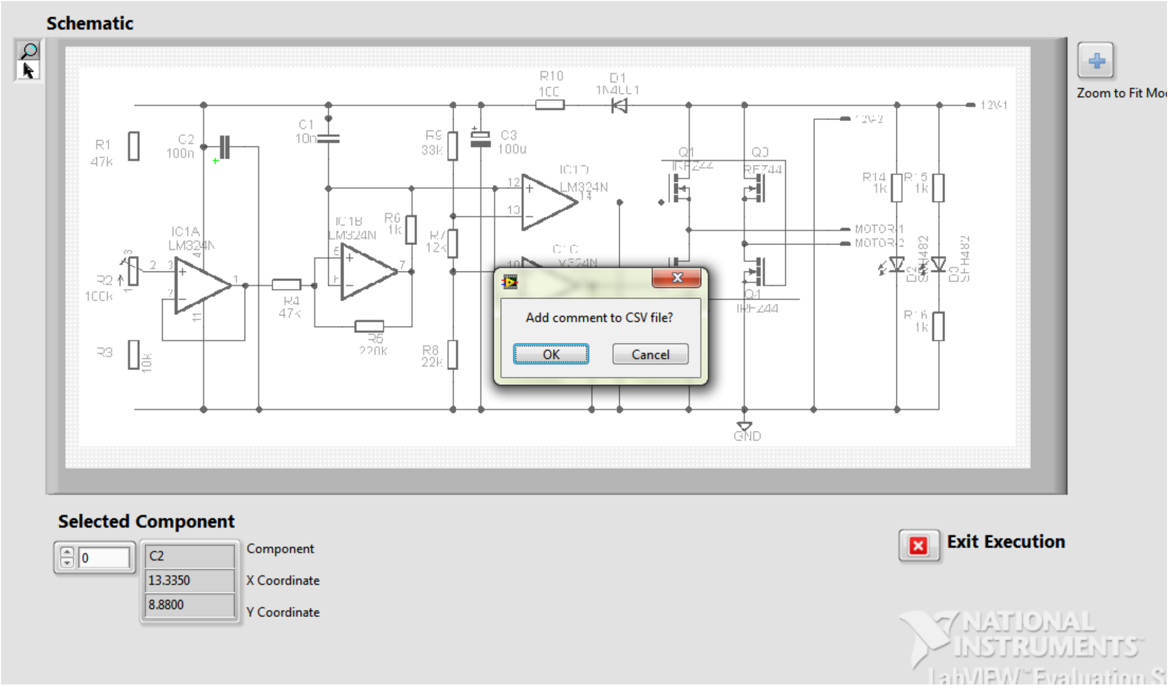


Figure 4.23: Screenshot of Prompting User to Add Comment

If the user clicked "Ok" to add comment to CSV file, another front panel popped out and the user clicked on "Return" after adding in comment, as shown in Figure 4.24.

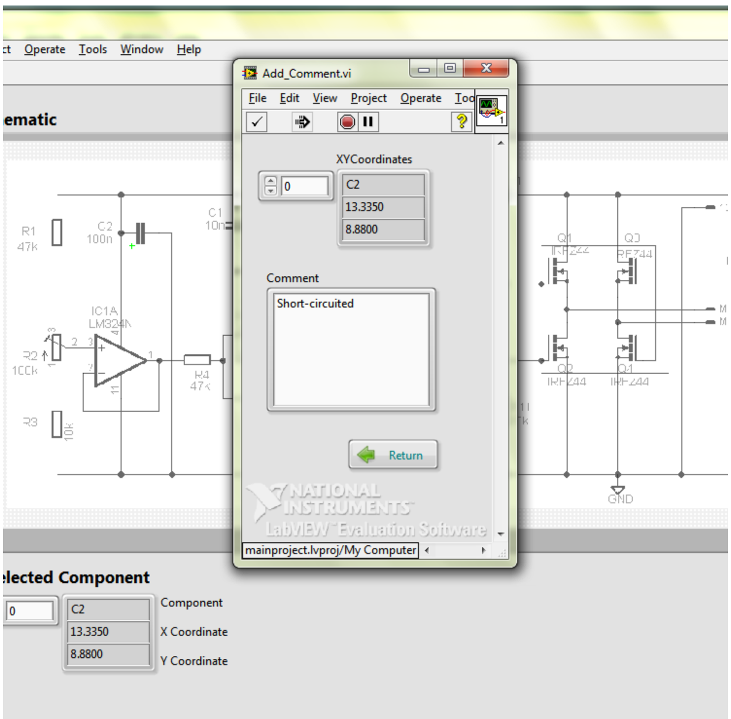


Figure 4.24: Screenshot of Adding Comment

User could continue to another component selection after returning from adding comment. As shown in Figure 4.25, another component Q1 was selected. Figure 4.26 shows the dialogue box popped out when user clicked on an invalid component, i.e. an empty area.

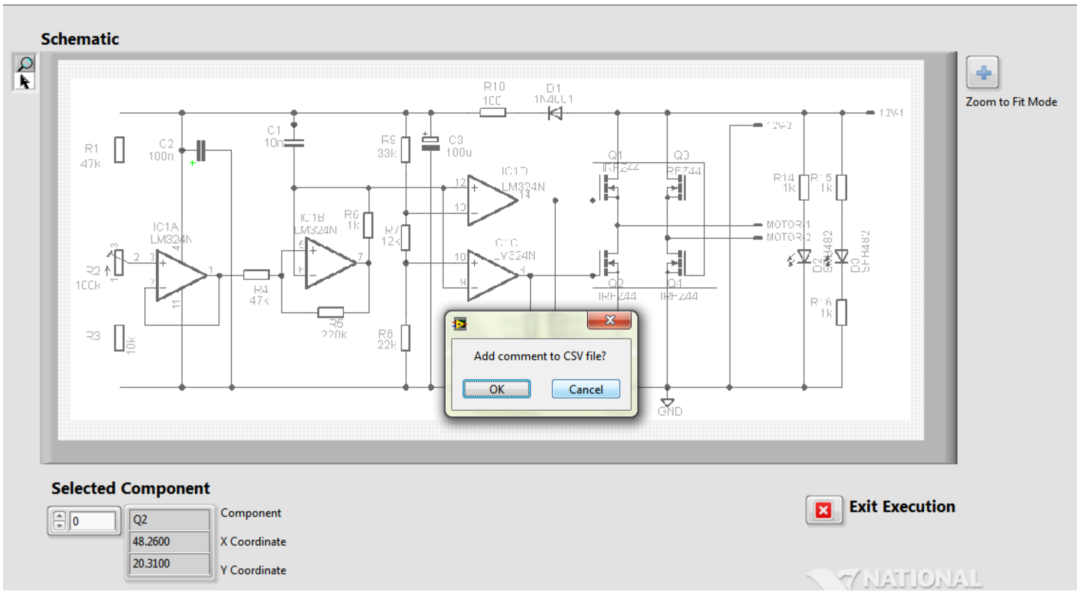


Figure 4.25: Screenshot of another Component Selection

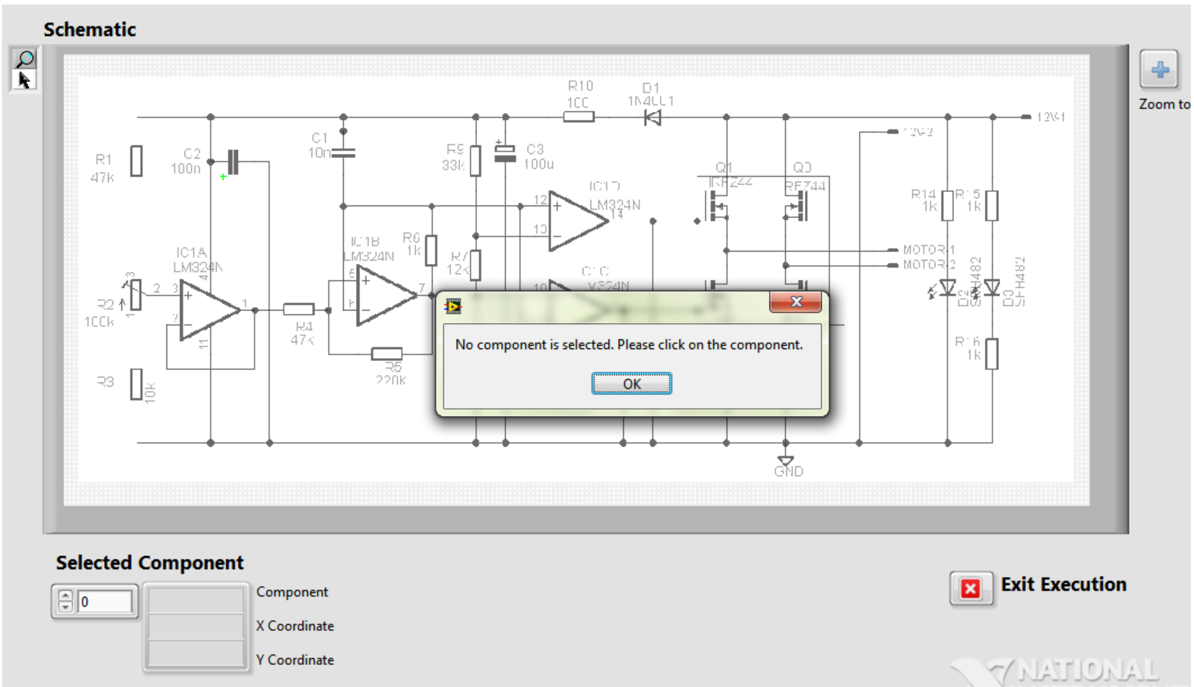


Figure 4.26: Screenshot of No Component Selected

When user had done all the testing on this DUT, he clicked on the button "Exit Execution" to exit from the front panel of Get_Coordinate.vi and to return to the main menu. In the meantime, user was given an option whether to send the CSV file to certain email address, as shown in Figure 4.27. Figure 4.28 shows the front panel of Email_Module.vi for user to key in all the email information.

