REMOTE PROGRAMMING OF A TURTLE ROBOT

By

AHMAD SHAHIR SAFFUAN BIN ZAINAL ABIDIN

FINAL PROJECT REPORT

Submitted to the Department of Electrical & Electronic Engineering
in partial fulfillment of the Requirement
for Degree
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

REMOTE PROGRAMMING OF A TURTLE ROBOT

by

Ahmad Shahir Saffuan Bin Zainal Abidin
13010

A project dissertation submitted to the
Department of Electrical &Electronic Engineering
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for
the Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:

____

Abu Bakar Sayuti Hj Mohd Saman

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

JANUARY 2013

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except 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.

____

Ahmad Shahir Saffuan Bin Zainal Abidin

ABSTRACT

Turtle graphics is a subset of Logo programming language, used extensively to introduce programming to children. Conventionally the turtle is visually represented on the computer screen as an object that performs movements based on the set of instructions issued by a programmer. A similar robotic turtle that can be programmed using turtle graphics commands should enhance the learning experience as well as sustain the interest of young programmers that the programming language is intended for. Programming the robot can be achieved by entering the commands directly onto the robot or on a separate terminal. The terminal can be attached via a wire, or wireless. Programming using a wireless programming terminal, which is more practical, can be achieved either via over-the-air (OTA) method or remote control. A hybrid of the two methods is also possible where commands are sent one line at a time the turtle robot to be interpreted and suitable movement performed. The hybrid method is proposed for this project so that the system will be more flexible and extensible for further development in future. There are two types of interpreter (Mark 1 and Mark 2) created as the result from this project which has different way in term of system, communication between graphic and robot, and the instruction's execution. Both interpreters have advantages and disadvantages which involved delay processing, transferring data, and instruction's execution. Mark 1 interpreter is chosen as the Turtle Graphic Interpreter based on the advantages (discussed in the result section) and a list of logo command is created based on the graphic interpreter Mark 1.

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. I have taken efforts in completing this report. However, it would not have been possible without the kind support and help of many individuals and organizations.

A special gratitude I give to my Final Year Project supervisor, Abu Bakar Sayuti Bin Hj Mohd Saman, whose contribution in stimulating suggestions, encouragement, guidance and also constant supervision during the FYP 1 and FYP 2 and necessary information especially regarding the project and writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the all FYP coordinator, especially head of FYP coordinator, Dr. Nazreen Bt Badruddin, for conducting a lot of seminars and briefing to assist and guiding final year student to finish up the project within the given duration of time. Finally, a special thanks goes to my colleagues who help me on opinion and suggestion about the project and other people who involved in this report which have willingly helped me out with their abilities.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background Study

"The role of the teacher is to create the conditions for invention rather than provide ready-made knowledge", quote from Seymour Papert [8], one of the pioneer and developer of the Logo and Turtle graphic. Seymour Papert, who was born in February 29, 1928, has worked at many different places as researcher, including at Massachusetts Institute of Technology (MIT) [9]. He conducts a lot of researches based on how to improve the educational and learning method for children towards understanding of mathematics and sciences, innovation and computer literature (computer language). Due to excessive passion on his researches, Logo and Turtle graphic were born.

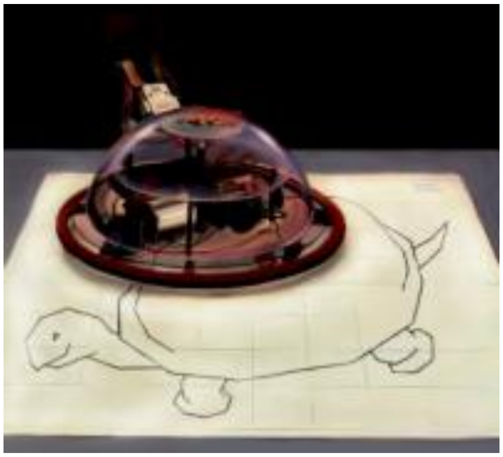
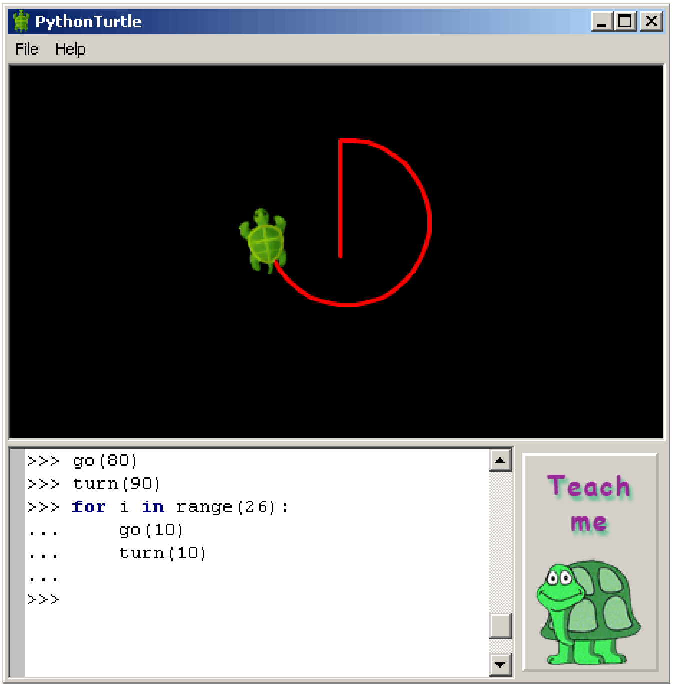


