**Lab Experiment**

**Course Title:** Peripherals and Interfacing Lab

**Course Code:** CSE-404

**Name of the Lab-Experiment:** Robots Programming and Interfacing

**Submitted By**

**Tarek Ahammed**

**Registration Number:** 275313

**Batch-Roll Number:** 13

**Semester Number**: 10th

**Batch-Number:** E-36th

**Session:** 2014-2015

**Program-Name:** B.Sc. in CSE



Submitted To

M.M.Musharaf Hussain

Assistant Professor

**Department of Computer Science and Engineering**

**Dhaka International University (DIU)**

**March-2017**

**Name of the experiment:** Robots Programming and Interfacing.

**Rationale / Importance:** Robot is most important peripherals or interfacing. To know about robot programming and interfacing in lab environment with CPU and computer system is vital issues for a student of CSE. This lab experiment we increase skill and experience to develop driver and embedded system in the industrial environment.

**Introduction:**

1. **What is Robotc?**

**ROBOTC** is a cross-robotics-platform programming language for popular educational robotics systems. **ROBOTC** is the premiere robotics programming language for educational robotics and competitions. **ROBOTC** is a C-Based Programming Language with an Easy-to-Use Development Environment.

1. **How is a robot programmed?**

Although offline **programming** has advanced hugely much of the **programming** is done using the teach pendant. This is a handheld device that allows the **robot** to be moved and allow the input of commands and instructions. Movements are sometimes controlled by a joystick (ABB **robots**) or often by buttons on the teach pendant.

1. **What is the best programming language for robotics?**

There has been a huge resurgence of **Python** in recent years especially in robotics. One of the reasons for this is probably that **Python** (and C++) are the two main programming languages found in ROS. Like **Java**, it is an interpretive language. Unlike **Java**, the prime focus of the language is ease of use.

1. **What programming language does vex robotics use?**

**ROBOTC** for VEX Robotics 4.x (VEX **EDR** & VEX IQ) **ROBOTC** for VEX Robotics 4.x by Robomatter Inc. allows users to program their VEX IQ robots using a brand new graphical drag-and-drop programming interface, or the popular industry standard C-based **ROBOTC** programming language – all in the same software!

1. **Industrial Robot Language?**

Almost every robot manufacturer has developed their own proprietary robot programming language, [which has been one of the problems in industrial robotics](http://blog.robotiq.com/bid/34399/5-Reasons-Why-Industrial-Robots-are-Stagnating). You can become familiar with several of them by learning Pascal. However, you are still going to have to learn a new language every time you start using a new robot.

ABB has its [RAPID](https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf) programming language. Kuka has [KRL (Kuka Robot Language)](http://drstienecker.com/tech-332/11-the-kuka-robot-programming-language/). Comau uses [PDL2](ftp://service.bosso.it/Manuali%20COMAU/IT/handbooks/files/lb-0-0-pdl.pdf), Yaskawa uses [INFORM](http://spaz.org/~jake/robot/155493-INFORM-LANGUAGE.pdf) and Kawasaki uses [AS](http://www.scribd.com/doc/92216565/D-Controller-As-Language-Reference-Manual-deb). Then, Fanuc robots use [Karel](http://onerobotics.com/2013/12/28/introduction-to-karel-programming.html), Stäubli robots use [VAL3](http://www.staubli.com/en/robotics/robot-software/val3-robot-programming/val3-language/) and Universal Robots use [URScript](http://www.wmv-robotics.de/home_htm_files/scriptmanual_en_1.5.pdf).

In recent years, programming options like [ROS Industrial](http://blog.robotiq.com/bid/70617/Why-use-ROS-Industrial-for-Industrial-Robot-Applications) have started to provide more standardized options for programmers. However, if you are a technician, you are still more likely to have to use the manufacturer's language.

1. **How can we make a robot?**

Part 5 Programming the Robot\*\*Start the Arduino IDE by double-clicking the arduino.exe file in the IDE folder. You'll be greeted with a blank project.\*\*Paste the following code to make your robot go straight. ... \*\*Build and upload the program. ... \*\*Add the kill switch functionality. ... \*\*Upload and test your code.