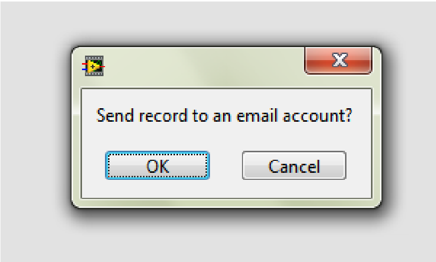


Figure 4.27: Screenshot of Prompting User to Send Email

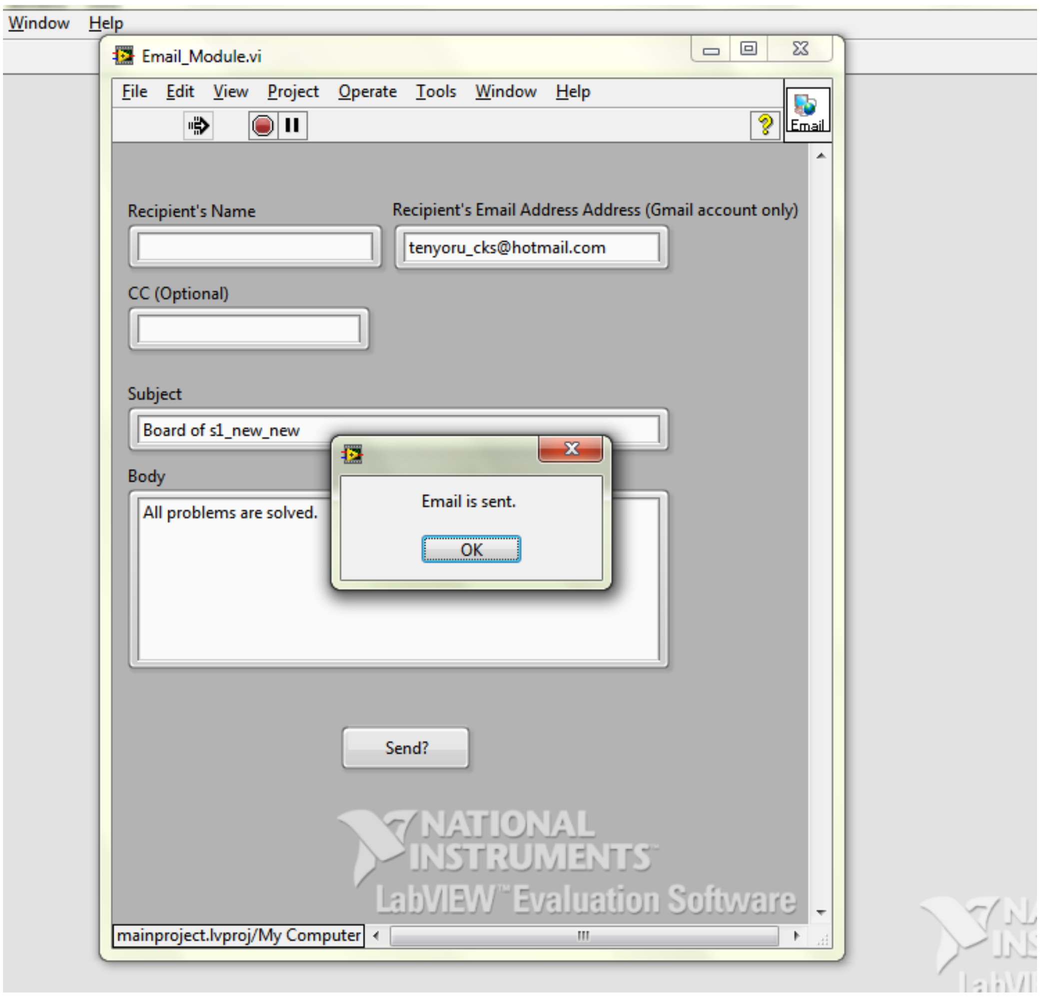


Figure 4.28: Screenshot of Sending Email

The whole process for probing a board was completed and the program returned to the main menu for the user to start browsing another board file from EAGLE. The program was terminated only will the user stopped the whole VI.

The results on the real PCB, when the user clickedon the components (C2 and Q1), are shown in Figure 4.29 and Figure 4.30. Figure 4.31 shows the location of the two components in the board design file of EAGLE (the highlighted parts).

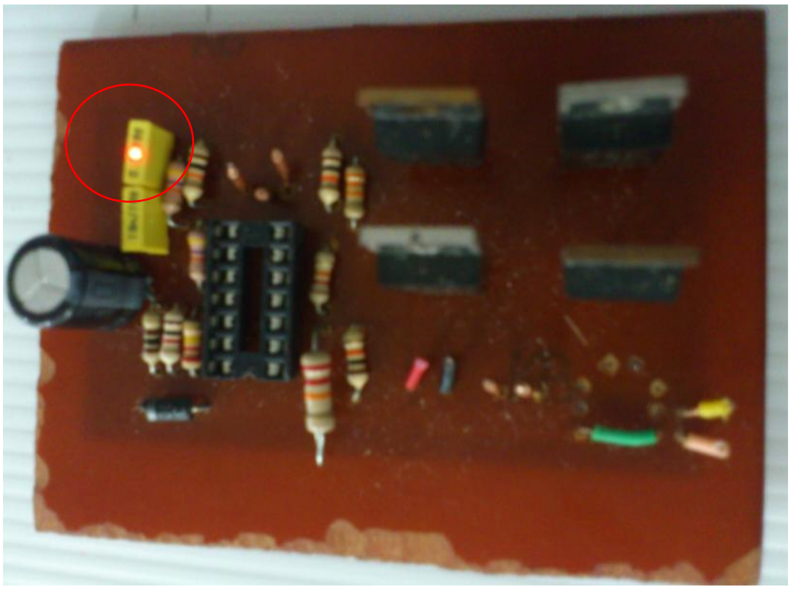


Figure 4.29: Laser Pointing at C2

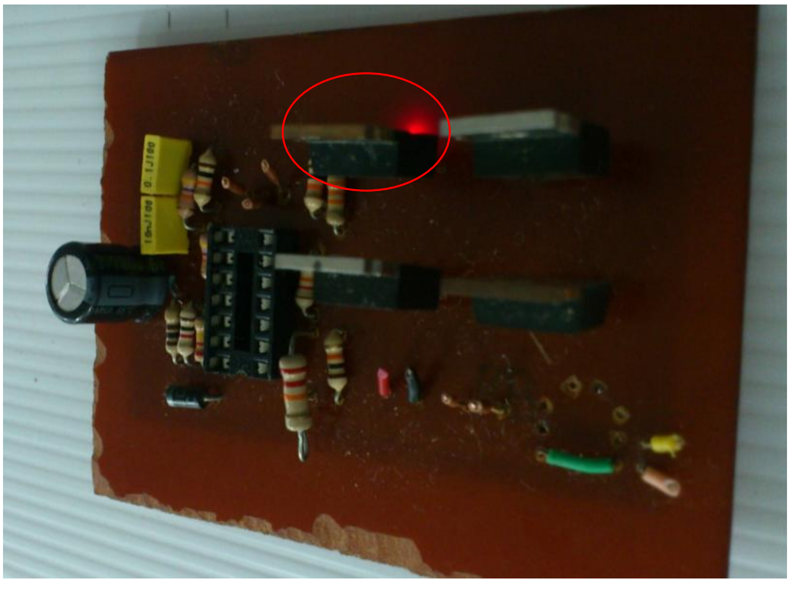


Figure 4.30: Laser Pointing at Q1

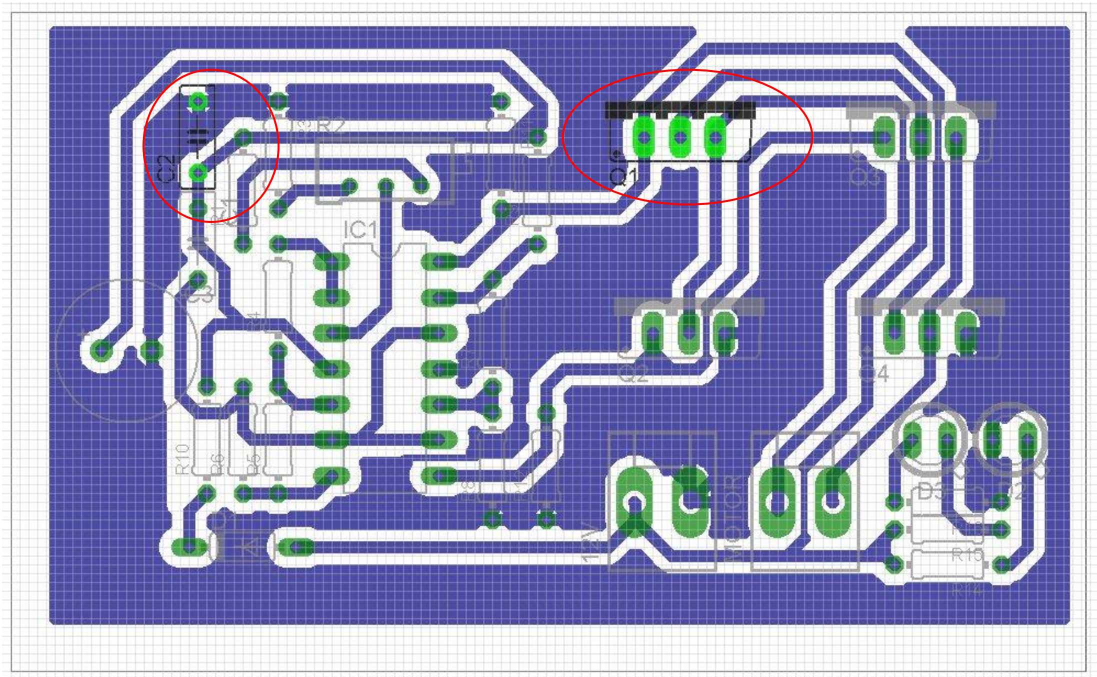


Figure 4.31: Location of C2 and Q1 in the Board File

By comparing the pointed location of the components in Figure 4.29 and Figure 4.30 with their location in board in Figure 4.31, it shows that the correct components were located in the real board by only clicking on component in the schematic file.

During the process, the data of C2 and Q2 were saved in a CSV file with the same name as the board, which is s1_new_new.csv. The date and time of the probing process as well as the comment from the user were arranged orderly in the file, as shown in Figure 4.32. Figure 4.33 shows the email received with the CSV file attached. The subject and body were keyed in by the sender.

