Turtle Robot On The Grid

By

Dadje Tchagop Nismin Valere
15686

Dissertation submitted to the department of Electrical and Electronic Engineering in partial fulfillment of the requirement for the Bachelor of Engineering (Hons) (Electrical and Electronic Engineering)

SEPTEMBER 2014

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATE OF APPROVAL

Turtle Robot On The Grid

By

Dadje Tchagop Nismin Valere

15686

A project dissertation submitted to the

Department of Electrical and Electronic Engineering

Universiti Teknologi PETRONAS

In partial fulfillment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL AND ELECTRONIC)

Approved by:

Abu Bakar Sayuti Hj Mohd Saman

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

September 2014

CERTIFICATION OF ORIGINALITY

This is to certified that I am responsible for the work submitted in this project, that the original work is my own accept as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

DADJE TCHAGOP NISMIN VALERE

Abstract

The combination of turtle graphics and physical robot has helped in introducing programming to kids with the aim of enhancing their comprehension and problem solving skills in mathematics and computer science. First developed in 1966, by Seymour Papert, turtle graphic was initially used to introduce Logo programming. More work has been done in that sense, where the user can virtually interact with the robot (here represented by an object on the screen) by entering the command on the screen and the robot performs the specific task. Our current work enables us to physically interact with the robot by sending the code wirelessly from a computer to the robot. The limitation with this work is that during the communication with the physical robot, the command can only be sent one line at the time (no possibility of sending multiple lines). Also, no sensor was used to give orientation to the robot. The main focus on this project will be to set the robot and the platform to be able to send and receive multiple line of code at the time (set of command can be executed at the time). Also the project will focus on how to use sensors to facilitate the movement and orientation of the robot.

AKNOWLEDGEMENT

First of all I am grateful to the almighty God for giving me the strength and wisdom during the completion of this project.

I wish to give a special thank to my family for their support throughout my studies.

I wish to express my sincere gratitude to my project supervisor, Abu Bakar Sayuti Bin Hj Mohd Saman for always being available to support and advice.

I place on record, my sincere gratitude to my lecturers, all the staffs who have so much available to help, guide and advice me during my studies.

To all my classmates, friends for the great time we spend together, it has been a great source of inspiration for me.

My sense of gratitude to all who, directly or indirectly, have lent their helpful hand on this venture.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

1.0 INTRODUCTION

1.1 Background Studies

Turtle graphics programming is a powerful vehicle for introducing children to computer programming. Children combine simple graphics commands to get a display screen cursor called a turtle, to draw designs on the display screen [1]

First developed in 1966 by the two pioneers Wally Feurzeig[1] and Seymour Papert[2], turtle graphic will very soon been recognized as an easy and popular way of introducing Logo Programming [3] to children or teenagers. In order to support his version of turtle robot, Seymour Papert added support for turtle graphic to Logo. The idea was to have a simple robot which is controlled by a user from the user's workstation and is able to carry out specific tasks (design or drawing) using a pen attached to its body.

Conventionally, the robot is virtually represented as an object using computer program and the user or programmer can just write a set of command on the screen in his work station to control the movement of the object (robot). There are several ways of communicating (programming) with the robot including wired communication (data is sent from terminal to robot via cable) and wireless communication (data is sent using air).

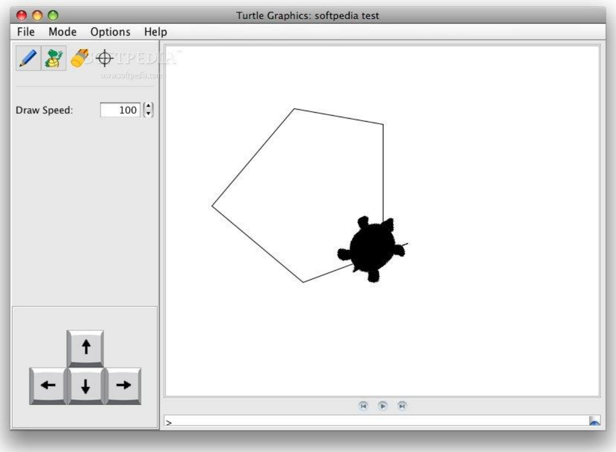


Figure 1.1: Turtle graphics [4]

Seymour Papert once said: “The role of the teacher is to create condition for inventions rather than providing ready-made knowledge”. That statement suggests that teachers should always bring learners to question themselves in order to give the best of them. Seymour Papert is recognized as one of the pioneer of Logo and Turtle graphic. Born in February 1928, he dedicated his life on searching how to improve children comprehension and sense of analogy using mathematics and science.

Developed mostly by Seymour Papert, Turtle graphics is an expression used in computer graphics for a method of programming vector graphic using a relative cursor upon a Cartesian plane; it is a software program. The cursor displayed on the screen represents the “turtle” and using the code, instructions or orientation are sent to the turtle. To move to turtle from one position to another or to design a specific pattern, instructions must be sent to the turtle (robot) using set of commands such as FORWARD, BACKWARD, LEFT or RIGHT following the distance or angle that the robot should move. By its simplicity and efficiency, the program attracts kids and by then introduces them to programming and computer science.

This text reports on parts of the ongoing work on "Over-the-air Programming of Tutle Robot" at the Electrical and Electronic Engineering Department of UTP. Specifically this report describes the project of enhancement of data formatting and transfer over wireless communication between a programming terminal and a turtle robot.

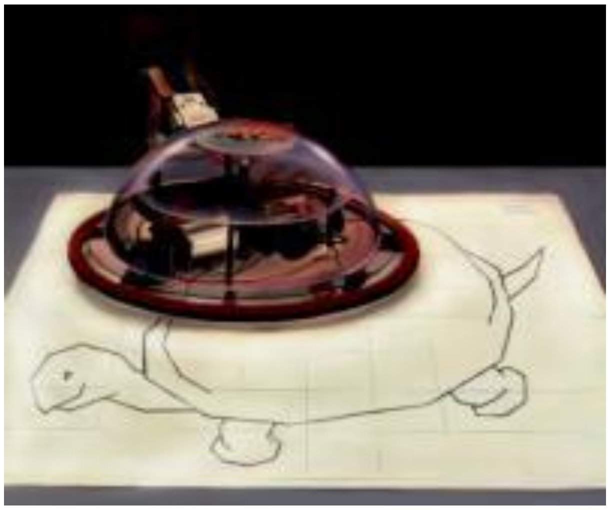


Figure 1.2: Turtle robot [5]

1.2 Problem Statement

Turtle graphic is a subset of the Logo programming language. It is a computer graphic system which allows the user to communicate with the robot through command instructions. Research have been conducted on both turtle graphic software which uses Logo programming and a physical turtle robot. Nowadays, different types of communication are been established between the computer and the robot including wired communication (using cable) or wireless communication (over the air). Previous work has allowed commands to be sent to the robot from a remote position, but the instruction could only be sent one line at the time. Enhancing this communication protocol will allow the robot to receive and execute multiple line statements at a time and therefore give more flexibility to the robot. Also previous work does not take into consideration the positioning and orientation of the robot. While moving, the robot cannot accurately evaluate the total displacement (distance covered) or does not follow a given direction accurately. An approach on line following technique (for orientation) or compass will help to overcome those limitations.

1.3 Objective

The main objective of this project is to enhance the wireless programming feature of the current turtle robot so that a complete logo programming consisting of multiple lines of instructions can be sent at once.

For example, instead of:

FORWARD 10 <send>
LEFT 90 <send>
FORWARD 20 <send>

The aim is to send all three instructions above as:

FORWARD 10
LEFT 90
FORWARD 20
<send>

The secondary objective is to provide the robot with more features to overcome the limitations in positioning and orientation. The robot will be equipped with Infrared sensors. Used for line following techniques and (or) junction count, those sensors will help in accurately moving the robot. The robot must be able to accurately follow a specific direction without deflection.

1.4 Scope of study

The focus points of this research are:

- Understand turtle graphic programming

  A better understanding of the turtle graphic software will help enhancing our approach in getting better results.
- Understand python programming language

  Python programming language is well spread programming tool, it enables us to easily communicate with the robot or any other hardware.
- Use sensors and line following technique to direct the robot.

  Line following technique is popular in autonomous vehicles, understanding how sensors work is crucial in obtaining an optimum result from their usage. Sensors enable the robots to sense their environment and send back the result which is then use to control the robot.

CHAPTER II

2.0 LITERATURE REVIEW

2.1 Evolution of Turtle Graphic

Going back in the years 80s, computers were still in the early stage. No one could at that time imagine that computer could play a bigger role in assisting humans in their daily activities. The increase usage of computer has been extremely remarkable over the past two decades, beating any possible forecast. Computer and computer science has always been consider difficult to comprehend and had to use, late in the years 70s, some researchers started working on ways to introduce computer to kids to help them accommodate with what will soon became indispensable for human being. In 1966, thanks to the hard work of Seymour Papert, Turtle graphic was born. Papert foresees the importance of computer for the future generation and to help them easily accommodate with it, he came out with turtle graphic. The turtle graphic at that time was a simple computer system which allows kids to interact with a virtual robot by sending command codes.