Figure 1: Turtle robot [3]

Turtle graphic is a software program, developed by Seymour Papert. The function of the program is to display a cursor (turtle) that move at Cartesian plane, according to the input from the user keyed in [10]. This program is similar to computer game, but more specific toward arts. The turtle graphic have four main command (to give direction of the turtle), that is FORWARD, BACKWARD, LEFT and RIGHT. The data instructing keyed in from the user will be send to the turtle robot (Logo Turtle) and the robot will execute the command by drawing on the paper. The original purpose of this program created is to provide a simple alternative to attract children to learn computer programming, which normally impossible for them to learn sophisticated computer language at such a young age. Due to its operation that use **Logo environment** as the basic computer literature platform, makes this program easily enough to be understood by the children.

Logo is originated from the word ***lógos***, a Greek word which indicates **word or speech** [11]. In other definition, logo is a shortening of logogram (sign or character representing a word) [12]. By the definition of the word itself, the basic operation of Logo can be seen and understand which used specific instruction and command to carry out the desired operation, almost similar to the object-oriented programming but much simpler ways. Logo is a one of the computer literature, adapted from Lisp language, created for the educational purpose. It helps for non-computer background user to understand and solve problems which are related to the computer literature, using Logo [13].



```
>>> go(80)
>>> turn(90)
>>> for i in range(26):
...     go(10)
...     turn(10)
...
>>>
```

Figure 2: Turtle graphic in PytonTurtle [1]

1.2 Problem Statement

Turtle graphic is a programming software which use logo language programming; function to receive the data instruction from the user and compile it. The compile data will be processed and send to Turtle Robot, function to execute the data instructions and send out feedback to the computer after finish executing. Thus, communication link between computer and robot must be established in order for the computer to deliver the data to the robot and receive feedback from the robot. There are 3 types of programming method that can be done which is direct, wired and wireless.

Using direct programming method, the robot is programmed remotely using remote controller, which has all the necessary hardware to program such as LCD screen and keypad. Using wired programming method, the robot is programmed by computer using wired transmission medium, connected between the robot and the computer. Last but not least, using wireless transmission, the robot is programmed wirelessly to the computer using the air as the medium transmission. There are some drawbacks of using wired transmission rather than wireless transmission. Wired robot has movement and distance limitation based on the type and size of the wire used. The wireless method is more favorable in this project.

The robot needs to communicate with the computer wirelessly, which requires wireless system design that is able to send sentence of word (more than 32bit) between robot and computer. Logo interpreter need to be created to interprets the data instruction from logo language into machine language and vice versa for the feedback of the robot. A set of library have to be created which contain the interpretation of logo language to machine language and vice versa.

1.3 Significance of the Project

This project is focused the integration of a small mobile robot, graphical user interface programming, and wireless communication, to implement a turtle graphics robotic system. To enable the communication between turtle robot and computer, a suitable Logo interpreter must be create, function to process and translate the logo command and instruction in turtle graphic into the computer language that the robot can execute, thus enable the user to control the turtle robot remotely [14]. Deep understanding and vast knowledge on computer literature is needed in order to create such suitable interpreter, especially on how Logo programming and wireless data transmission works.

1.4 Objective

The objectives of this research project are to:

1) Create a simple interactive Logo interpreter, capable of processing logo instructions from the user and sending data instructions to turtle robot.
2) Create wireless system between the program and turtle robot, enable for the program to transmit data instructions to the turtle robot and receive feedback from the turtle robot remotely.