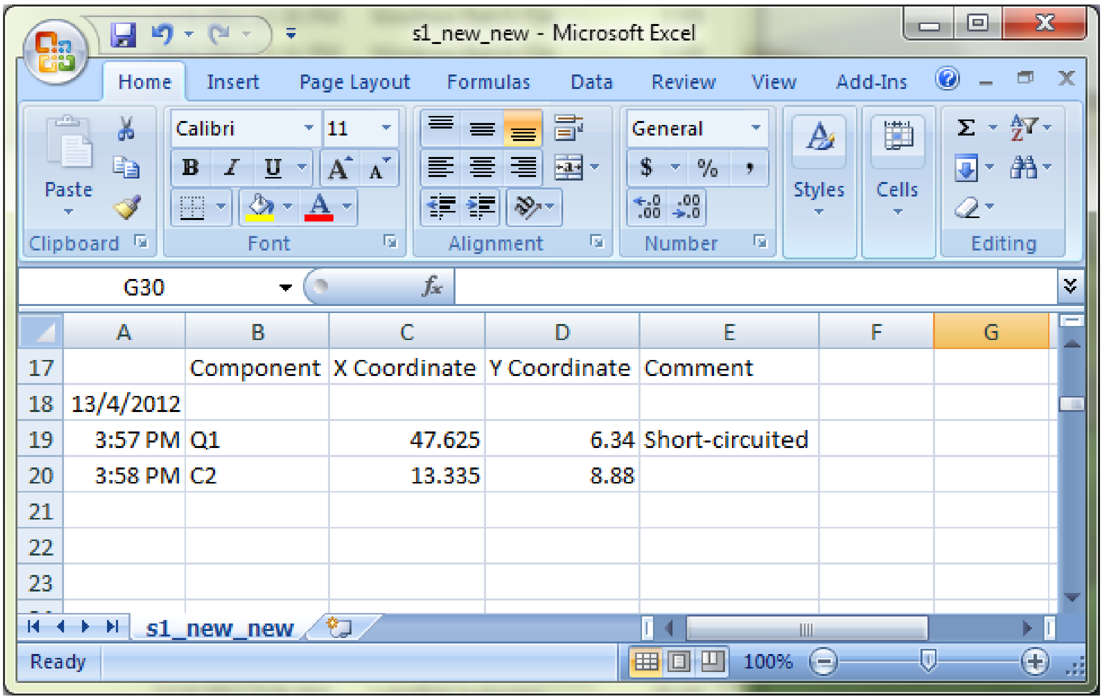


Figure 4.32: Screenshot of CSV File

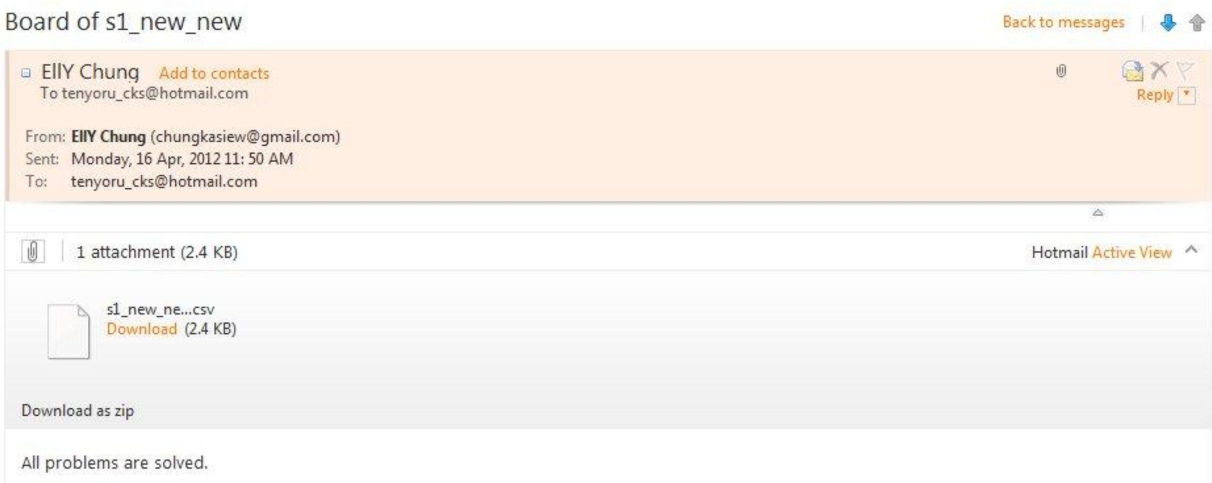


Figure 4.33: Screenshot of Email Received

The images of the overall hardware structure are shown in Appendix A, the circuit designs of the printed circuit boards are shown in Appendix B, and the block diagrams of each VI in this project are shown in Appendix C.

4.3 Multipurpose Flexible Holder

4.3.1 Development of Mechanical Structure

The mechanical structure was developed building up small parts separately and integrating all these parts together, as shown in Figure 4.34 and Figure 4.35. The precise measurements and arrangement were done in the design software SolidWorks before developing them by real material.

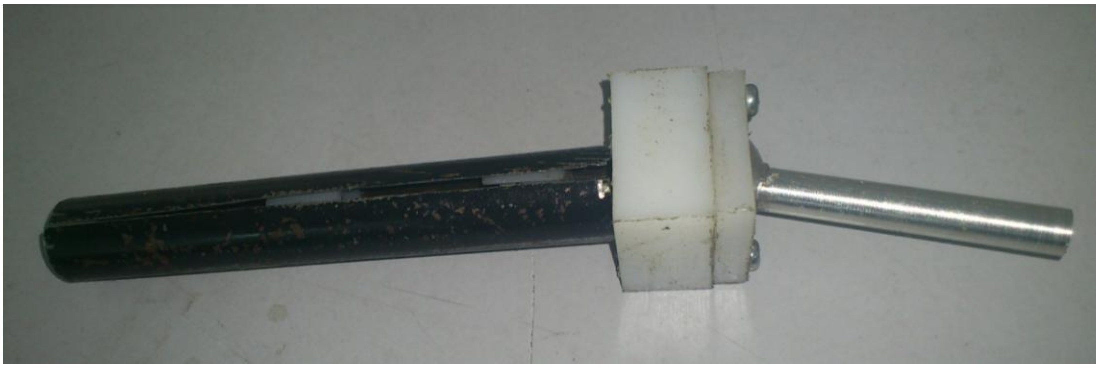


Figure 4.34: Outer Look of Holder

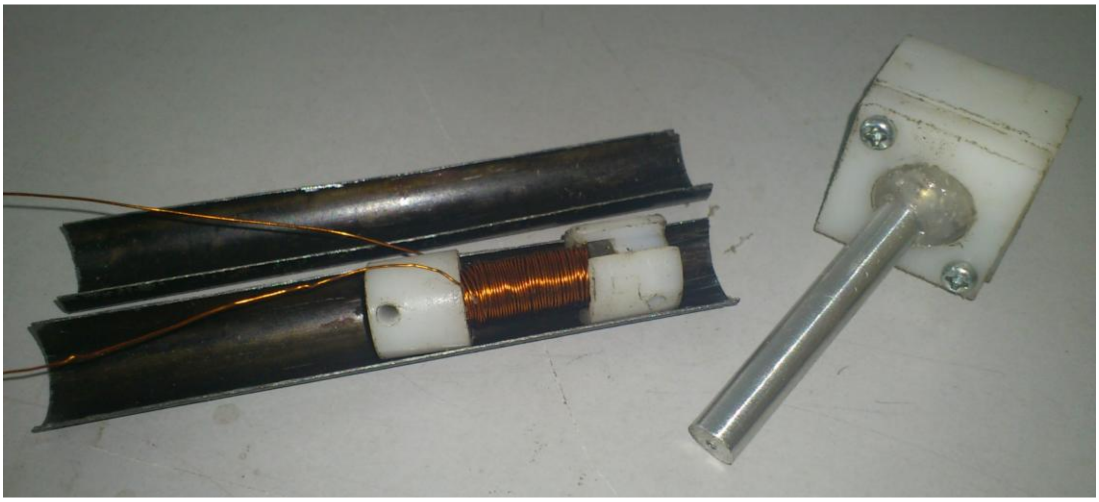


Figure 4.35: Inner Look of Holder

Generally, the principle of electromagnetism was working. The core became a magnet that attracted (held) the side magnets to it when current was applied in certain direction, indicating the “lock” action of the holder. When the direction of current was inverted, the poles of the magnetized core were inverted too and it repelled both the side magnet away, creating the “unlock” action.

However, as the amplitude of the magnetic force of the two side magnets was too strong, the repelling force was not able to “unlock” the holder. If weak side magnets were used, the attracting force was not strong enough to “lock” the holder in a fixed position. Hence, searching for magnets with the right magnetic force was very important and it was yet to be determined due to the limited time frame for this project. The magnetic field magnitude inside a solenoid is given by Ampère’s law as shown in the Equation 4.7:

$$B = \mu_0 * I * N / L \qquad (4.7)$$

where
B = magnetic field, T
μ_0 = permeability of free space, T.m/A
I = current flow in the wire, A
N = number of loops
L = length of solenoid, m

Both the magnetic field of the solenoid and the side magnets needed to be properly matched in order to get the perfect attraction force and repelling force.

4.3.2 Development of Speech_Recognition.vi

The speech recognition required two pre-defined module from LabVIEW – Speech_Setup.vi and Speech_Check.vi. It allowed user to set the list of speech, which were lock, unlock and stop in this project. When the user spoke “Lock”, it sent a signal to activate the locking process of the holder. A signal of unlocking the holder was sent when the phrase “Unlock” was detected. In order to stop the speech recognition, “Stop” was used as the command. The complete flow of the flow chart for this VI is shown in Figure 4.36. The result when the "Lock" is detected by the program is shown in Figure 4.37.

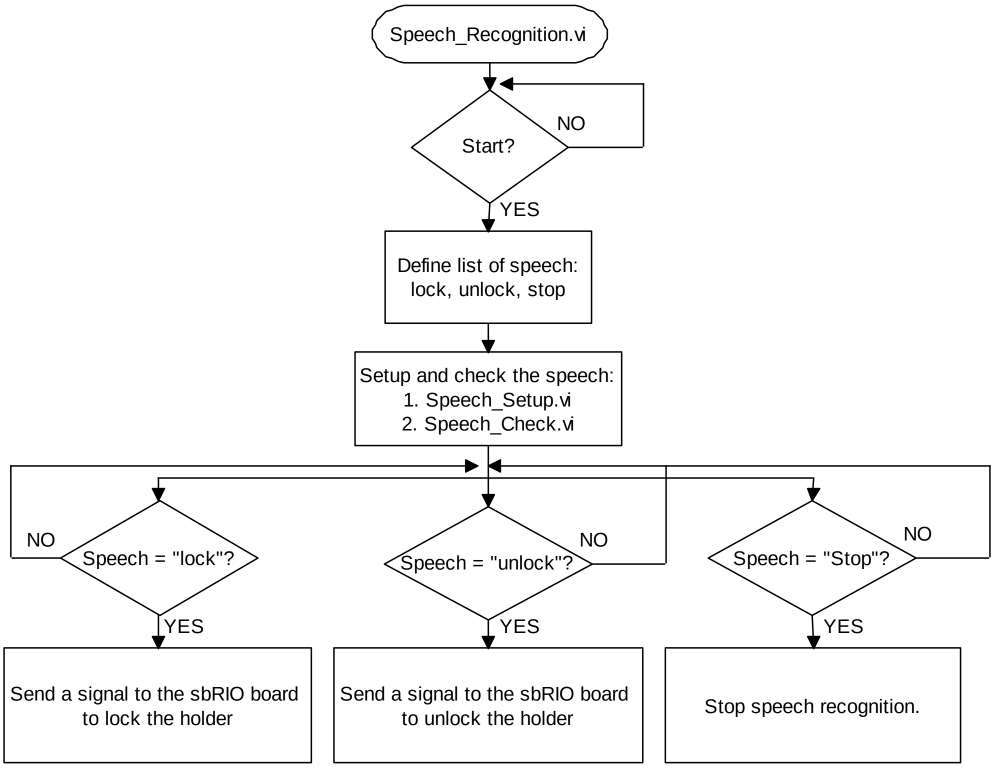


Figure 4.36: Flowchart of Speech_Recognition.vi

Figure 4.37: Screenshot of User Speaking "Lock"

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

As a conclusion, the objectives in the first section of this project have been met. The 2-D Location Pointing System for Individual Component on Device under Test has allowed user to link the schematic in computer to the PCB in real world. When user clicks on any component in the schematic, the laser pointer will lead the user to the real component on the PCB which will speed up the process of component searching. It has been achieved by linking the PCB drawing software that contains the schematic, to LabVIEW that will trigger the movement of the laser pointer to the desired location. The PCB drawing software used is EAGLE as it provides full accessibility of its internal script whereas the intermediate link used is a Windows script - batch file. It executes the EAGLE script to run ULP to generate coordinate files which can be accessed by LabVIEW.

In addition to controlling the system by using software, external buttons and switches are integrated in case of the malfunction of the software. LEDs and LCD are used as indicators and display so that users will be on track of the current processing steps of the system.