The concept of turtle graphic has since then been enhance and a new concept been introduced. In 1991, John A. Fulcher conducted a research project on the turtle robot using a Motorola 68KECB, the project will have some limitations because the robot is attached to the computer by cable and the movement is reduced [6].

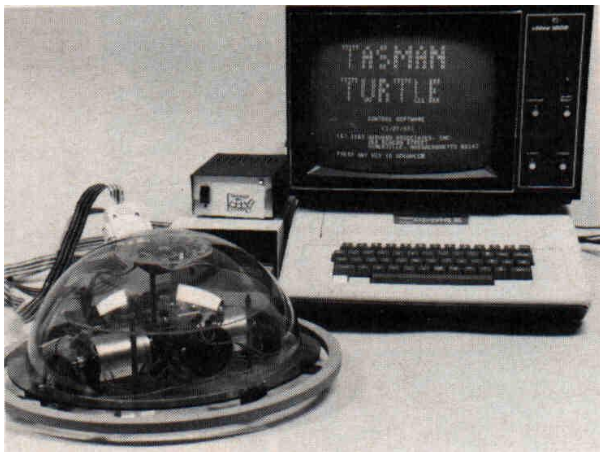


Figure 2.1: Tasman turtle [7]

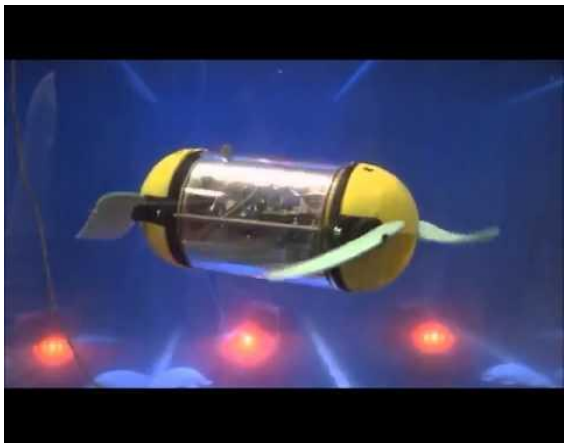


Figure 2.2: Underwater turtle robot [8]

2.2 Implementation of Turtle Graphic in the Industry

In the years 2000s, turtle robot will begin to expand into industrial applications. Using wireless data transmission, many projects involving autonomous and semi-autonomous robots will be developed. Researches will induce the development amphibious robots with the capability of carrying underwater monitoring and recovery operations (AUV). Robots will be used to access removed and hazardous areas. The semi-autonomous underwater vehicle SAUVIM was design base on the turtle graphic.

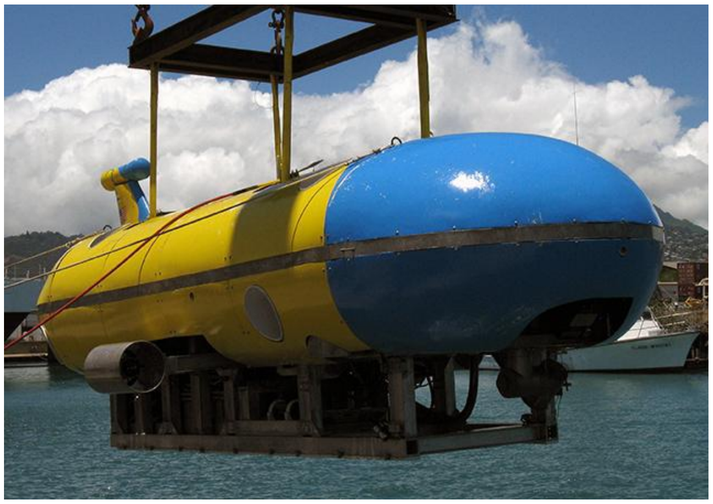


Figure 2.3: SAUVIM Robot [9]

CHAPTER III

3.0 METHODOLOGY

3.1 Hardware-Software co-design

Hardware and software co-design represent the layout of the sequences in which the data are being written, interpreted, compiled and executed. The hardware consists of computer hardware (PC hardware) and the robot hardware (PIC microcontroller, servo motors and LCD screen). The software has the turtle graphic on the PC and Logo interpreter on the robot. Data are sent wirelessly (over the air). Using the user interface in the turtle graphic (Python environment), the user enters the set of command to be executed; the command is translated and sent to the robot via Radio Frequency (RF) transmitter. The data received by the robot are interpreted by the logo interpreter and executed by the microcontroller. Sensors are used to sense the environment and ensure the operation runs smoothly. Servo motor and LCD screen will perform the instruction from the microcontroller. The diagram below gives the flow chart of the system.

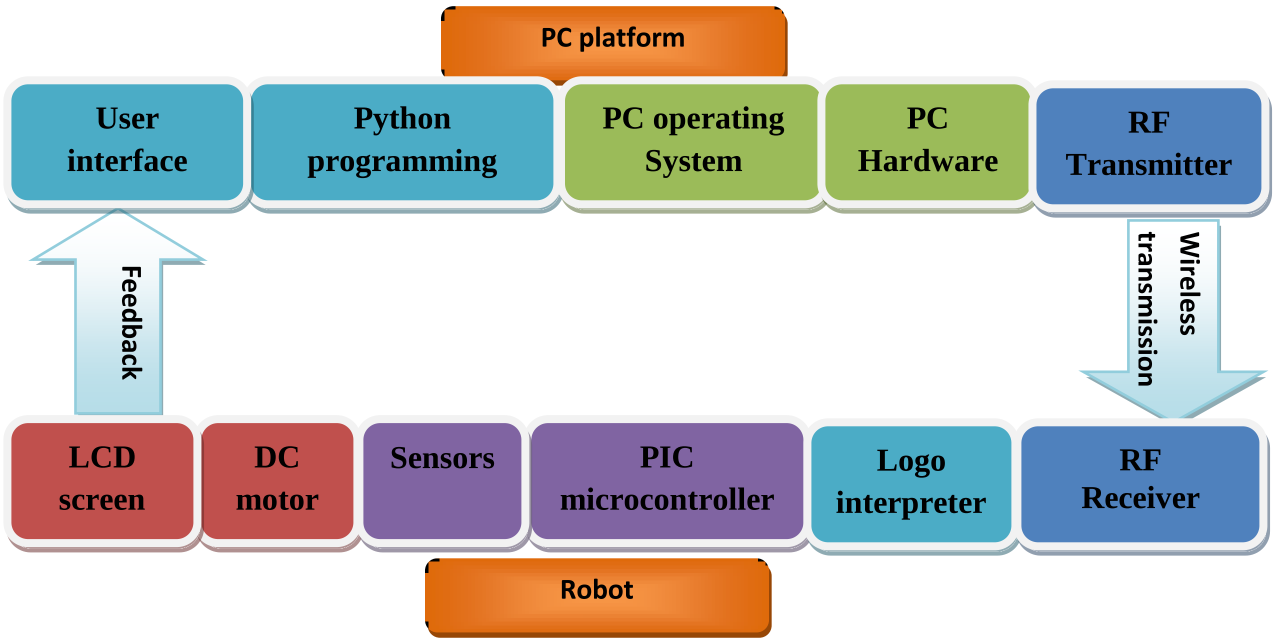


Figure 3.1: hardware/software design

3.2 hardware- Software integration

Integrating software and hardware is a key element of the project. By integrating both software and hardware, we increase the possibility of interaction between both. It is part of the system design requirement. The idea is to keep hardware and software unified all the time as much as possible. The system integration is mainly consists of four phases: System Requirement, Software & Hardware Design, System integration and Testing. [10]

System Requirement: here we specify the elements needed to develop the prototype. The procedure chosen is base on constraint of the availability of the materials and the time. Using the finite state machine (FSM), we modeled hardware and software.

Software and Hardware Design: here we design software and hardware separately. The design of the software includes designing the interface using python programming language, developing the transmission protocol. The hardware design includes the design and fabrication of the robot's structure, the design of the circuit and set the hardware. The choice of hardware such as controller and Radio Frequency module must take in consideration the time, cost and availability of the materials. After the design, the software and hardware are tested separately.

System integration: hardware, software and interface implementation are synthesized using FSM. The system is integrated and tested for compatibility.

Co-simulation: all the components are executed; they all function together at real time.