1.5 Scope of Study

In this study, the main subjects under investigation are:

i. Understanding and learning python language to create turtle graphic which is user friendly
ii. To design a reliable wireless system between Turtle Graphic and Turtle Robot to transmit instruction data
iii. Implementing logo command as the default input command in Turtle Graphic
iv. Designing Turtle robot which capable of processing logo command

1.6 Relevancy of the Project

Today, turtle graphic is more than just ordinary computer game to attract children from elementary school to learn the art of computer literature. The application of turtle graphic has been widely used throughout the world, because of simplicity of its computer language (in Logo environment) than can be easily understood by people from any range of age. The application of turtle graphic will be specifically discuss later in literature review. A lot of adaptation and improvement has been made by researchers; incorporate to the Turtle Graphic technology to be implemented on the real life usage.

1.7 Feasibility of the Project

In term of feasibility, there are few aspects that can be estimated. For technical aspect in this project, hardware (Turtle robot) and software (Turtle Graphic Interpreter) has to be created. The hardware can be created using existing robot chasis which are available on the market, while software can be created using existing turtle Python module on python 2.7 IDLE. Both hardware and software need to be modified before implementing wireless system which is the hardest task in this project. A few skill in terms of robotic programming and graphical programming is needed. For time consumption aspect in this project, is not a major issue due to existing resources is used in this project.

Chapter 2

LITERATURE REVIEW

2.1 Evolution of Turtle Logo

In late 70's, computers were not considered as basic needs in daily life, but through the time, people have began to slowly accept the fact were computer is crucial to assist human daily life operation, including teaching method for children using computer. The children see computer everyday but they are not be able to do something with them due to complexity of the computer literature itself which is impossible for children like the age of 9 to understand. Due to this fact, the traditional teaching method have been implement, which using computers to teach children (computer aided teaching). Seymour Papert foresaw this event and likely dream of the future where children themselves teach the computers. The work of Jean Piaget had helping him so much in developing Turtle Graphic [10].

Turtle Graphic has made a decent breakthrough over the decades, which make Turtle Graphic have been evolved so much. A lot of improvement and adaptation have been made by the innovators and researchers for both software (Turtle Graphic) and hardware (Turtle robot), which is not just only focus on children's learning, but also some useful application for some industries as well.

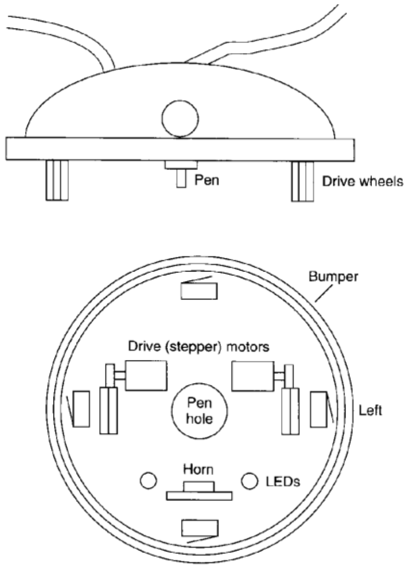


Figure 3 Turtle Robot design in john's research [4]

At the early age of Turtle Graphic technology in early 90's, turtle graphic application is still more toward on enhancing the learning of children in programming. In the research project conduct by John A. Fulcher in 1991, the Turtle Robot were controlled using Motorola 68KECB single-board computer and connected to the computer in Unix environment [4]. Wired based transmission data is used for the communication system between Turtle Robot and the computer in this project research. Thus the mobility of the Turtle Robot has some limitation due to the wired condition.

In the early 20's, the turtle graphic have improve in term of software which is more conveniently to be used by children and more fun. In the research done by Robert Sheehan in year 2000, two things were point out for traditional Turtle Graphic, which is having an environment that is easily understand by children but due to Turtle Graphic used in Logo, there are no restriction to the programmer as Logo is a general purpose of programming [5]. In other words, Turtle Graphic is referring to having lower floor, but high ceiling (lower starting, higher aim). In this research, the Turtle Graphic software used is more likely to be a painting program, which required user to drag the turtle over the screen instead of typing the command to move the turtle to make icon that is full with sets of line. Due to dragging, complex shape can be produce.