Documentation of the process is done throughout the component searching procedure, where the selected component, its coordinate and optionally user's comment are saved in a CSV file, and sending this CSV file through email directly from the system is available.

The second objective of this project is implementation of a multipurpose flexible holder with speech driven function. It has been met theoretically as both the electromagnetism mechanism and the function of speech recognition are working. However, the overall functionality of the holder is yet to be implemented after the magnets with appropriate magnetic field are obtained.

5.2 Recommendations

There are still some limitations in this project due to the time constraint, which subsequently provide a wide space for optimization and further improvement in the future. First and foremost, the project designed can only access one type of PCB drawing software, which is EAGLE. It is because different software will have their specific type of scripting, and it requires individual development of the script if other PCB drawing software is to be integrated in this project. In the future, more choices of PCB drawing software, such as Altium and Proteus, will be included in the project so that schematic developed by different software can be used by the user.

In addition, the function of sending CSV file to email limits the sender's email to only Gmail server. It is because different mail service providers will require different configurations, such as its server and port number. A recommendation is to include more mail service providers, such as Hotmail and Yahoo, so that the user who does not have a Gmail account can have an option to choose other mail types.

REFERENCES

Agilent. (2009). Agilent Probe Positioners Datasheet.

Building a Stand-Alone Application. (2009). Retrieved December 8th, 2011, from http://zone.ni.com/reference/en-XX/help/371361B-01/lvhowto/building_a_stand_alone_app/

Cadsoft. (2011a). EAGLE Manual Version 6

Cadsoft. (2011b). Learn More About CadSoft EAGLE. Retrieved Jan 3rd, 2012, from http://www.cadsoftusa.com/

Clinick, A. (2000). Introducing JScript .NET. Retrieved December 10th, 2011, from http://msdn.microsoft.com/en-us/library/ms974588.aspx#scripting0714_topic1

Configuring other mail clients. (2011). Retrieved December 10th, 2011, from http://support.google.com/mail/bin/answer.py?hl=en&answer=13287

Giancoli, D. C. (2005). Physics (6th ed., pp. 567): Pearson Education.

Incoming and Outgoing Mail Server Settings for Hotmail, Yahoo! Mail, GMail, MSN, AOL and more. (2011). Retrieved April 16th, 2012, from http://www.emailaddressmanager.com/tips/mail-settings.html

Jezek, G. (2006). Electromagnetism. Retrieved December 15th, 2011, from http://www.howmagnetswork.com/Electromagnetism.html

Kiesewetter, J., Kreissig, S., & Kanev, S. (2009). Probe holder for a probe for testing semiconductor components: Google Patents.

Laurie, V. (2009a). Batch Files (Scripts) in Windows. Retrieved December 9th, 2011, from http://commandwindows.com/batch.htm

Laurie, V. (2009b). PowerShell in Windows XP, Windows Vista, and Windows 7. Retrieved December 10, 2011, from http://commandwindows.com/powershell.htm

Laurie, V. (2009c). The Windows Command Line, Batch Files, and Scripting. Retrieved December 9th, 2011, from http://commandwindows.com/

Magna-Vue. (2010). Oscilloscope Probe Holder. Retrieved March 15, 2012, from http://www.magna-vue.com/oscilloscope.html

Moore's Law 40th Anniversary. (2005). Retrieved December 8th, 2011, from http://www.intel.com/pressroom/kits/events/moores_law_40th/index.htm

Moore, G. (2003). Moore’s Law: Raising the Bar. *ISSCC February*.

PCB Layout Data. Retrieved Jan 3rd, 2012, from http://www.eurocircuits.com/index.php/technology-guidelines/pcb-layout-data

Product Information: What is NI LabVIEW. (2011). Retrieved December 8th, 2011, from http://www.ni.com/labview/whatis/

Seidle, N. (2008). Lecture 8 - Eagle: Schematics. Retrieved Jan 3rd, 2012, from http://www.sparkfun.com/tutorials/108

What is JScript. (1997). Retrieved December 10th, 2011, from http://www-sbras.nsc.ru/docs/ms/js/0.htm

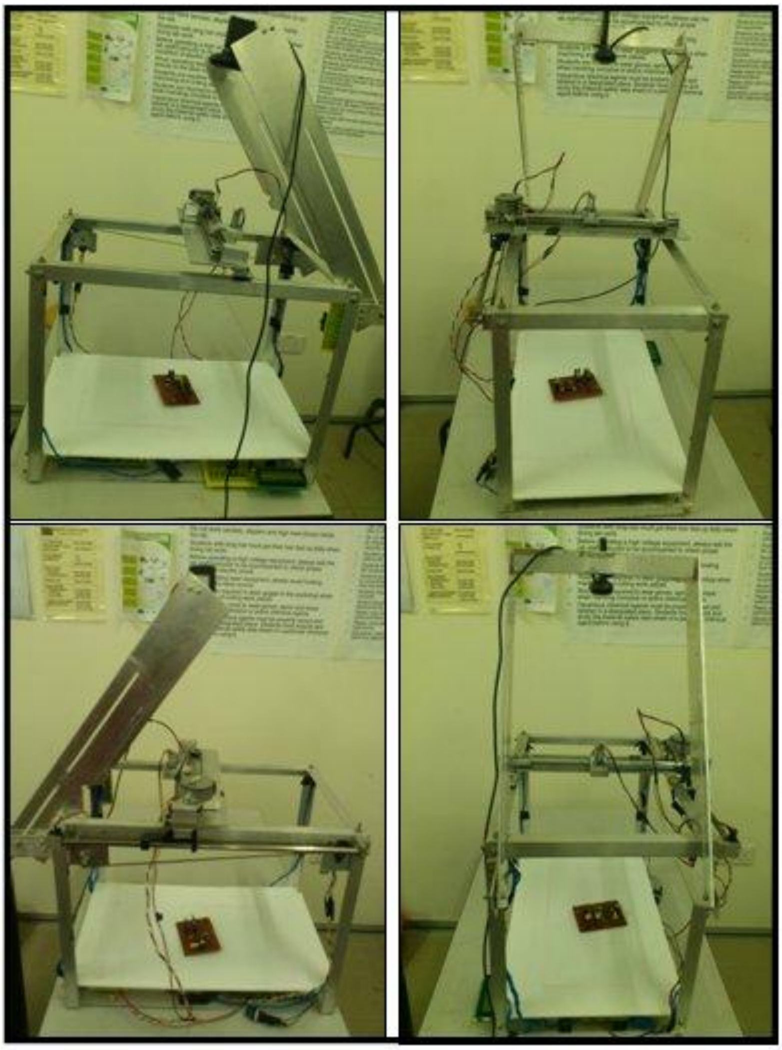
What is WSH. (2009). Retrieved December 10th, 2011, from http://msdn.microsoft.com/en-us/library/shzd7dy4(v=VS.85).aspx

Wong, K. (2010). Integrated Circuit and Method of Testing the Integrity of Electrical Connection of the Circuit to External Equipment: Google Patents.