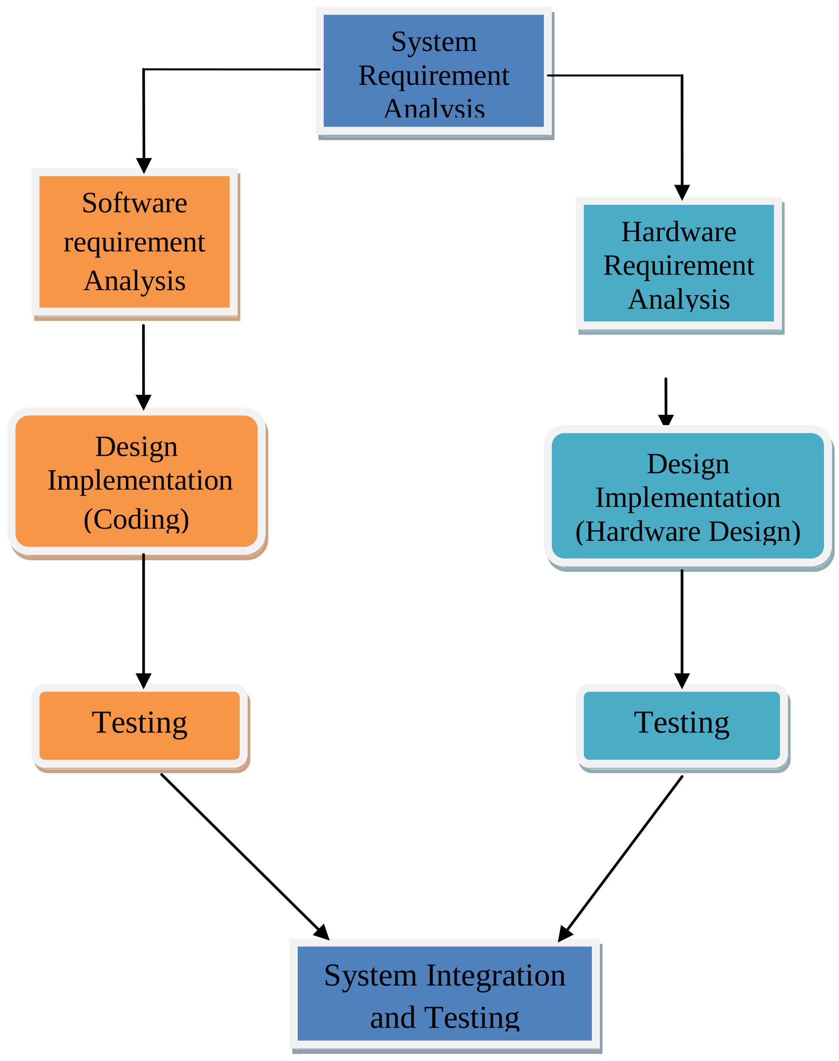

System Requirement Analysis
Software requirement Analysis
Hardware Requirement Analysis
Design Implementation (Coding)
Design Implementation (Hardware Design)
Testing
Testing
System Integration and Testing

Figure 3.2: system design and integration

3.3 Radio Frequency (RF) communication protocol

The development of factory automation in the past decades has contributed on the increase of research in the areas such as communication, automation, computer, sensors and control. Focusing specially on communication, we identify Bluetooth, Ultra Wideband (UWB), ZigBee, Radio Frequency (RF) and WiFi as five standards for short range communication. Those communication standards have the advantage of being low power consumption. In term of application, Bluetooth is design for cordless mouse, keyboard and hands-free headset. UWB is toward high-bandwidth multimedia links, ZigBee is design for wireless network monitoring and diagnosis, WiFi is mostly for computer to computer connection and Radio Frequency (RF) signal refers to wireless broadcasting. In this case, the signal is created by alternating AC current having the characteristic such as if the current is an input to an antenna electromagnetic (EM) is created, suitable for wireless communication.

For the purpose of this project, the better understanding of the Radio Frequency communication protocol will enable us to efficiently communicate with the robot.

Wireless communication technology

- RF is an alternating current which, when an antenna is supplied with an alternating current, an electromagnetic field is generated and propagates through space.
- Cheap and widely used

  Nowadays, over 40 millions systems manufactured each year utilizing low-power wireless (RF) technology for data links, telemetry, control and security.
- Wide range of applications

  Cordless and cellular telephones, radio and television broadcast stations, hand-held computer and PDA data links, wireless bar-code readers, wireless keyboards for PCs, wireless security systems, consumer electronic remote control, etc.

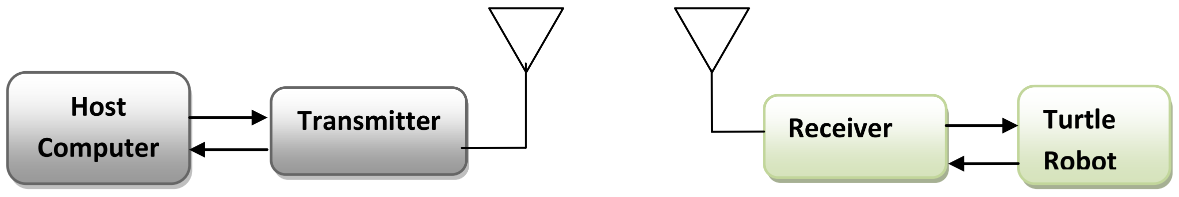


Figure 3.3 Communication protocol between PC and Robot

3.4 Wireless programming method

The communication is establishes between the computer and the robot using RF communication. The type of communication is a two way communication; the computer send instructions to the robot and after executing the robot send the feedback. This method is call hybrid type of communication. Data are processed by both computer and the robot and thus required less memory. The user uses logo interface on the turtle graphic software to enter the command. The command is then sent wirelessly (via RF) to the robot, the logo interpreter on the robot will translate the code and sent to the microcontroller which will compile and run the code.

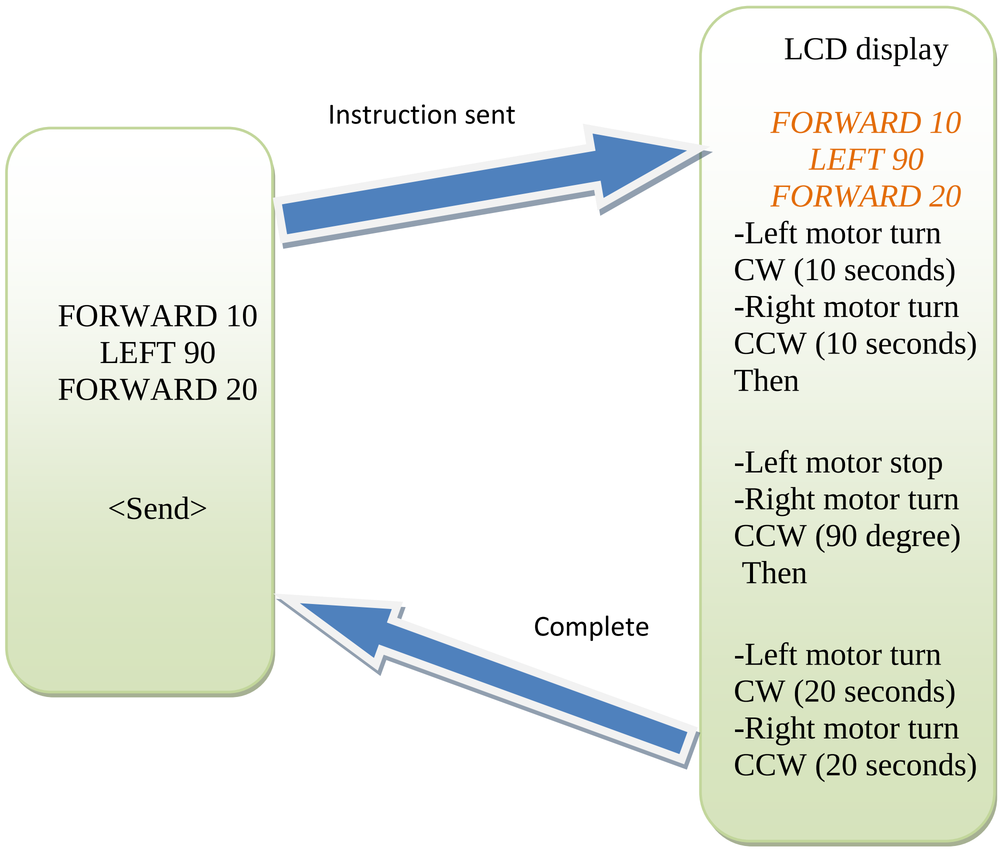

Instruction sent
FORWARD 10
LEFT 90
FORWARD 20
<Send>
LCD display
FORWARD 10
LEFT 90
FORWARD 20
-Left motor turn CW (10 seconds)
-Right motor turn CCW (10 seconds)
Then
-Left motor stop
-Right motor turn CCW (90 degree)
Then
-Left motor turn CW (20 seconds)
-Right motor turn CCW (20 seconds)
Complete

Figure 3.4: turtle robot programming

CHAPTER IV

4.0 RESULTS AND DISCUSSION

4. 1 Robot configuration

During the designing phase in FYP I, the hardware chosen was the MC40 from the Educational Mobile Robot 2.0 from Cytron [13], the hardware is changed to arduino Maga 2560 main board. Arduino is a powerful dynamic programming that uses a wide variety of application and also have library module included for specific functions such as lcd display, servo, and many more. It offers digital and analog input/output.