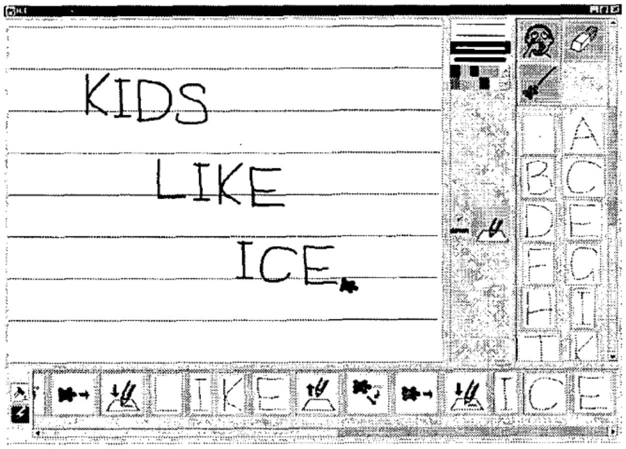


Figure 4: Squiggly line drawn by the user [5]

The evolution of the Turtle graphic has not stopped there yet. Lego has been introduced in the late 20’s which is in present are the most popular toys among kids. In the research of G.Barbara Demo, Giovanni Marciano and Simonetta Siega in 2008, NXT lego bricks was used as the Turtle robot base, while NQCBaby and NXCJunior as the base programming [6]. NQCBaby and NXCJunior were Logo-like programming, which use textual languages which is native to the children. Instead of pen, the robot stimulates the directions of the logo commands without drawing. In this case, the sensors and conditions instruction (if.. then.. else) was used by the kids to avoid obstacles.

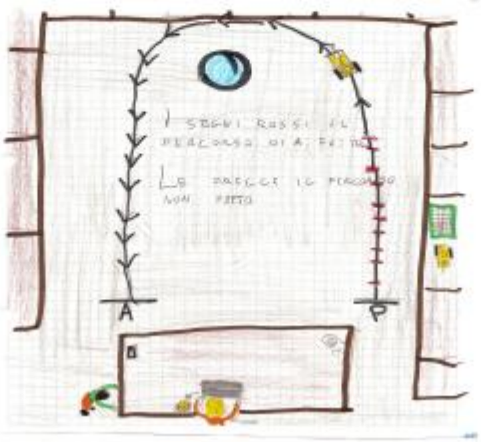
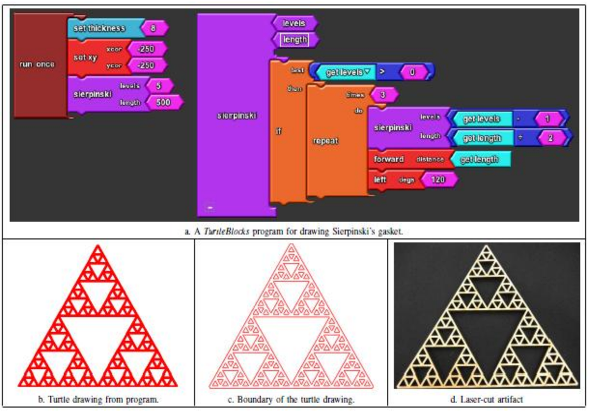

I SEGNI ROSSI IL PERCORSO DI A. FATTO
LE FRECCE IL PERCORSO NON FATTO
A
P

Figure 5 : Path of the robot, which have to avoid obstacles [6]

2.2 Implementation of Turtle Logo in industries application

Turtle Graphic have made some important place towards the industries application starting in early 20’s which conveniently remarks that turtle graphic have brighter future, which is more that a tool for teaching kids. In 2003, turtle graphics were used in control of SAUVIM (semi-autonomous Underwater Vehicle for Intervention Missions)[15]. The robot was developed to perform tasks in underwater which man could not do. The control of the robot was done in Turtle Graphic software. Another example of the application for Turtle graphic, in 2012, was used in control of laser cutting. The laser cutter was controlled using Turtle graphic to make and fabricates tangibles artifacts, which is a set of geometrical shape that drawn in turtle graphic and made from vinyl sheets[7].

Figure 6 : SAUVIM robot [2]



a. A *TurtleBlocks* program for drawing Sierpinski's gasket.

b. Turtle drawing from program.

c. Boundary of the turtle drawing.

d. Laser-cut artifact

Figure 7 : Tangible artifacts produce [7]

2.3 Wireless system for Turtle Logo

Wireless communication system between the robot and the computer can be establish by using specific wireless technology and protocol such as Wi-Fi, Bluetooth, Radio frequency (RF) or GPS and others. This project requires a real-time and fast communication system and therefore Wi-Fi, Bluetooth and Radio Frequency (RF) are the most considerable choices.