## What Language to Choose?

There are many programming languages which can be used to program microcontrollers, the most common of which are:

* **Assembly**; its just one step away from machine code and as such it is very tedious to use. Assembly should only be used when you need absolute instruction-level control of your code.
* **Basic;** one of the first widely used programming languages, it is still used by some microcontrollers ([Basic Micro](http://www.robotshop.com/basic-atom-microcontroller.html), [BasicX](http://www.robotshop.com/basicx-microcontroller.html), [Parallax](http://www.robotshop.com/basic-stamp-microcontrollers.html)) for educational robots.
* **C**/**C++;** one of the most popular languages, C provides high-level functionality while keeping a good low-level control.
* **Java**; it is more modern than C and provides lots of safety features to the detriment of low-level control. Some manufacturers like [Parallax](http://www.robotshop.com/javelin-stamp-microcontroller.html) make microcontrollers specifically for use with Java.
* **.NET**/**C#**; Microsoft’s proprietary language used to develop applications in Visual Studio. Examples include [Netduino](http://www.robotshop.com/productinfo.aspx?pc=RB-Sec-01&lang=en-US), [FEZ Rhino](http://www.robotshop.com/productinfo.aspx?pc=RB-Ghi-09&lang=en-US) and [others](http://www.robotshop.com/productinfo.aspx?pc=RB-Lin-36&lang=en-US)).
* **Processing** ([Arduino](http://www.robotshop.com/arduino-2.html)); a variant of C++ that includes some simplifications in order to make the programming for easier.
* **Python**, one of the most popular scripting languages. It is very simple to learn and can be used to put programs together very fast and efficiently.

If you have chosen a hobbyist microcontroller from a known or popular [manufacturer](http://www.robotshop.com/suppliers.html), there is likely a large book available so you can learn to program in their chosen programming language. If you instead chose a microcontroller from a smaller, lesser known manufacturer (e.g. since it had many features which you thought would be useful for your project), it’s important to see what language the controller is intended to be programmed in (C in many cases) and what development tools are there available (usually from the chip manufacturer).

## Useful Tips

1. Create manageable chunks of functional code: By creating segments of code specific to each product, you gradually build up a library. Develop a file system on your computer to easily look up the necessary code.
2. Document everything within the code using comments: Documenting everything is necessary in almost all jobs, especially robotics. As you become more and more advanced, you may add comments to general sections of code, though as you start, you should add a comment to (almost) every line.
3. Save different versions of the code – do not always overwrite the same file: if you find one day that your 200+ lines of code do not compile, you won’t be stuck going through it line by line; instead you can revert to a previously saved (and functional) version and add / modify it as needed. Code does not take up much space o a hard drive, so you should not feel pressured to only save a few copies.
4. Raise the robot off the table or floor when debugging (so its wheels/[legs](http://www.robotshop.com/hexapod-development-platforms-1.html)/[tracks](http://www.robotshop.com/tracked-development-platforms-1.html) don’t accidentally launch it off the edge), and have the power switch close by in case the robot tries to destroy itself. An example of this is if you try to send a servo motor to a 400us signal when it only accepts a 500 (corresponding to 0 degrees) to 2500us (corresponding to 180 degrees) signal. The servo would try to move to a location which it cannot physically go to (-9 degrees) and ultimately burn out.
5. If code does something that does not seem to be working correctly after a few seconds, turn off the power – it’s highly unlikely the problem will “fix itself” and in the meantime, you may be destroying part of the mechanics.
6. Subroutines may be a bit difficult to understand at first, but they greatly simplify your code. If a segment of code is repeated many times within the code, it is a good candidate to be replaced with a subroutine.

## Practical Example

We have chosen an Arduino microcontroller to be the “brain” of our robot. To get started, we can take a look at the [Arduino 5 Minute Tutorials](http://www.robotshop.com/blog/en/?p=3640). These tutorials will help you use and understand the basic functionality of the Arduino programming language. Once you have finished these tutorials, take a look at the example below.