The design includes two main parts:

- The Turtle Robot
- The Turtle Graphic
- The communication between turtle Robot and Turtle graphic

The chassis of the robot is made of aluminum plate; the wheels are connected to a 360 degree servo motor. The robot is designed to have a low gravity in order to reduce the interference during line following, the robot uses LSS05 infra-red sensors for line following. A RF module is added on the robot for communication.

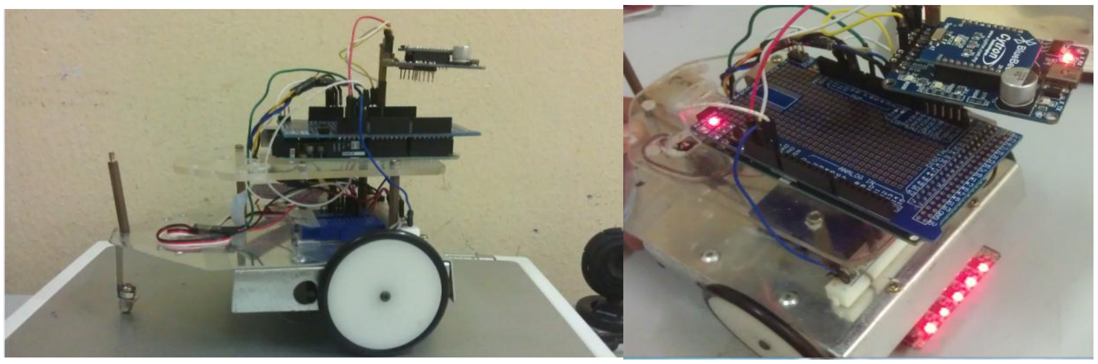


Figure 4.1: Turtle Robot

4.1.1 Hardware Settings

- RF description

The Radio Frequency communication module consists of transmitter-receiver pair. It is used in a variety of applications. They can operate on a supply voltage range from 3V to 5V. The transmitter has a transmission power of 1 watt, providing a range of nearly 100 meters.

It is to be noted that this transmitter-receiver pair cannot be used to transmit and receive analog signal, rather it can only be used to transmit and receive serial digital data streams with maximum 20 Kbps baud rate.

Notice also that one-transmitter-to-many-receivers can be implemented, while many transmitters cannot be used together in vicinity to avoid interference problems. Consequently, this pair can only be used for half duplex, one-way communications (i.e. the transmitter is in one side and one or multiple receivers only on the other side. The other side cannot reply back on the transmitting side, only listens to it).

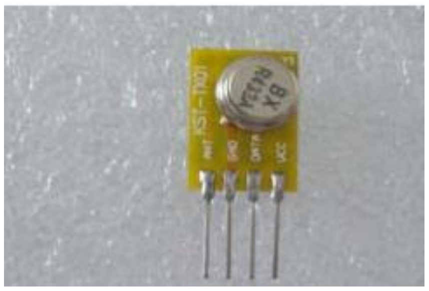


Figure 4.2: RF Transmitter

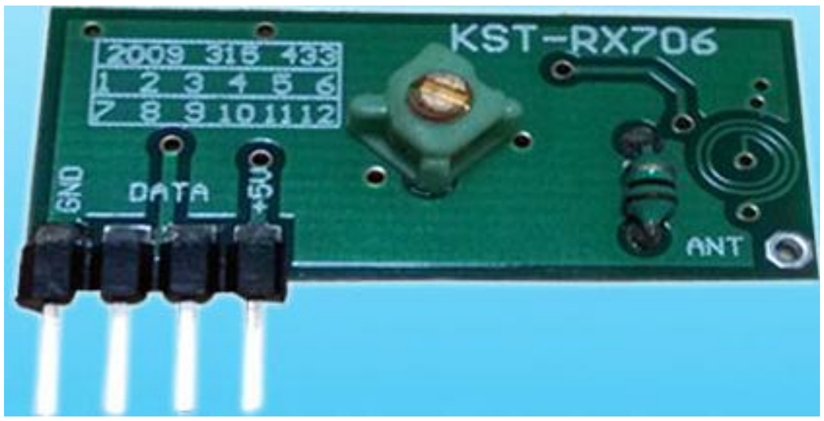


Figure 4.3: RF Receiver

4.1.2 Servo Motor Tuning

Servo motors are generally known to be accurate. They are widely used in precise and accurate systems. They generally rotate up to 180 degrees, but for the project we are using two 360 degrees free rotation servos. Servos come with three wires for the connection, power, ground and the signal. In clockwise rotation, servos operate full speed at zero (0), and zero speed at ninety (90), in counter clockwise full speed is at hundred thirty (130) while zero speed is at ninety (90). They generally need to be tuned before they can accurately respond to a specific parameter.

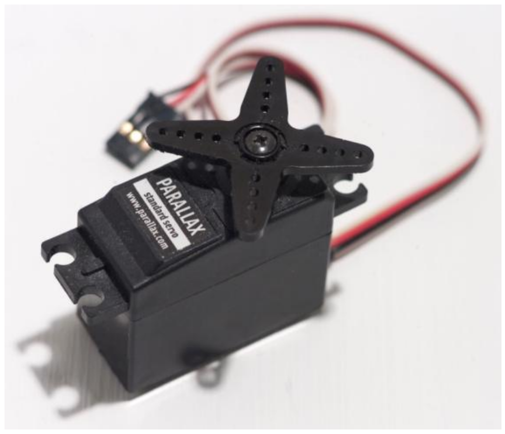


Figure 4.4: Servo motor

The logo interpreter is created and loaded in the hardware, the interpreter contains five (5) main functions: FORWARD, BACKWARD, LEFT, RIGHT and LINE FOLLOWING. For every function, groups of specific command are set, different from other functions these command allow the robot to execute the task. The input is filtered and assorted by the interpreter function, which passes to the respective five functions.

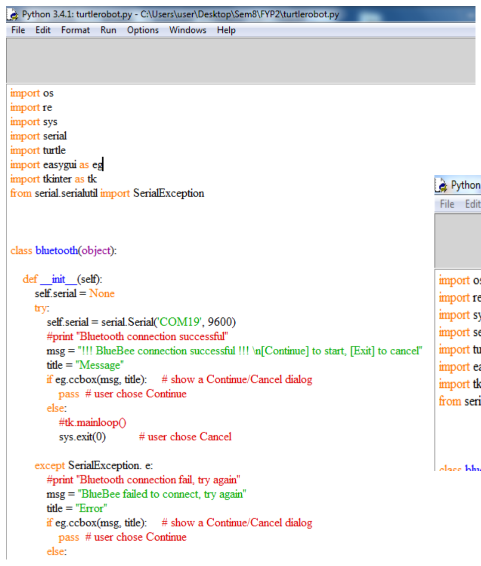
The figure bellow shows on the left hand side the command to move the robot and on the right hand side the line following command.

There are three version of coding for Turtle Graphic: TKinter, Mark1 and Mark2. Focus will be given mostly on TKinter, Mark1 and Mark2 having fixe delay function. The code will be developed using arduino first and later using Python programming language.

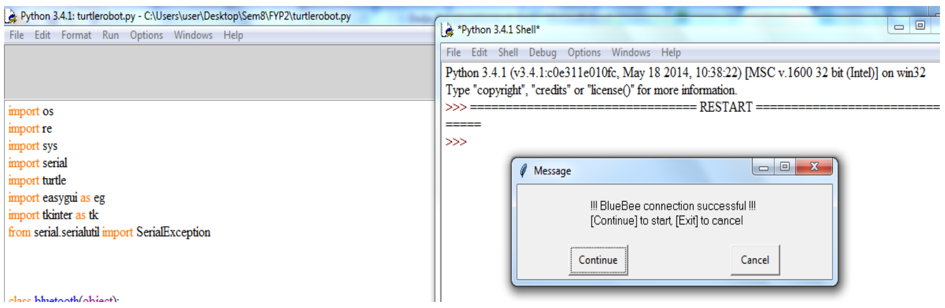
4.1.3 Turtle Graphic Interpreter

One version of suitable turtle graphic is created (TKinter) using Python programming language in Python IDLE 3.4 shell, on Window 7 OS. Python is a powerful dynamic programming language that used in a wide variety of application domains and it is also have many library modules which carry out specific functions.