Through the first choice, which is Wi-Fi, requires Wi-Fi modems in both robot and computer to communicate each other by using wireless internet access point [16]. The advantages of this technology are that can be used to control robot from a far distance and complex data transfer can be establish, which means a faster data feed. For disadvantages, the Wi-Fi modem is expansive and Wi-Fi technology is complex to be implemented, that might increase the duration of completing this project.

As for the second choice, which is Bluetooth, is simpler from Wi-Fi and yet offers data transmission reliability closer to the Wi-Fi characteristics. Based on the research perform by the Yong Che Fai [17], to implement Bluetooth technology requires Bluetooth module for both computer and robot to communicate without needing any internet access. Thus, the data transfer by Bluetooth technology is fast and yet simpler to operate. Bluetooth offers a cheap wireless technology and vast area of development for a medium distance communication which are the characteristic needed in this project.

Lastly, the third choice, which is Radio Frequency (RF), is a one-way communication wireless technology. In order to make it two-way communication, requires transmitter and receiver in both computer and robot to communicate [18], and from the computer to the transmitter/receiver requires a decoder to decode computer command into binary data as for the RF technology operate on binary transmission. RF is simple technology, cheap and easy to operate but the data transfer in binary limit the data transmission capabilities, which are not favor in this project. The communication distance using RF is quite short compare to Bluetooth and Wi-Fi.

There are 3 ways of programming Turtle Robot, which is through direct, wired or wireless (remotely). This research project will be specifically focus on how to remotely control the Turtle Robot through wireless transmission which is run on structured programming language, C programming. The need of logo interpreter is to convert and interpret the logo command into structured based programming language, which can be understood by the robot. Thus, sets of library containing the interpretation of the logo command – C programming have to be build. Suitable methodology that fit the flow of the program needs to be proposed for this project.