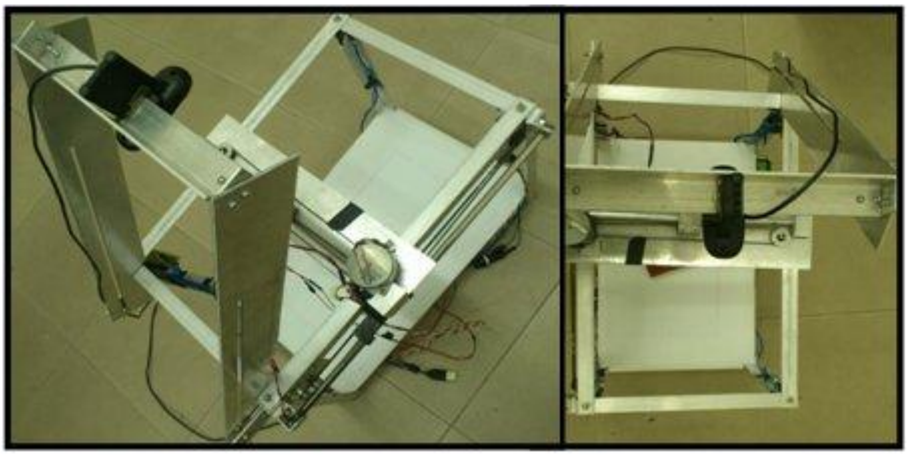
APPENDICES

APPENDIX A: Hardware Photos

Four-sides View of Structure

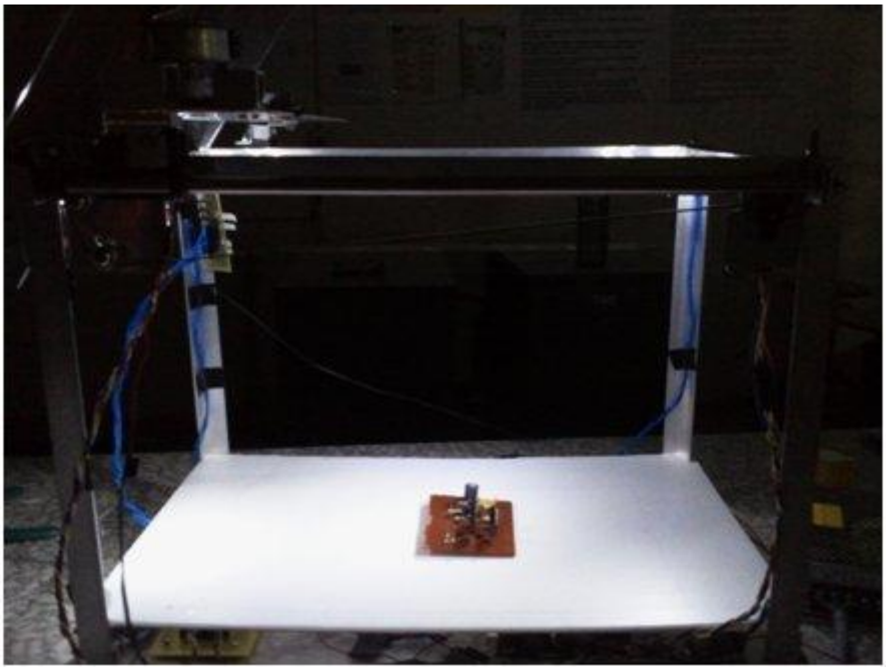


Top View of Structure

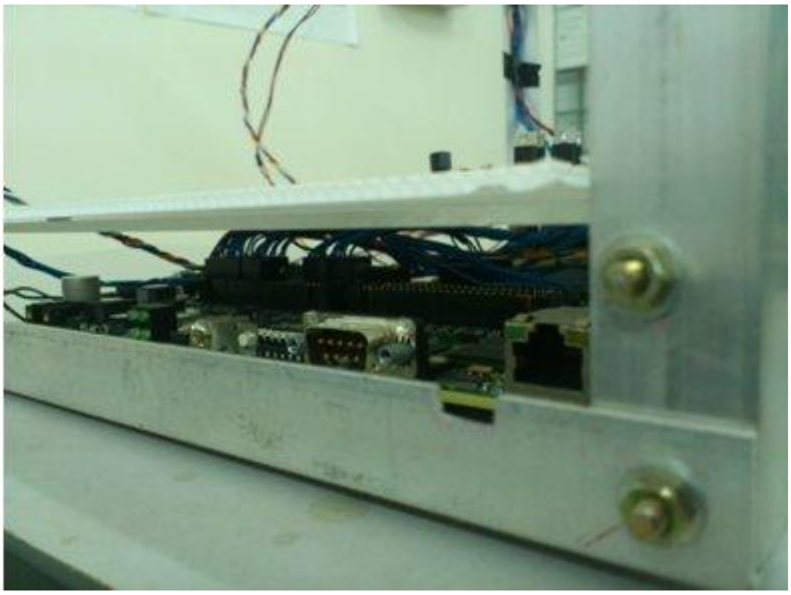


Location of LED Lights

LED Lights On in Dark Environment



Electronics Boards underneath the Structure



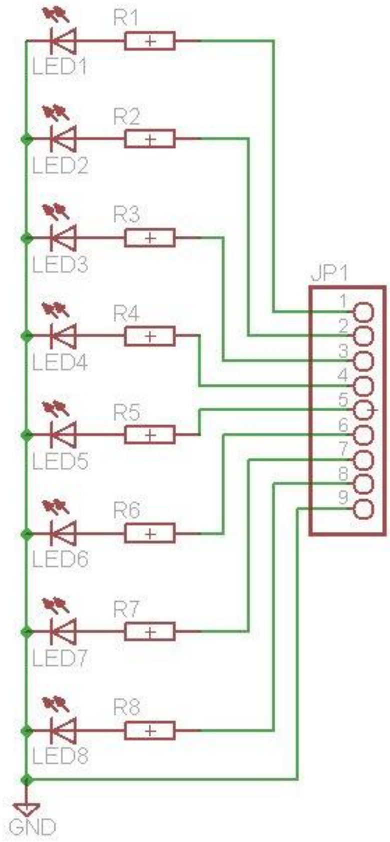
Arrangement of Boards underneath the Structure

NI sbRIO-9632XT board

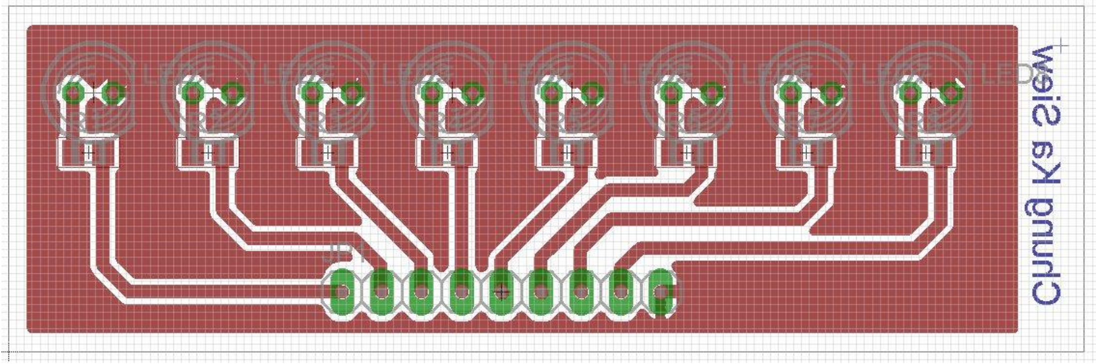
Power Supply for NI sbRIO-9632XT board

APPENDIX B: Schematic and Board Design

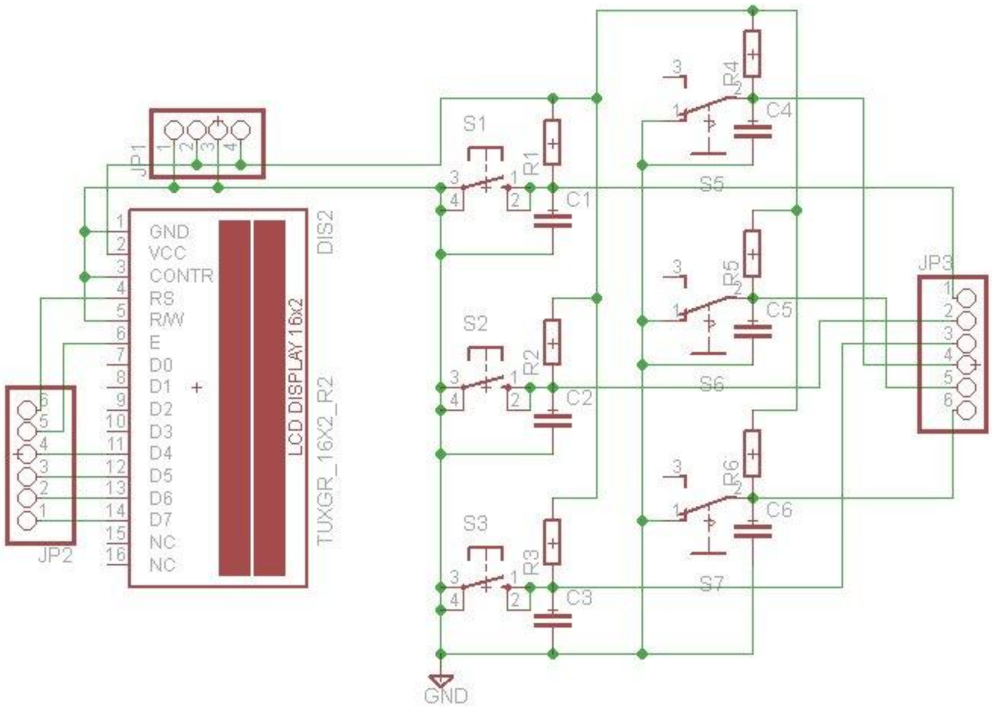
Schematic Design of Board 1 (for LED indicators)



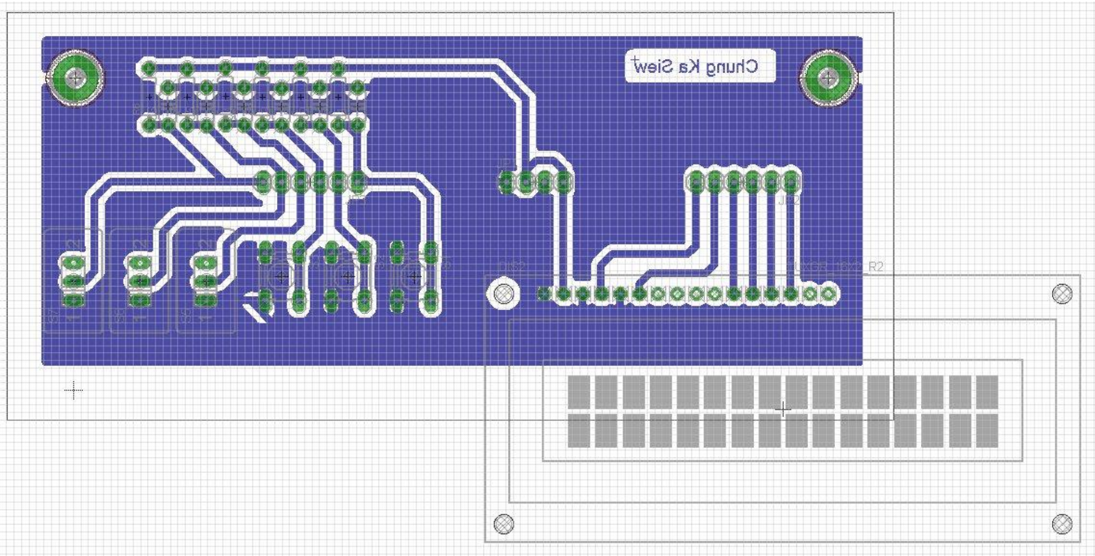
Board Design of Board 1 (for LED indicators)



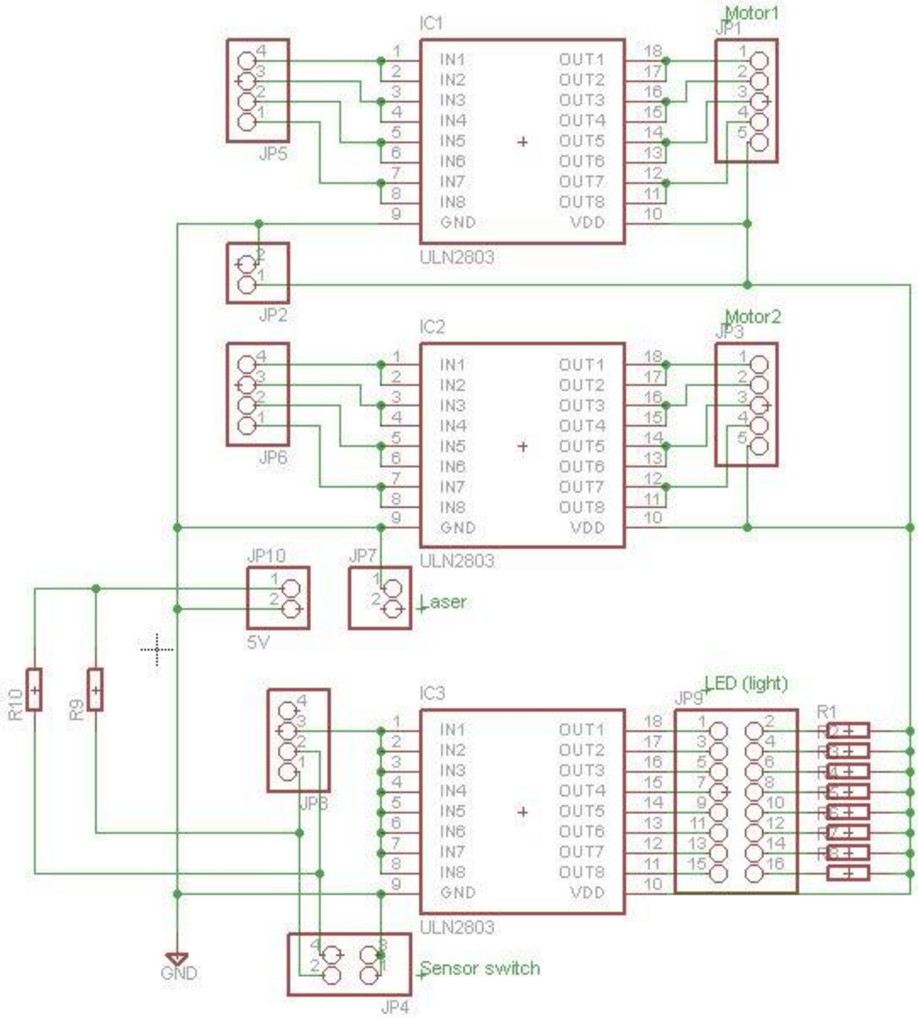
Schematic Design of Board 2 (for LCD and control switches)