EasyGui is used to designed simple interface that is easy to understand. The use of easyGui do not always required prior knowledge on tkinter, frames, widgets or callbacks. With easyGui, all interactions are invoked by a simple function calls.


Python 3.4.1: turtlerobot.py - C:\Users\user\Desktop\Sem8\FYP2\turtlerobot.py
File Edit Format Run Options Windows Help

```
import os
import re
import sys
import serial
import turtle
import easygui as eg
import tkinter as tk
from serial.serialutil import SerialException



class bluetooth(object):

    def __init__(self):
        self.serial = None
        try:
            self.serial = serial.Serial('COM19', 9600)
            #print "Bluetooth connection successful"
            msg = "!!! BlueBee connection successful !!! \n[Continue] to start, [Exit] to cancel"
            title = "Message"
            if eg.ccbox(msg, title):     # show a Continue/Cancel dialog
                pass  # user chose Continue
            else:
                #tk.mainloop()
                sys.exit(0)          # user chose Cancel

        except SerialException. e:
            #print "Bluetooth connection fail, try again"
            msg = "BlueBee failed to connect, try again"
            title = "Error"
            if eg.ccbox(msg, title):     # show a Continue/Cancel dialog
                pass  # user chose Continue
            else:
```


Python 3.4.1: turtlerobot.py - C:\Users\user\Desktop\Sem8\FYP2\turtlerobot.py
File Edit Format Run Options Windows Help
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Message
Continue
Cancel

```
import os
import re
import sys
import serial
import turtle
import easygui as eg
import tkinter as tk
from serial.serialutil import SerialException
```

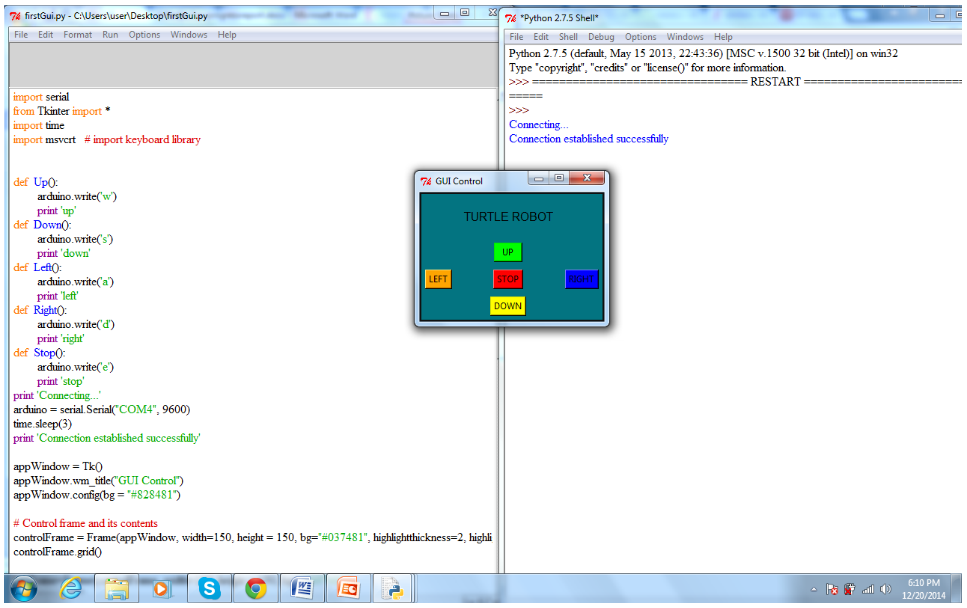
```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ===========================
=====
>>>
```

!!! BlueBee connection successful !!!
[Continue] to start, [Exit] to cancel

The caption above shows how to import easygui and tkinter in python programming language.

The python GUI is designed using tkinter, the button designed is linked to different function (Up, Down, Right, and Left); the function will be interpreted using the interpreter function in the arduino, respectively (‘f’, ‘b’, ‘r’, ‘l’) for (‘Up’, ‘Down’, ‘Right’, ‘Left’).

In other word, when “Up” is sent, the character ‘f’ is coded in ISCII and sent wirelessly (RF) to the robot which will interpret it and execute the forward function.



```
import serial
from Tkinter import *
import time
import msvcrt   # import keyboard library


def Up():
       arduino.write('w')
       print 'up'
def Down():
       arduino.write('s')
       print 'down'
def Left():
       arduino.write('a')
       print 'left'
def Right():
       arduino.write('d')
       print 'right'
def Stop():
       arduino.write('e')
       print 'stop'
print 'Connecting...'
arduino = serial.Serial("COM4", 9600)
time.sleep(3)
print 'Connection established successfully'

appWindow = Tk()
appWindow.wm_title("GUI Control")
appWindow.config(bg = "#828481")

# Control frame and its contents
controlFrame = Frame(appWindow, width=150, height = 150, bg="#037481", highlightthickness=2, highli
controlFrame.grid()
```