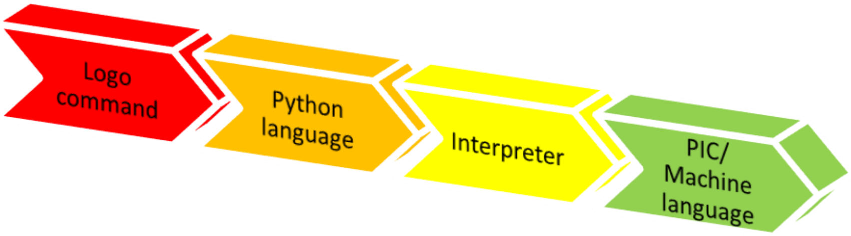


Figure 8 the flow of the programming language usage

Chapter 3

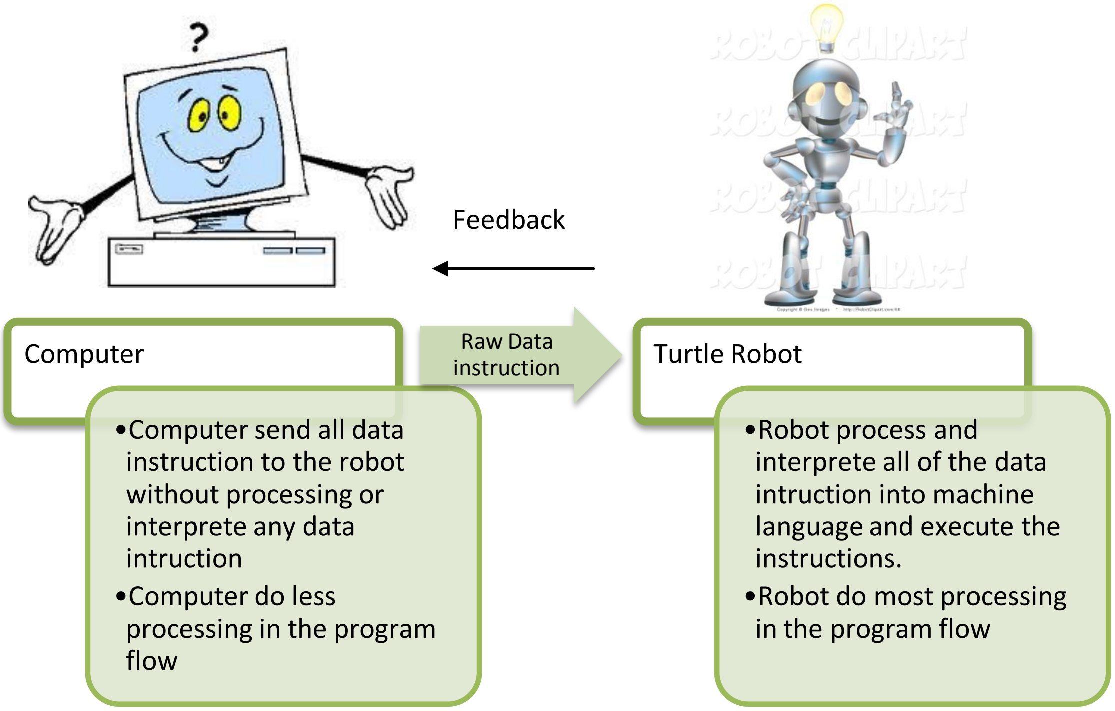
METHODOLOGY

3.1 Research Methodology

The problem and challenge in this project is on creating logo interpreter and wireless transmission system design for turtle graphic and robot. The technique used to design the, processing, interpreting and communication system between computer and robot can be illustrated generally in methodology in which discipline of the technique belongs to. In order for the project can be achievable, methodology need to be proposed. There are a few methodologies that can be proposed in this project which is possible to achieve.

1) Over the Air programming

Computer will send raw data instruction to the robot without process any data instruction at all. The robot receives the data instruction, process and interprets the data instruction into machine language and execute. Robot will send feedback to the computer.



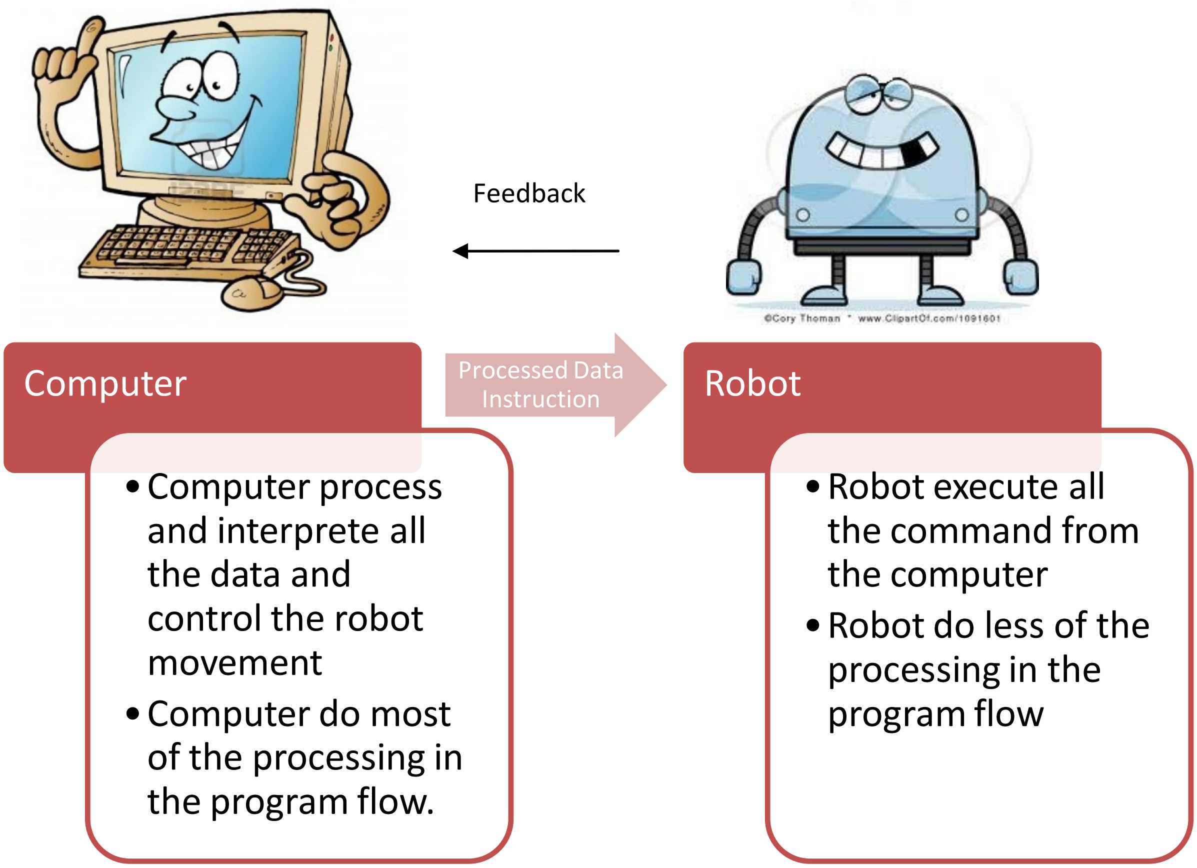
- Computer send all data instruction to the robot without processing or interprete any data intruction
- Computer do less processing in the program flow