For the robot we have made, we will create code to have it move around (left, right, forward, reverse), move the two servos (pan/tilt) and communicate with the distance sensor. We chose Arduino because of the large user community, abundance of sample code and ease of integration with other products.

**Objective:**

* To know about robot in lab environment.
* To know about robot Programming in lab environment.
* To know about robot interfacing in lab environment.
* To develop skill about the robot driver development in lab environment.
* To earned experience about robot interfacing in lab environment.
* To apply experience and skill about robot in industrial environment.

**Required Hardware & Software:**

* Computer System
* Operation System.
* Emu 8086
* Virtual Peripherals and Device.

**Working Procedure:**

* Power on at the computer system.
* Install the emu 8086.
* Install required or dependency programming language.
* Open the emu software.
* Create new project.

**Implementation Code:**

#start=robot.exe#

name "robot"

#make\_bin#

#cs = 500#

#ds = 500#

#ss = 500# ; stack

#sp = ffff#

#ip = 0#

; this is an example of contoling the robot.

; this code randomly moves the robot,

; and makes it to switch the lamps on and off.

; robot is a mechanical creature and it takes

; some time for it to complete a task.

; status register is used to see if robot is busy or not.

; c:\emu8086\devices\robot.exe uses ports 9, 10 and 11

; source code of the robot and other devices is in:

; c:\emu8086\devices\developer\sources\

; robot is programmed in visual basic 6.0

; robot base i/o port:

r\_port equ 9

;===================================

eternal\_loop:

; wait until robot

; is ready:

call wait\_robot

; examine the area

; in front of the robot:

mov al, 4

out r\_port, al

call wait\_exam

; get result from

; data register:

in al, r\_port + 1

; nothing found?

cmp al, 0

je cont ; - yes, so continue.

; wall?

cmp al, 255

je cont ; - yes, so continue.

; switched-on lamp?

cmp al, 7

jne lamp\_off ; - no, so skip.

; - yes, so switch it off,

; and turn:

call switch\_off\_lamp

jmp cont ; continue

lamp\_off: nop

; if gets here, then we have

; switched-off lamp, because

; all other situations checked

; already:

call switch\_on\_lamp

cont:

call random\_turn

call wait\_robot

; try to step forward:

mov al, 1

out r\_port, al

call wait\_robot

; try to step forward again:

mov al, 1

out r\_port, al

jmp eternal\_loop ; go again!

;===================================

; this procedure does not

; return until robot is ready

; to receive next command:

wait\_robot proc

; check if robot busy:

busy: in al, r\_port+2

test al, 00000010b

jnz busy ; busy, so wait.

ret

wait\_robot endp

;===================================

; this procedure does not

; return until robot completes

; the examination:

wait\_exam proc

; check if has new data:

busy2: in al, r\_port+2

test al, 00000001b

jz busy2 ; no new data, so wait.

ret

wait\_exam endp

;===================================

; switch off the lamp:

switch\_off\_lamp proc

mov al, 6

out r\_port, al

ret

switch\_off\_lamp endp

;===================================

; switch on the lamp:

switch\_on\_lamp proc

mov al, 5

out r\_port, al

ret

switch\_on\_lamp endp

;===================================

; generates a random turn using

; system timer:

random\_turn proc

; get number of clock

; ticks since midnight

; in cx:dx

mov ah, 0

int 1ah

; randomize using xor:

xor dh, dl

xor ch, cl

xor ch, dh

test ch, 2

jz no\_turn

test ch, 1

jnz turn\_right

; turn left:

mov al, 2

out r\_port, al

; exit from procedure:

ret

turn\_right:

mov al, 3

out r\_port, al

no\_turn:

ret

random\_turn endp

;===================================

**Input for testing: N/A**

**Output Analysis:**

### C:\Users\DIU\Desktop\Capture.PNG

### Conclusion:

* I have gathered knowledge know about robot in lab environment.
* I have acquired knowledge about robot Programming in lab environment.
* I have earned knowledge about robot interfacing in lab environment.
* I have gathered knowledge develop skill about the robot driver development in lab environment.
* I have earned experience about robot interfacing in lab environment.
* I have applied experience and skill about robot in industrial environment.