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ==========================
=====
>>>
Connecting...
Connection established successfully
```

Figure 4.5: python GUI

Module in python is used to provide desirable function in the program. The turtle graphic library is suitable to create a functional Turtle Graphic. The modules involved in this turtle graphic are:

a) **re.py**: provides regular expression matching operations[16]. Used to filter input and assorted to the functions assigned

b) **serial.py**: encapsulates the access for the serial port. Use to communicate to the robot using COM port by Bluetooth devices

c) **sys.py**: provide access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. Use to close the program and turtle graphic’s window.

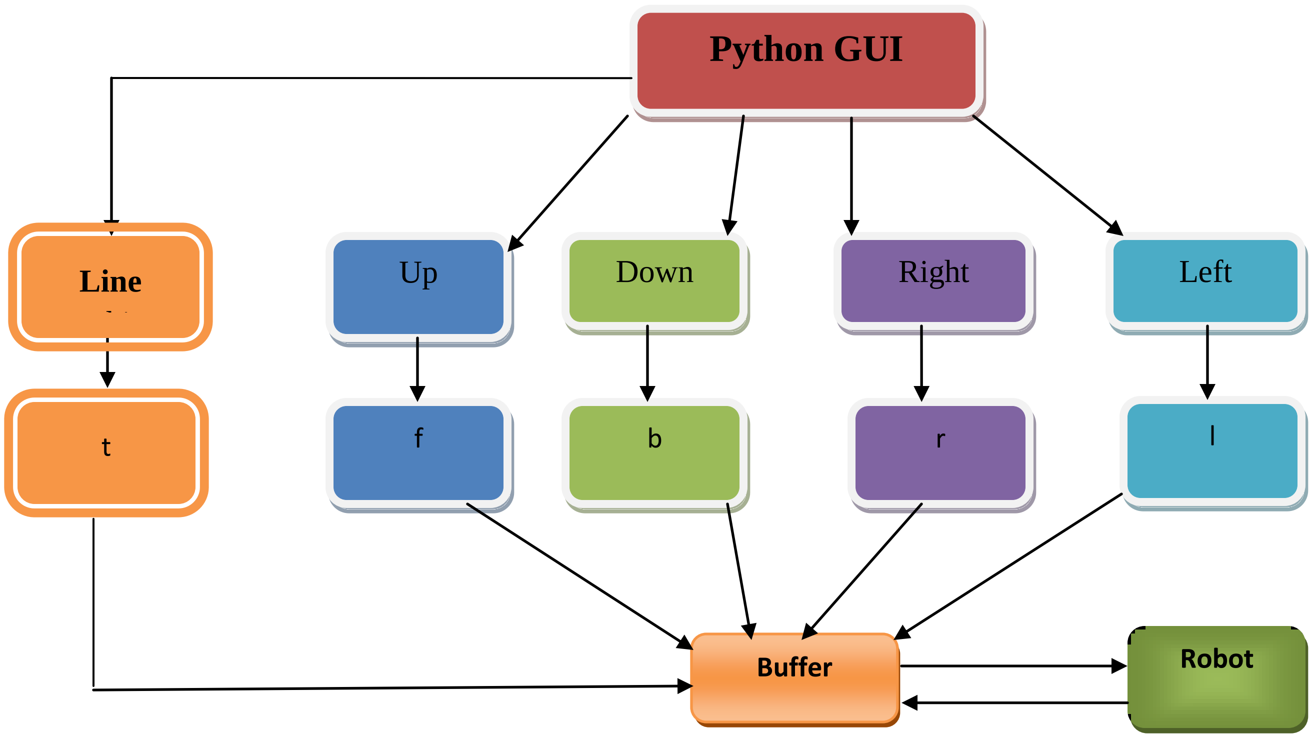
d) **turtle.py**: provides turtle graphics primitives, in both object-oriented and procedure-oriented ways[16]. Turtle.py is a basic turtle graphic in python IDLE.

4.1.4 Communication Between Graphic and Robot

The communication between the graphic and the robot is done using RF module. The choice of RF over other wireless modules (ZigBee, Wi-Fi, UWB (Ultra-Wideband)) is mostly because it's a low power device, consume less energy, easy to use and is cheap. The graphic system is simulated using python software. The advantage of python as graphic interface is the smooth execution of the command. The RF module is easy to use and the connection straight forward. For the project, we are using two arduino boards. One is arduino (Uno) is connected to the computer (PC) via serial port and connected to the transmitter. The second arduino (Mega) is connected to the receiver and the robot.

The robot can be switch from line following system. In the line following system, the robot follows a white line using the infra-red sensor. This technique can be used to reposition the robot automatically.

If switched to line following mode, the robot will execute the arduino line tracking program till its call to stop the execution.

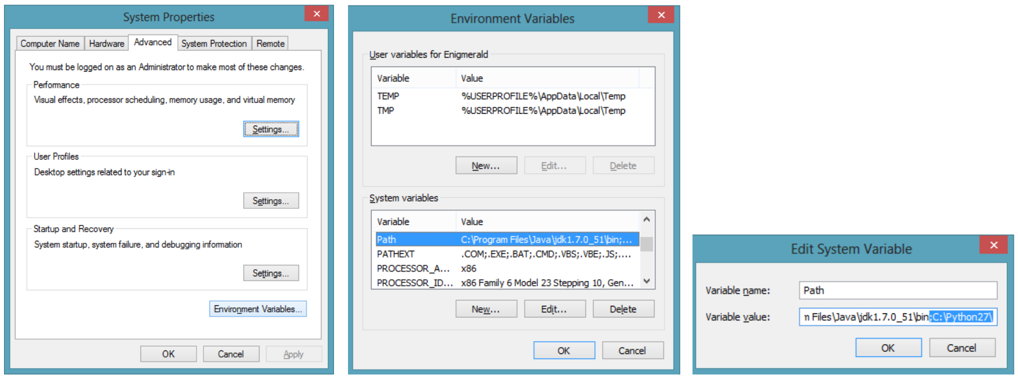


Flow of instructions from user to robot

4.2.1 Software Settings

- Python installation

Python can be downloaded online from python website, there are different versions of python (2.x and 3.x) and for this project we are using python 2.7. After installation after installation of python, we need to do some settings to make sure it runs properly. The settings consist to configure the environment variables to be able to properly execute python command. To do so, we edit the path in the environment variables by adding **C:\Python27\ at the end.**



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

User variables for Enigmerald

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

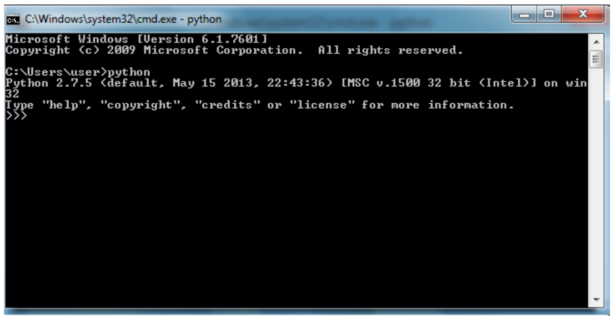
System variables

Variable	Value
Path	C:\Program Files\Java\jdk1.7.0_51\bin;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 6 Model 23 Stepping 10, Gen...

Variable name: Path
Variable value: n Files\Java\jdk1.7.0_51\bin;C:\Python27\

Figure 4.6: python configuration

A simple test can be done to insure that python is running properly. Open command prompt on your computer and type "python", the result should look like the figure below. But you can also check from the python build-in IDE or using notepad ++ if you have it install in your computer (if you are using notepad ++, make sure to save your code with the extension ".py").



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 4.7: python installation

The programming and development of applications such as GUI require us to have pyserial for the communication, tkinter and pygame in some cases. Python generally comes with tkinter in its library but in case you have problem with the GUI, you can always download pyserial, tkinter and pygame online.

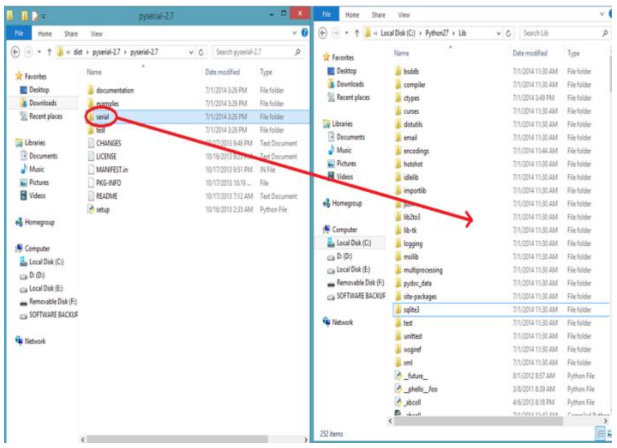
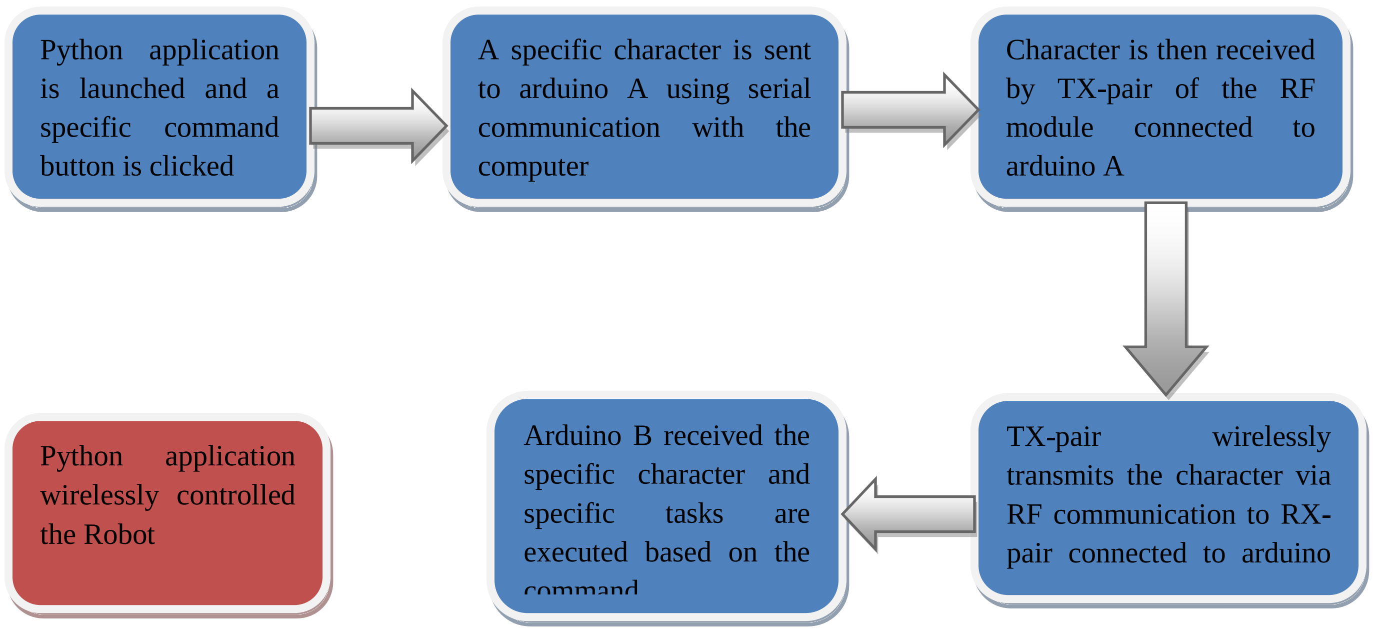


Figure 4.8: pyserial installation

The concept we are using here is to control the robot via a Python GUI which is very similar to that of controlling it via keyboard strokes. The communication process between the two Arduinos is RF communication.


Python application is launched and a specific command button is clicked

A specific character is sent to arduino A using serial communication with the computer

Character is then received by TX-pair of the RF module connected to arduino A

TX-pair wirelessly transmits the character via RF communication to RX-pair connected to arduino

Arduino B received the specific character and specific tasks are executed based on the command

Python application wirelessly controlled the Robot

Figure 4.9: instruction flow from PC to robot

- **Arduino software installation**

  Arduino is open source software; it can be directly downloaded from arduino website or any other related website. It is easy to install and to use. The code in arduino is generally divided into two main sections: the setup (void setup) and the loop (void loop).

  For the purpose of this project you will have to download virtualwire and add to your arduino library if you cannot import it.

CHAPTER 5

5.0 CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

Python GUI gives an interesting and attractive platform to interact with the robot, a simple logo interpreter was created using tkinter, capable of processing logo instruction received from the user and send the data to the robot using radio frequency communication. This graphic is very attractive to children and they can therefore easily learn programming. The graphic system is user friendly.

During this project, the choice of communication was first oriented toward Bluetooth, but due to instability of the connection, the choice was reoriented toward Radio Frequency module.

5.2 Recommendation

There are few recommendations on the things to add on the robot to make more autonomous: adding ultrasonic sensors for object detection will give more autonomy to the robot in term of direction.

	Task/activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14		15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	Titles selection	🟨	🟨																											
2	Literature Review	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨																
3	Extended proposal																													
	-Background studies				🟨	🟨	🟨	🟨	🟨	🟨																				
	-Problem statement										🟨	🟨																		
	-Objectives										🟨	🟨	🟨																	
	-Literature review						🟨	🟨	🟨	🟨	🟨																			
	-Methodology					🟨	🟨	🟨	🟨	🟨	🟨	🟨																		
4	Proposal defense and progress evaluation							🟨	🟨	🟨																				
5	Draft report											🟨	🟨	🟨	🟨															
6	Final report														🟨															
7	Design wireless Turtle robot																🟨⬛	🟨⬛	🟨⬛	🟨⬛	🟨⬛									
8	Design Turtle graphic in arduino and python																	🟨⬛	🟨⬛	🟨⬛	🟨⬛	🟨⬛								
9	Create interpreter for Logo to python																						⬛	⬛	⬛					
10	Synchronization between Turtle graphic and robot																							⬛	⬛	⬛	⬛			
11	Testing and adjustment																									⬛	⬛	⬛	⬛	

⬛ Planned

🟨 Achieved

Table 1: Gantt-Chart FYP I & FYP II

References

1. Diane O. Cuneo: *young children and turtle graphics programming: understanding turtle commands.* University of Massachusetts, Amherst
2. Cynthia Solomon, Logo Papert: *A case study of a young child doing turtle graphics in LOGO*, AFIPS '76 proceedings of the June 7-10, 1976, national computer conference and exposition. P. 1049-1056
3. Wurman, R.S., et al., *Information anxiety 2*. 2001: Que. 308.
4. Dictionary.com, in *Collins English Dictionary - Complete & Unabridged*, HarperCollins Publishers.
5. Papert, S., *MINDSTORM : Childern, Computers, and Powerful Ideas*. 1980, New York: Basic Books, Inc., Publishers. 230.
6. Eveland, E. *win32 F/OSS development - Logo*. 12/3/2008 [cited 2013 25/6/2013]; Available from: http://evelands.net/evan/fossDevel_logo.php.
   Harvey, B., *Beyond Programming*. Computer Science Logo Style. Vol. 3. 1997.

   Sheehan, R. *Lower floor, lower ceiling: easily programming turtle-graphics*. in *Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on*. 2000.

7. Fulcher, J.A., *Fun and games and microcomputer interfacing (laboratory exercises).* Micro, IEEE, 1991. **11**(1): p. 18-21.
8. Robert Tinney, Tasman turtle robot; available from: http://incolor.inetnebr.com/bill_r/tasman_turtle_page.htm
9. Megan Treacy, Technology / Gadgets, December 4, 2013; available from: http://www.treehugger.com/gadgets/robotic-sea-turtle-will-dive-explore-ship-wrecks.html
10. AUVAC. *SAUVIM configuration*. AUV System Spec Sheet [cited 2013 25/6/2013]; Available from: http://auvac.org/uploads/configuration/SAUVIM%202008%20[Hanging].jpg.
11. Micaela Serra, Department of computer science, University of Victoria; available from: http://webhome.cs.uvic.ca/~mserra/HScodesign.html
12. Micaela Serra, Department of computer science, University of Victoria; available from: http://webhome.cs.uvic.ca/~mserra/HScodesign.html
13. Cytron.com. *Cytron Bluetooth Module*. [cited 2013 24/6/2013]; Available from: http://www.cytron.com.my/viewProduct.php?pcode=BLUEBEE&name=Cytron%20Bluetooth%20Module.
14. Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen, The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Nov. 5-8, 2007, Taipei, Taiwan
15. Papert, Seymour and Cynthia Solomon, Twenty Things to Do with a Computer, MIT, AI Lab. LOGO Memo 3, July 1971. Also in Educational Technology, April 1972.
16. The Python Standard Library, 1990, Python Software Foundation.

APPENDIX A

TOOLS REQUIRED

Hardware

- Arduino Mega 2560 R3-Main board
- RF module
- Servo motor
- Infra-Red Sensors

RF module's characteristics

Transmitter:

- ➢ Operating voltage: 3V to 5V
- ➢ Operating frequency: 315Mhz to 433,92Mhz
- ➢ Operating temperature: -40C to 80C
- ➢ Transmission power: 1watt
- ➢ Modulation type: ASK

Receiver:

- ➢ Operating voltage: 5V
- ➢ Operating frequency: 315Mhz to 433,92Mhz
- ➢ Operating current: 8mA maximum rating
- ➢ Modulation type: ASK
- ➢ Receiver sensibility: -115dB

Software

- Python v3.4
- Turtle graphic software
- Arduino software

APPENDIX B

Python code for GUI design