- Robot process and interprete all of the data intruction into machine language and execute the instructions.
- Robot do most processing in the program flow

Figure 9 Over the Air Methodology

2) Remote control

In this option, the computer will process and interprets all of the raw data instructions into machine language. The computer will send data for the robot to execute. The robot will not process any data, but only do the execution of the instructions. The robot sends feedback after executing finish. The computer manipulates the robot.



©Cory Thoman * www.ClipartOf.com/1091601

Computer

- Computer process and interprete all the data and control the robot movement
- Computer do most of the processing in the program flow.

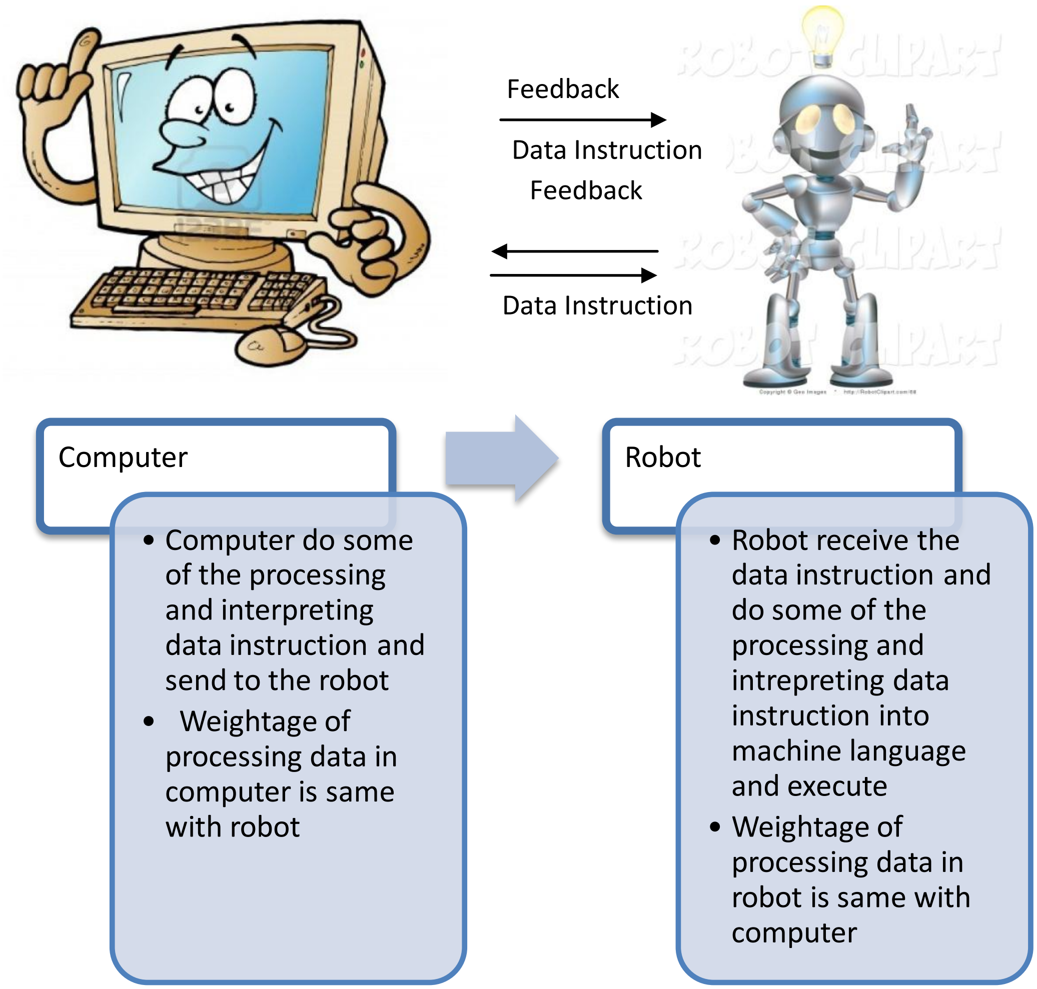
Robot

- Robot execute all the command from the computer
- Robot do less of the processing in the program flow

Figure 10 Remote control methodology

3) Hybrid

Computing and processing level is the same for both computer and robot, which both require to process and interpret the data instruction. This methodology is more on pairing and communicates with each other between computer and robot. The feedback is send to the computer for every data instruction that is sent to the robot.