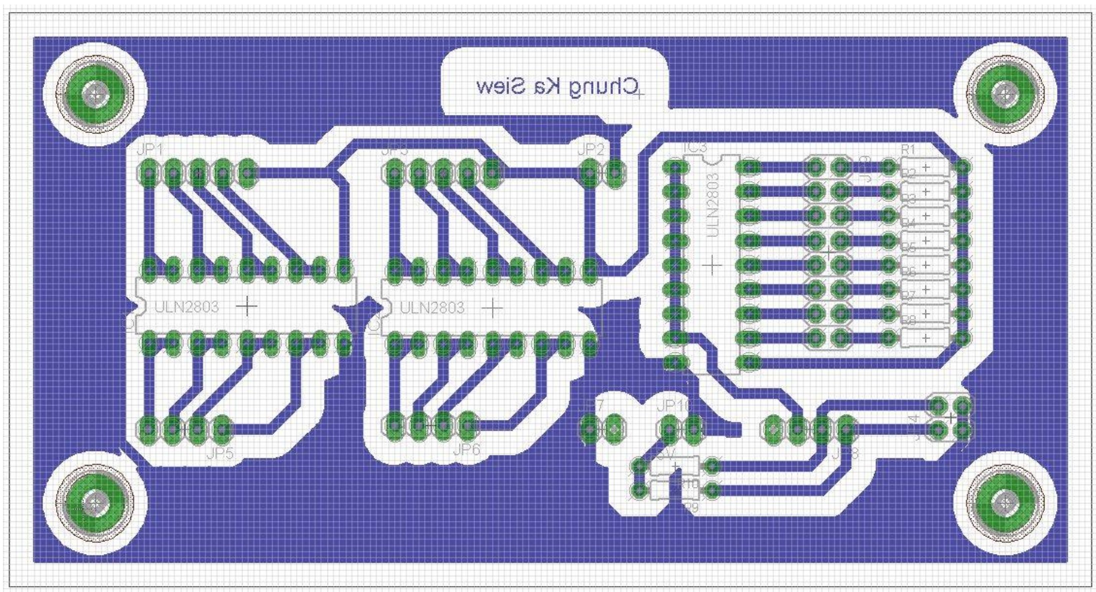
Board Design of Board 2 (for LCD and control switches)



Schematic Design of Board 3 (for other inputs/outputs)

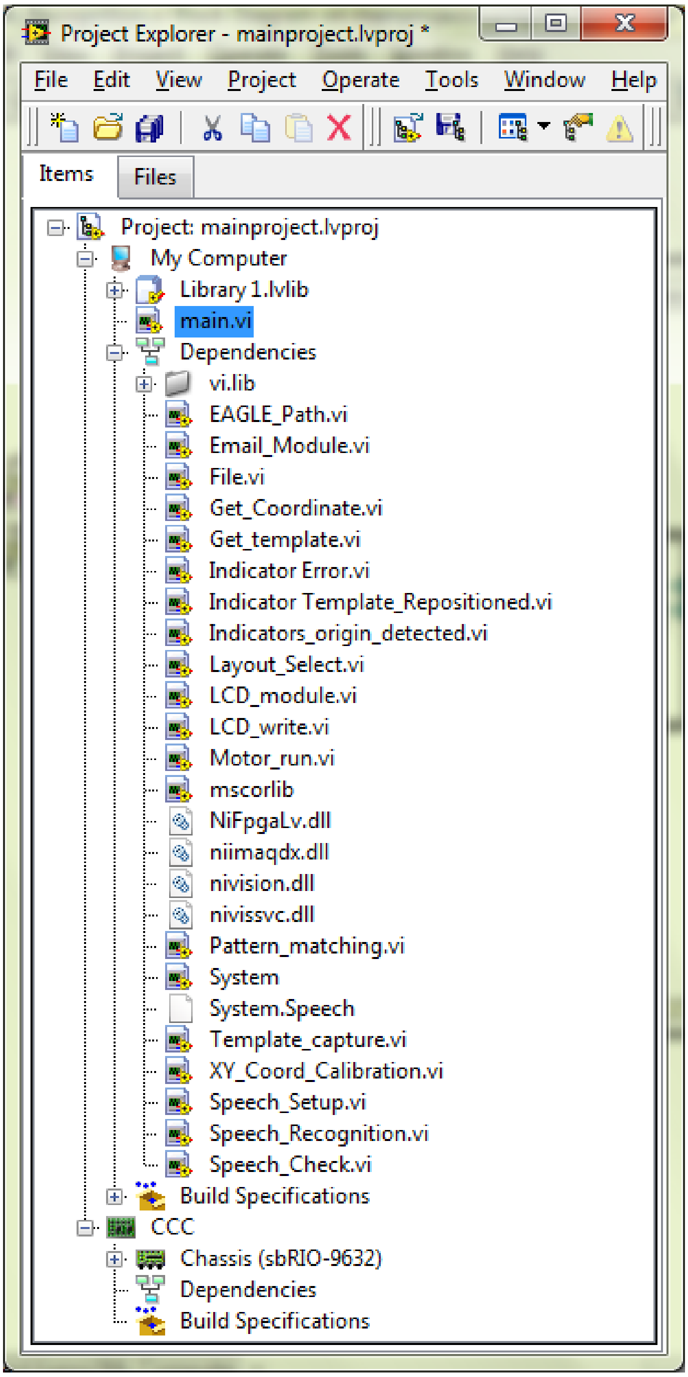


Schematic Design of Board 3 (for other inputs/outputs)