```
import serial
from Tkinter import *
import time
import msvcrt   # import keyboard library


def  Up():
        arduino.write('w')
        print 'up'
def  Down():
        arduino.write('s')
        print 'down'
def  Left():
        arduino.write('a')
        print 'left'
def  Right():
        arduino.write('d')
        print 'right'
def  Stop():
        arduino.write('e')
        print 'stop'
print 'Connecting...'
arduino = serial.Serial("COM4", 9600)
time.sleep(3)
print 'Connection established successfully'

appWindow = Tk()
appWindow.wm_title("GUI Control")
appWindow.config(bg = "#828481")

# Control frame and its contents
controlFrame = Frame(appWindow, width=150, height = 150, bg="#037481", highlightthickness=2,
highlightbackground="#111")
controlFrame.grid()

btnFrame=Frame(controlFrame, width=150, height = 150, bg="#037481")
btnFrame.grid()
about = "TURTLE ROBOT"

name = Label(btnFrame, width=15, height=3, text=about, font="bold", justify=CENTER, bg="#037481")
name.grid(row=0, column=2)


upBtn = Button(btnFrame, text = "  UP  ", command=Up, bg="green",)
upBtn.grid(row=1, column=2, padx=5, pady=5)

downBtn = Button(btnFrame, text= "DOWN", command = Down, bg = "yellow")
downBtn.grid(row=5, column=2, padx=5, pady=5)

leftBtn = Button(btnFrame, text="LEFT", command = Left, bg = "orange")
leftBtn.grid(row=3, column=0, padx=5, pady=5)

rightBtn = Button(btnFrame, text="RIGHT", command = Right, bg = "blue")
rightBtn.grid(row=3, column=4, padx=5, pady=5)

stopBtn = Button(btnFrame, text="STOP", command = Stop, bg = "red")
stopBtn.grid(row=3, column=2, padx=5, pady=5)

appWindow.mainloop()
```

APPENDIX C

Transmitter code on arduino A

```
#include <VirtualWire.h>

void setup()
{
 int ledPin = 13;
  Serial.begin(9600);
  vw_setup(2000);      //bits per second
  vw_set_tx_pin(3);    //transmitter data to pin 3
}

void loop()

{
   if (Serial.available() > 0)    // if there is incomming data
 {

     int sendByte = Serial.read();  // read data

     digitalWrite(13, HIGH);
     delay(1000);
     digitalWrite(13, LOW);
     delay(1000);
     digitalWrite(13, HIGH);
     delay(1000);
     digitalWrite(13, LOW);
     delay(1000);

     switch (sendByte)
     {
      case 'w':                    // if w is pressed
        {
         char *msg2 = "w";
         digitalWrite(13, HIGH);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to the receiver
         vw_wait_tx();                          // wait untill the whole message is gone
        // digitalWrite(13, false);
         break;
        }

        case 's':                    // if s is pressed
        {
         char *msg2 = "s";
         digitalWrite(13, true);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to the receiver
         vw_wait_tx();                          // wait untill the whole message is gone
         digitalWrite(13, false);
         break;
        }

         case 'a':                    // if a is pressed
        {
         char *msg2 = "a";
         digitalWrite(13, true);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to the receiver
         vw_wait_tx();                          // wait untill the whole message is gone
         digitalWrite(13, false);
         break;
        }

         case 'd':                    // if d is pressed
        {
         char *msg2 = "d";
         digitalWrite(13, true);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to receiver
         vw_wait_tx();                          // wait untill the whole message is gone
         digitalWrite(13, false);
         break;
        }

        case 'e':                    // if e is pressed
        {
         char *msg2 = "e";
         digitalWrite(13, true);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to the receiver
         vw_wait_tx();                          // wait untill the whole message is gone
         digitalWrite(13, false);
         break;
        }
        default: //if any other value entered
        {
        Serial.println("wrong entry");
        }
     }
 }}
```

APPENDIX D

Receiver code on arduino B

```
//#include <LiquidCrystal.h>
//#include <Servo.h>
//#include <Wire.h>
#include <VirtualWire.h>

#define rightMotor 6
#define leftMotor 11
//Servo servo1;
//Servo servo2;

//int servoPin1 = 11;
//int servoPin2 = 6;


void setup()
{
    vw_setup(2000);
    vw_set_rx_pin(4);   //reciever at Digital pin 4
    vw_rx_start();       //start the reciever PLL running

   pinMode(rightMotor, OUTPUT);
   pinMode(leftMotor, OUTPUT);

}


void loop()

{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
      uint8_t buflen = VW_MAX_MESSAGE_LEN;
      if (vw_get_message(buf, &buflen))    // Non-blocking
      {
          int i;
          digitalWrite(13, true);
          for (i=0; i<buflen; i++)
          {
            Serial.print(buf[i]);
           if (buf[i] =='w')
               {
                 forward();
               }
            if (buf[i] == 's')
               {
                 backward();
               }
            if (buf[i] == 'e')
                {
                  stop();
                }
             if (buf[i] == 'a')
                {
                  left();
                }
              if (buf[i] == 'd')
                {
                   right();
                }
          }
        }
      }
```

```
  void forward()
 {

   analogWrite(rightMotor, 130);
   analogWrite(leftMotor, 0);
   //Serial.println("going straight..");
   //lcd.print("FORWARD");
 }

  void backward()
 {

   analogWrite(rightMotor, 0);
   analogWrite(leftMotor, 130);
   //Serial.println("going backward..");
   //lcd.print("BACKWARD");
 }


void right()
 {
    analogWrite(rightMotor, 130);
    analogWrite(leftMotor, 95);
    //Serial.println("going right..");
    //lcd.print("GOING RIGHT");

 }


void left()
{

   analogWrite(rightMotor, 130);
   analogWrite(leftMotor, 97);
   // Serial.println("going left..");
    //lcd.print("GOING LEFT..");

}


 void stop()
 {

   analogWrite(rightMotor, 97);
   analogWrite(leftMotor, 97);

 }
```

APPENDICE E

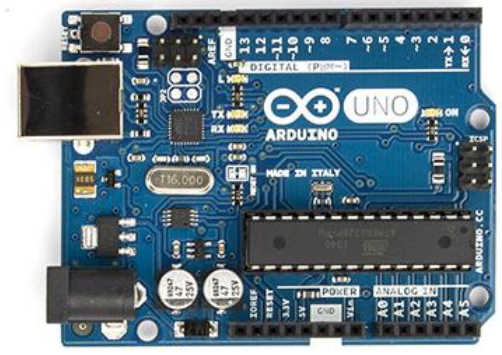
Line following code

```
#include <LiquidCrystal.h>

#include <Wire.h>
//include servo library
//create servo object

#include <Servo.h>


Servo servo2;
Servo servo1;
/*
  * RS: Pin 29
  * EN: Pin 51
  * D4: Pin 22
  * D5: Pin 23
  * D6: Pin 24
  * D7: Pin 25

*/
LiquidCrystal lcd(29, 51, 22, 23, 24, 25);
//define the pins
int servoPin1 = 11;
int servoPin2 = 6;

int LeftSen = 2;  //s5
int LeftMSen = 3;  //s4
int MidSen = 8;    //s3
int RightMSen = 9;  //s2
int RightSen = 10;  //s1
char cmd[10] = "t";
int x;


//function declaration
void setup()
{
  /*  pinMode(RightEn, OUTPUT);
    pinMode(RightDir, OUTPUT);
    pinMode(LeftEn, OUTPUT);
    pinMode(LeftDir, OUTPUT);
    digitalWrite(RightDir,HIGH);
    digitalWrite(LeftDir, LOW); */
    lcd.begin(16,2);
    lcd.print("Turtle Graphic");

    servo1.attach(servoPin1); //can write 11 insted of servoPin1
    servo2.attach(servoPin2); //can write 12 insted of servoPin2


     Serial.begin(9600);
     Serial.println("Turtle Graphic");
}


   void straight()
   {
     if((digitalRead(LeftSen) == 0) && (digitalRead(LeftMSen) == 1) && (digitalRead(MidSen) == 1) && (digitalRead(RightMSen) ==
0) && (digitalRead(RightSen) == 0))
    {
```

```
    servo2.write(130);
    servo1.write(0);
    Serial.println("going straight..");
    lcd.print("FORWARD");
   }

  }

  void TurnLeft()
  {
    if((digitalRead(LeftSen) == 0) && (digitalRead(LeftMSen) == 0) && (digitalRead(MidSen) == 1) && (digitalRead(RightMSen) ==
1) && (digitalRead(RightSen) == 0))
   {
     servo2.write(110);
     servo1.write(90);
     Serial.println("going right..");
     lcd.print("GOING RIGHT..");
   }

  }

  void TurnMidLeft()
  {
    if((digitalRead(LeftSen) == 0) && (digitalRead(LeftMSen) == 0) && (digitalRead(MidSen) == 0) && (digitalRead(RightMSen) ==
1) && (digitalRead(RightSen) == 1))
   {
     servo2.write(100);
     servo1.write(90);
     Serial.println("going more right..");
     lcd.print("GOING RIGHT..");
   }

  }

  void TurnMostLeft()
  {
    if((digitalRead(LeftSen) == 0) && (digitalRead(LeftMSen) == 0) && (digitalRead(MidSen) == 0) && (digitalRead(RightMSen) ==
0) && (digitalRead(RightSen) == 1))
   {
     servo2.write(100);
     servo1.write(90);
     Serial.println("going all right..");
     lcd.print("GOING RIGHT..");
   }

  }

  void TurnRight()
  {
    if((digitalRead(LeftSen) == 1) && (digitalRead(LeftMSen) == 1) && (digitalRead(MidSen) == 0) && (digitalRead(RightMSen) ==
0) && (digitalRead(RightSen) == 0))
   {
     servo2.write(110);
     servo1.write(95);
     Serial.println("going left..");
     lcd.print("GOING LEFT");
   }

  }

  void TurnMidRight()

  {
    if((digitalRead(LeftSen) == 1) && (digitalRead(LeftMSen) == 0) && (digitalRead(MidSen) == 0) && (digitalRead(RightMSen) ==
0) && (digitalRead(RightSen) == 0))
   {
     servo2.write(130);
     servo1.write(95);
     Serial.println("going more left..");
     lcd.print("GOING LEFT ");
   }

  }

  void Cont()
  {
    if((digitalRead(LeftSen) == 1) && (digitalRead(LeftMSen) == 1) && (digitalRead(MidSen) == 1) && (digitalRead(RightMSen) ==
1) && (digitalRead(RightSen) == 1))
   {
     servo2.write(130);
     servo1.write(90);
     Serial.println("Junction occurs..");
     lcd.print("JUNCTION");
   }

  }



void loop()
{

  lcd.setCursor(0,1);
  case ‘t’

  {
               char *msg2 = "w";
         digitalWrite(13, HIGH);    // flash the light
         vw_send((uint8_t *)msg2, strlen(msg2)); // send byte to the receiver
         vw_wait_tx();                          // wait untill the whole message is gone
        digitalWrite(13, false);
    straight();
   TurnLeft();
   TurnMidLeft();
   TurnMostLeft();
   TurnRight();
   TurnMidRight();
   Cont();

}   }
```

APPENDIX F

ARDUINOS BOARD

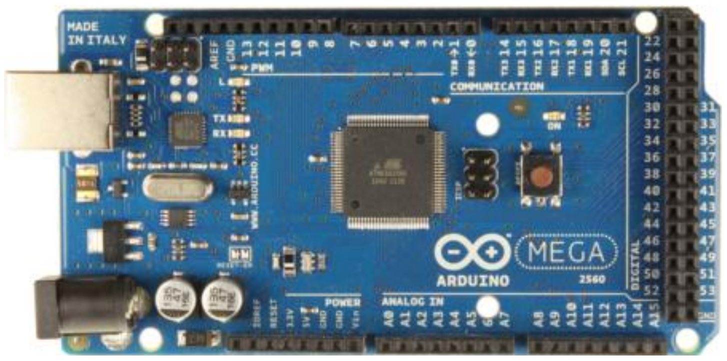


Arduino Uno main board

Specifications

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 Ma
DC Current for 3.3V Pin	50 Ma
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

ARDUINO MEGA MAIN BOARD



Arduino Mega board

Specifications

Microcontroller ATmega2560

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 54 (of which 14 provide PWM output)

Analog Input Pins 16

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 256 KB of which 8 KB used by bootloader

SRAM 8 KB

EEPROM 4 KB

Clock Speed 16 MHz