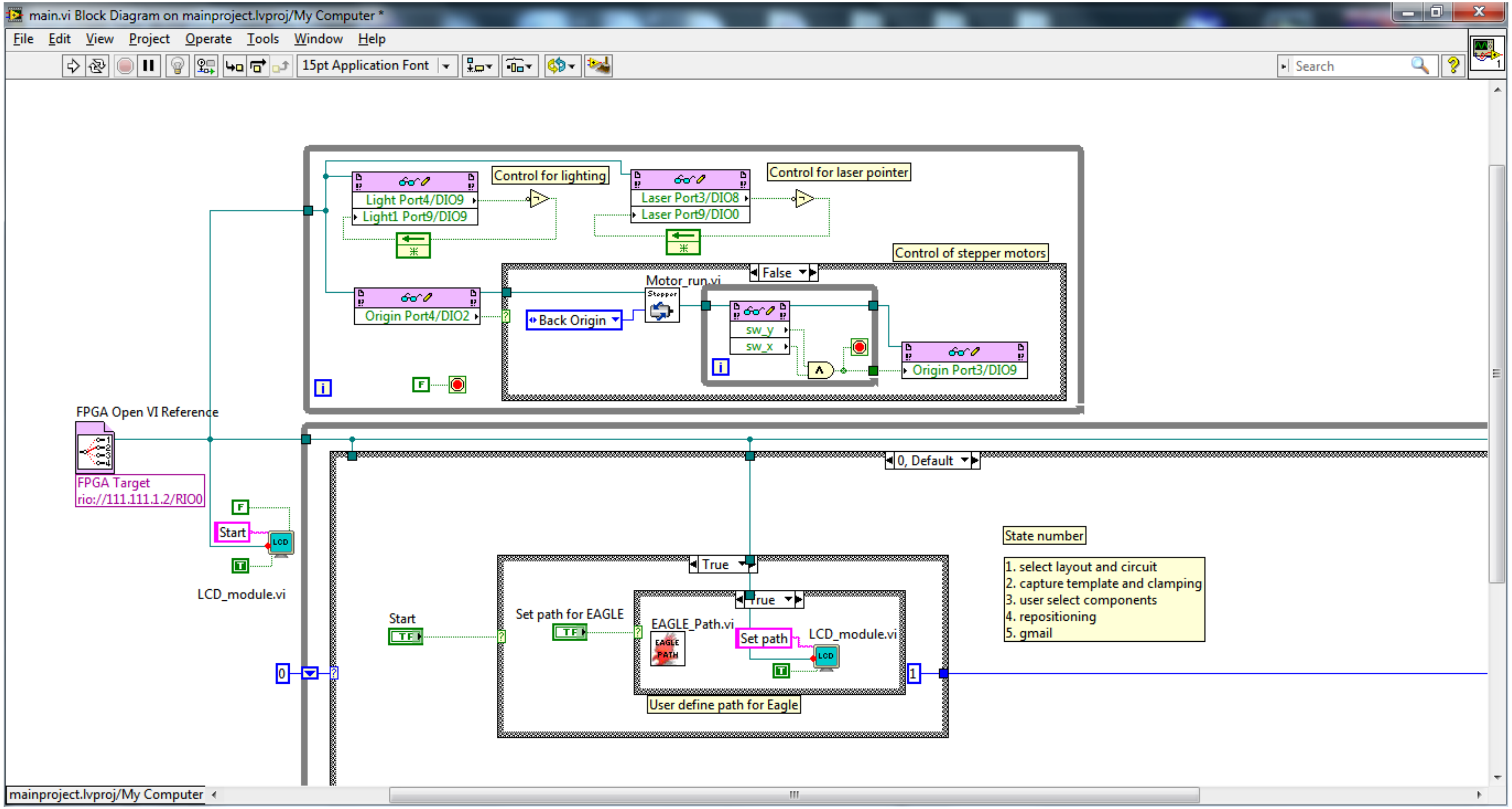


APPENDIX C: LabVIEW Codings

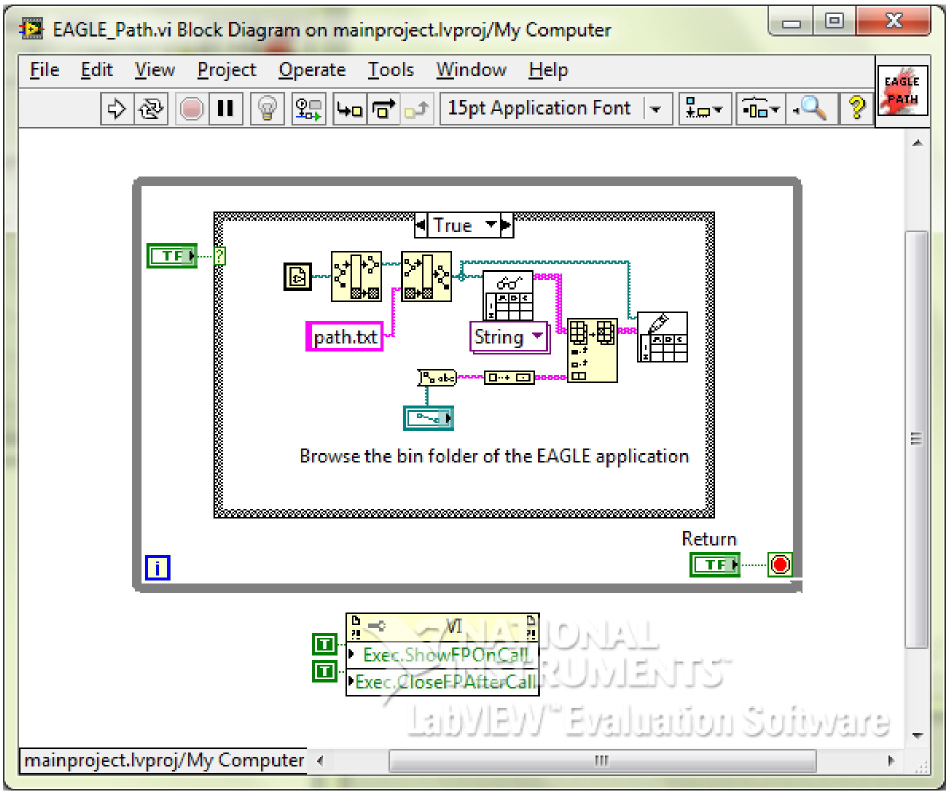
List of VIs



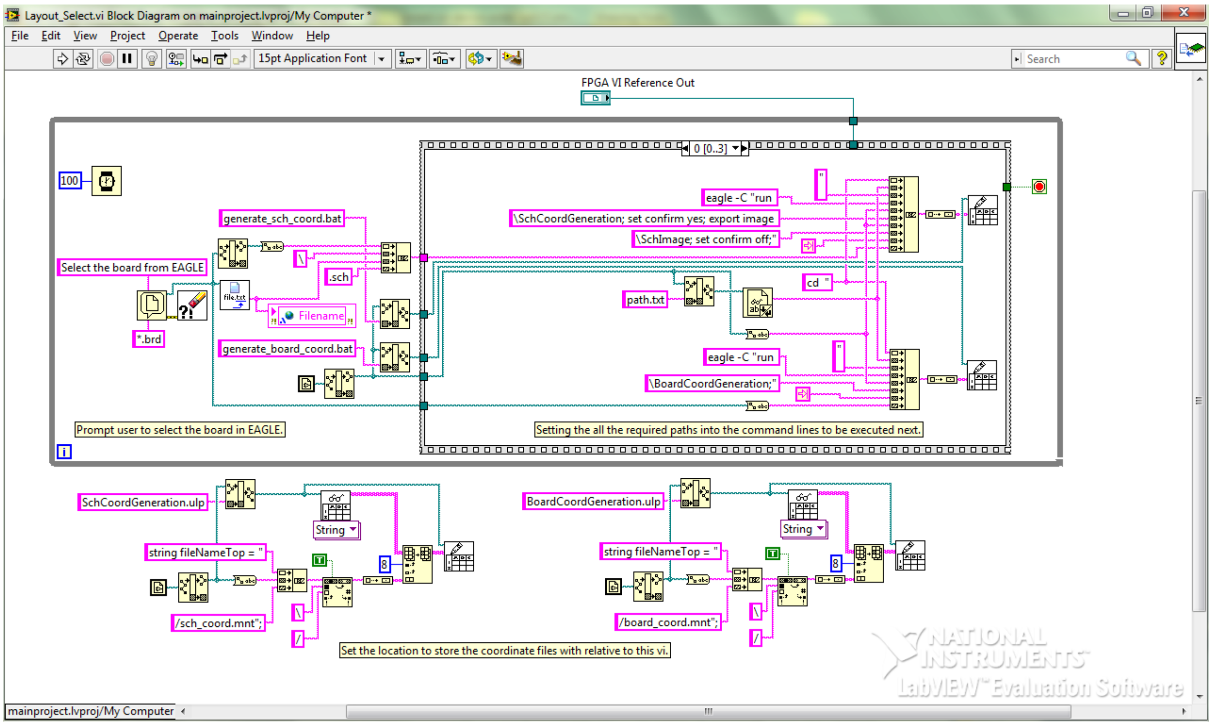
Block Diagram of main.vi



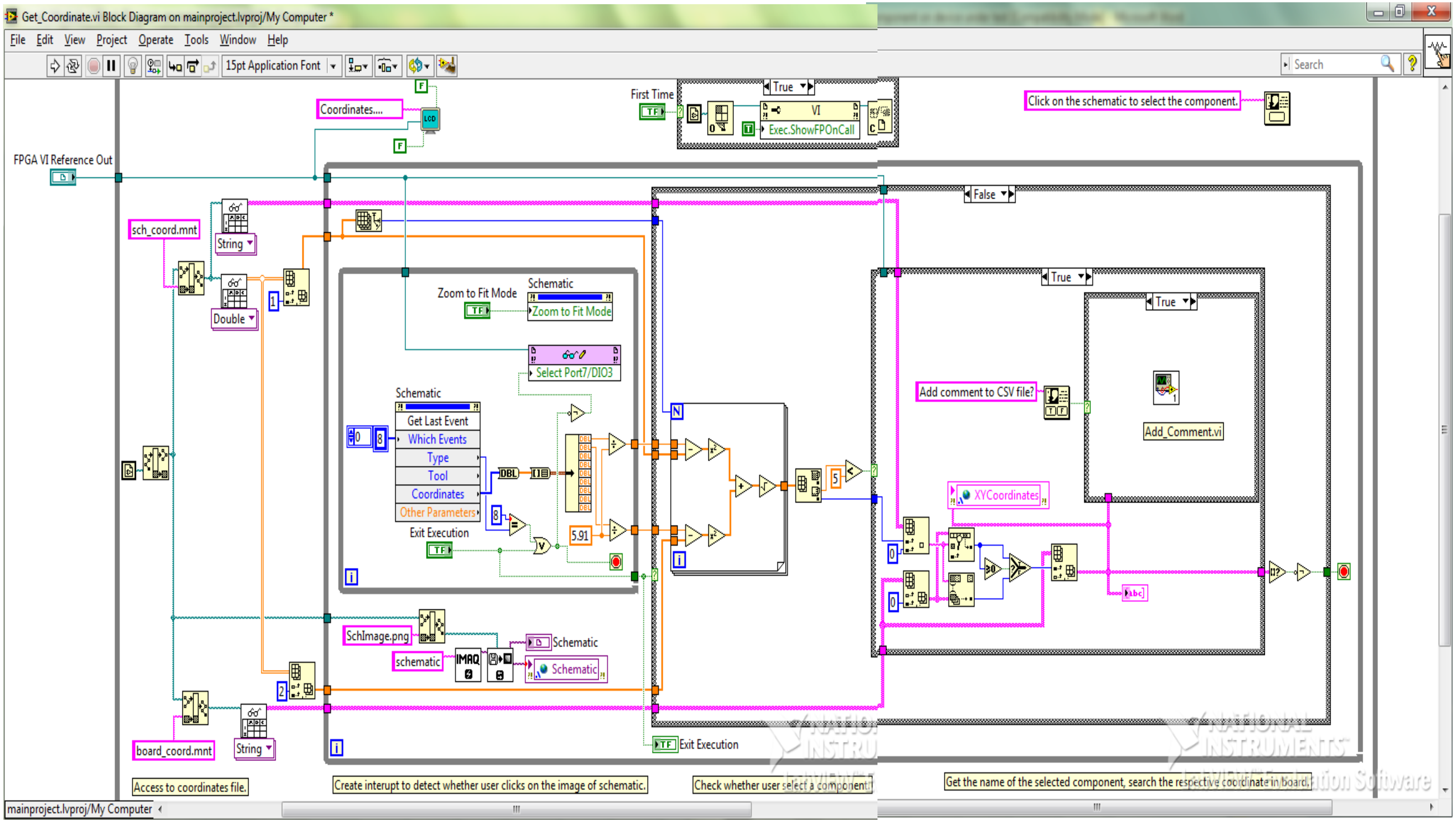
Block Diagram of EAGLE_Path.vi



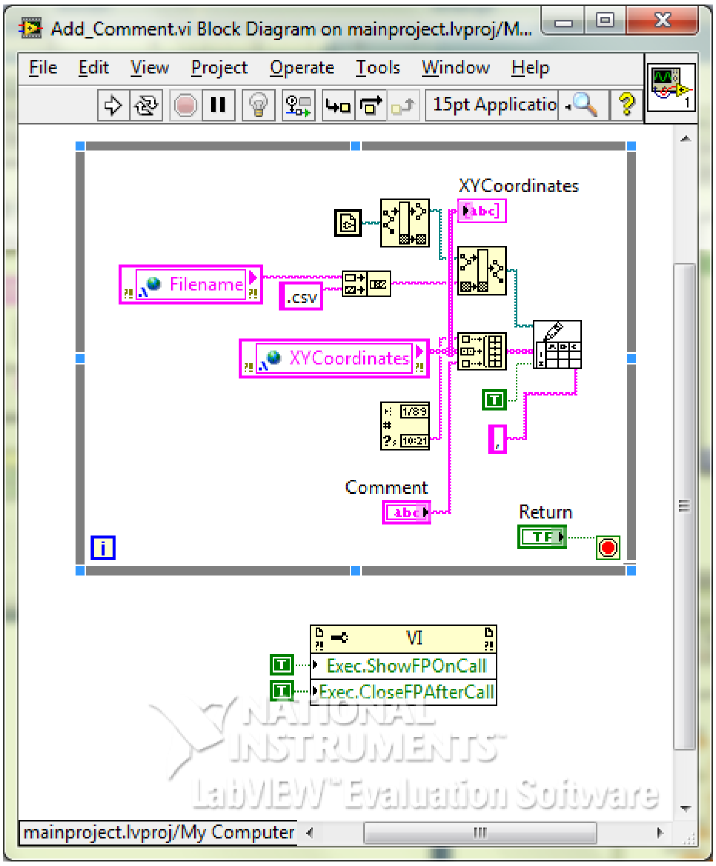
Block Diagram of Layout_Select.vi



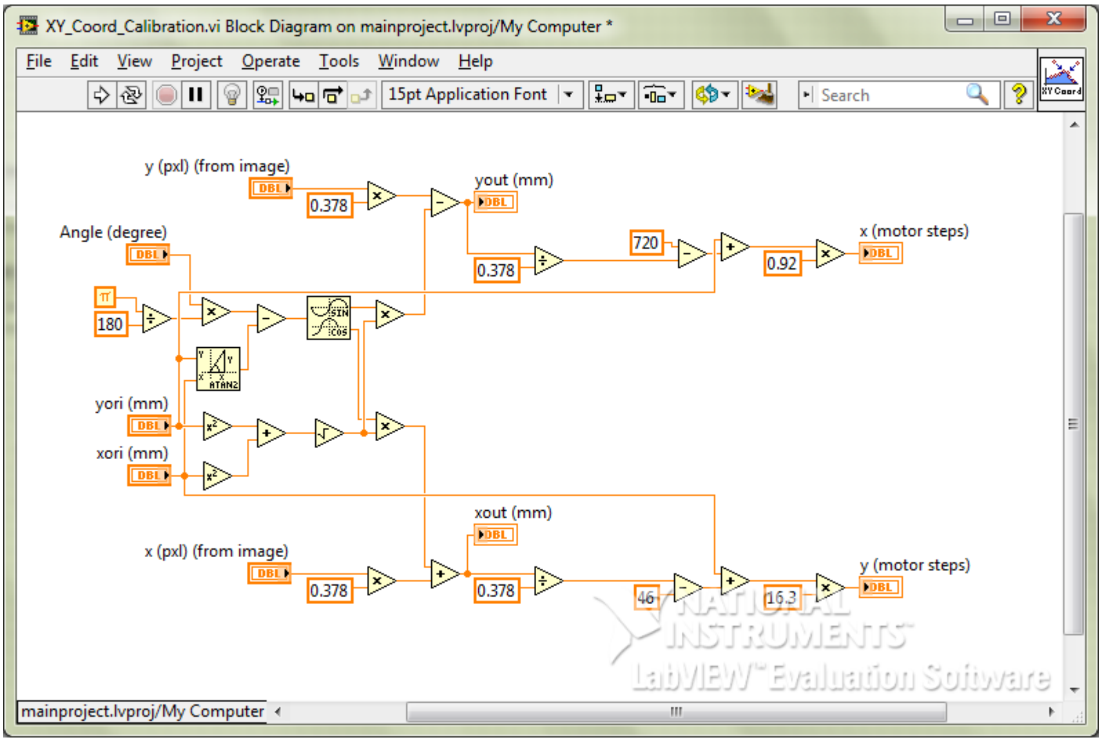
Block Diagram of Get_Coordinate.vi



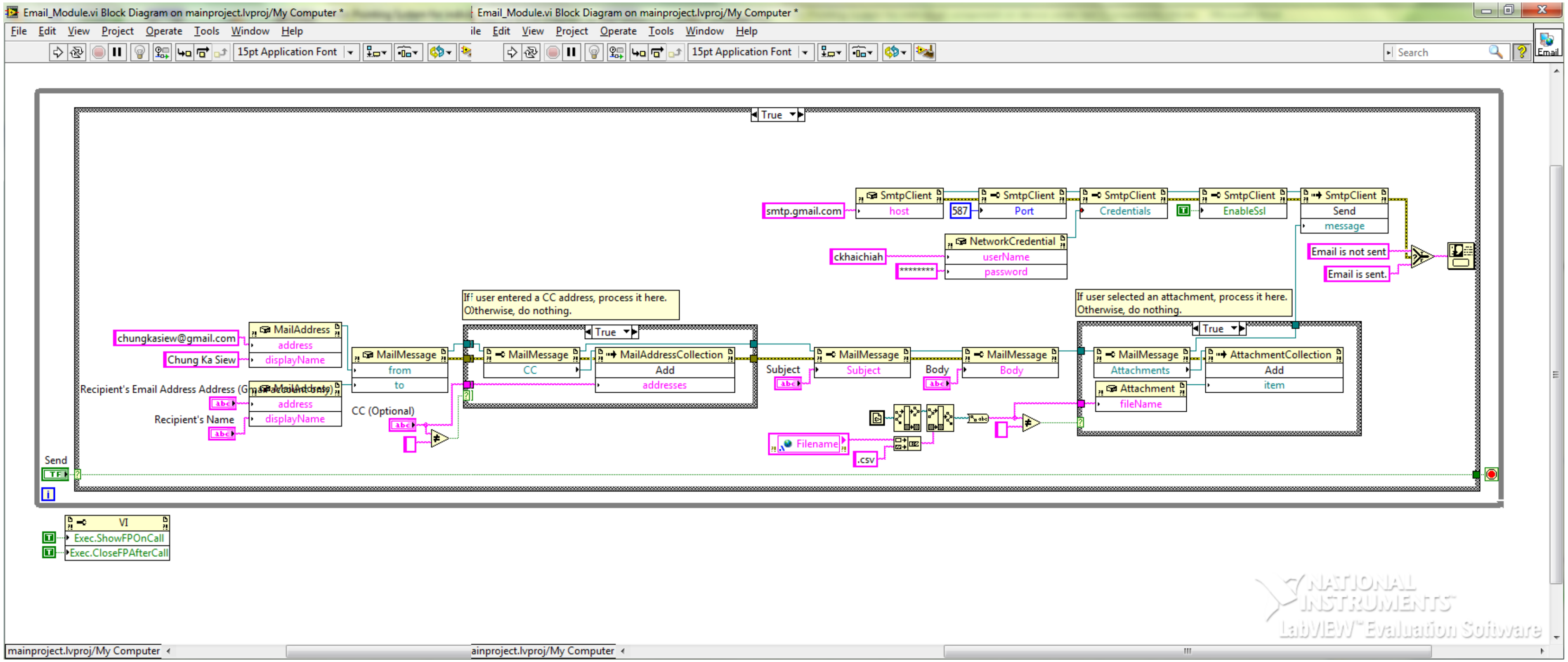
Block Diagram of Add_Comment.vi



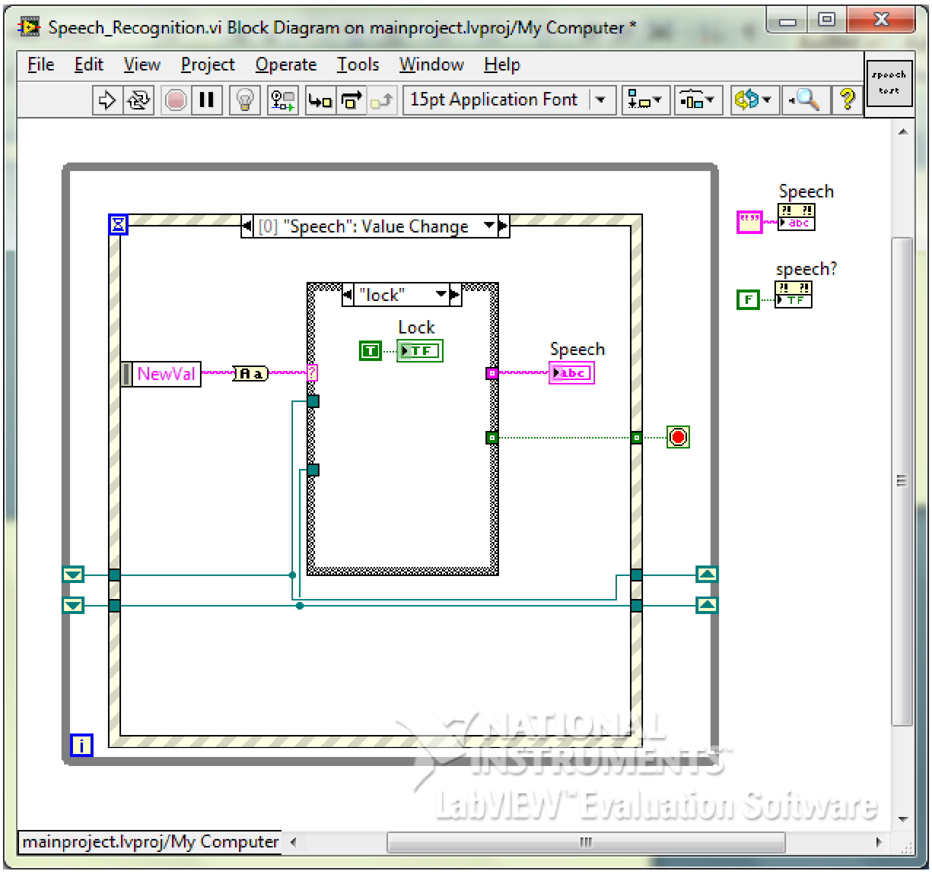
Block Diagram of XY_Coord_Calibration.vi

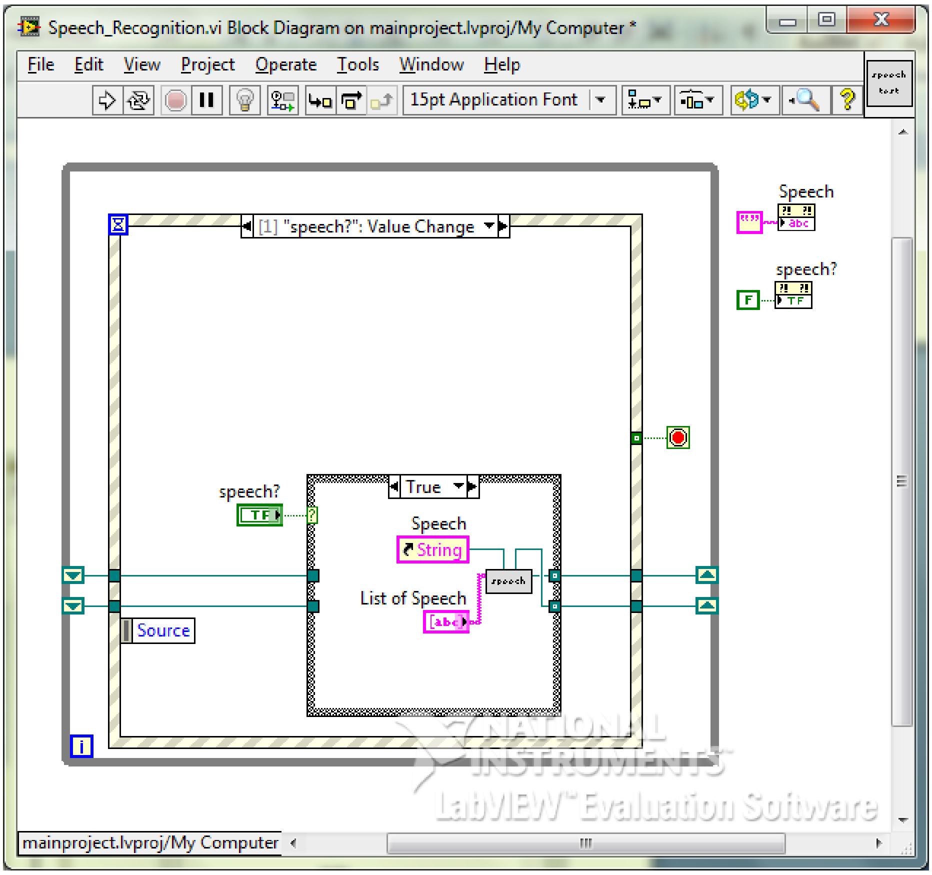


Block Diagram of Email_Module.vi

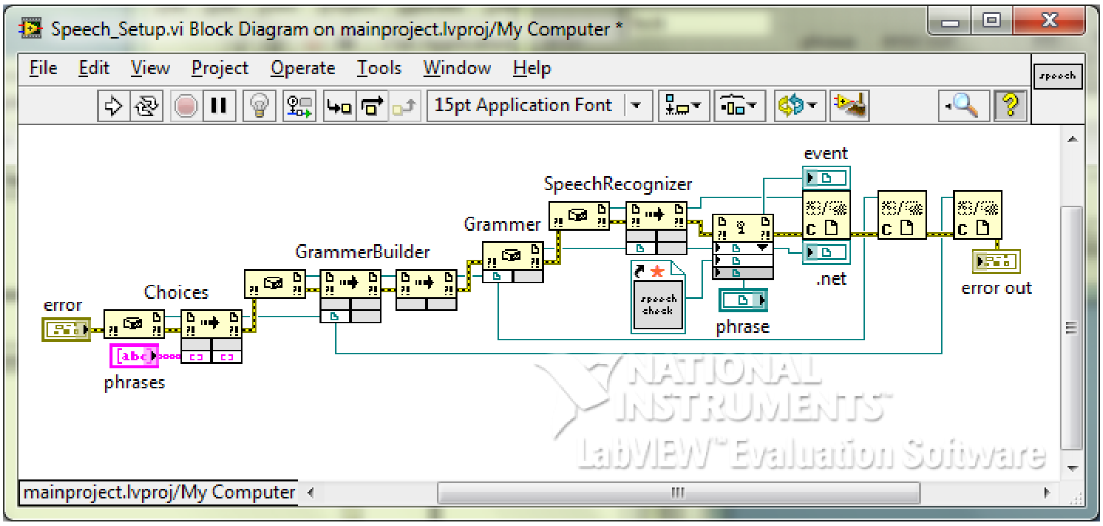


Block Diagram of Speech_Recognition.vi





Block Diagram of Speech_Setup.vi



Block Diagram of Speech_Check.vi