Computer

- Computer do some of the processing and interpreting data instruction and send to the robot
- Weightage of processing data in computer is same with robot

Robot

- Robot receive the data instruction and do some of the processing and intrepreting data instruction into machine language and execute
- Weightage of processing data in robot is same with computer

Figure 11 Hybrid methodology

Option 3 has been chosen for the methodology of this project based on several advantages. Both computer and robot do the processing, which requires low memory requirement needed inside the robot and complexity in both Turtle Graphic and Turtle robot in terms of computing system and design reduces to medium, thus reduce the duration to finish this project. Besides, with hybrid methodology, more development in the future that can be done in both Turtle Graphic and Turtle robot.

Other than that, the computer and robot establish their own communication link, which processing data and sending feedback is done for every instruction data. Thus, error checking can be done for every data instruction. The chosen technology for the wireless system in this project is Bluetooth, which is suitable for this programming methodology and also with the advantages such as the price of the module over the distance and data transmission capabilities are acceptable.

3.2 Hardware/software design

The hardware and software design is shown as in diagram below. The user enters the logo command in the interface software (Turtle Graphic), which is made inside Python environment. The command will be translated and send to the robot via PC hardware which is Bluetooth. The python software and Bluetooth PC hardware use Microsoft Window operating system. The data transmission is in Bluetooth protocol technology and sends the data to the robot Bluetooth module which is BlueBee module from Cytron Technologies. The data will be translated by the logo interpreter to assort the command in which library the command belong, such as FORWARD, BACKWARD,LEFT and RIGHT. Then, the robot will be executing the library function of the command. The output of the robot is the DC motor and LCD screen, controlled by PIC microcontroller board.

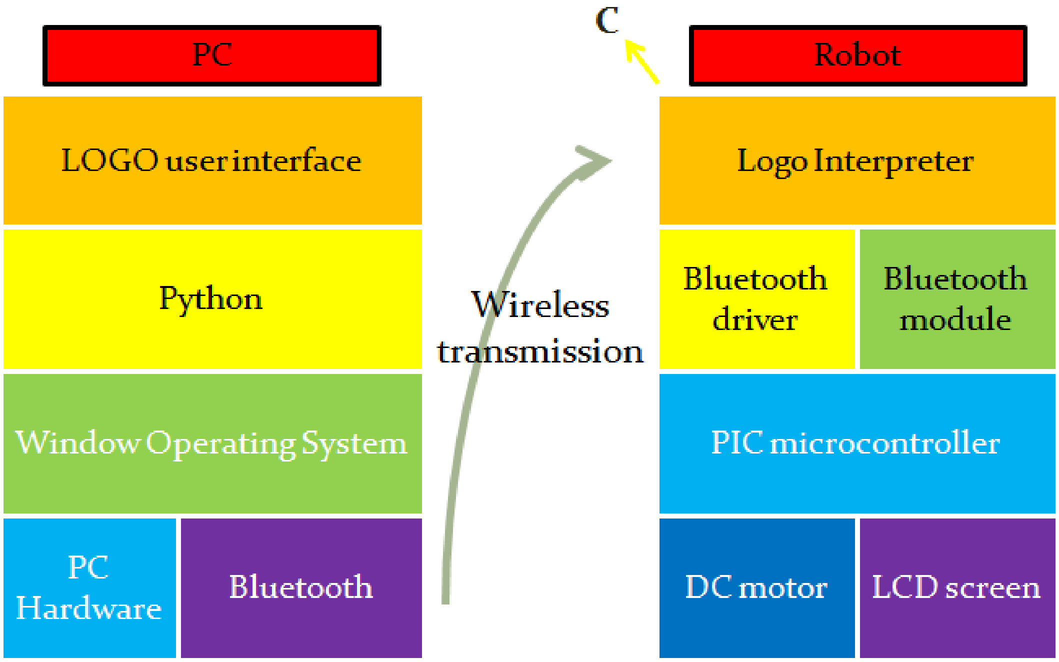


Figure 12 Hardware and software design

3.3 Turtle robot programming process (step-by-step)

This is the programming process as shown in diagram below, which is happening when the logo interpreter is translating the logo command, received from the computer. The logo interpreter will assort the command into their respective library and the robot will execute the command based on the library's function.

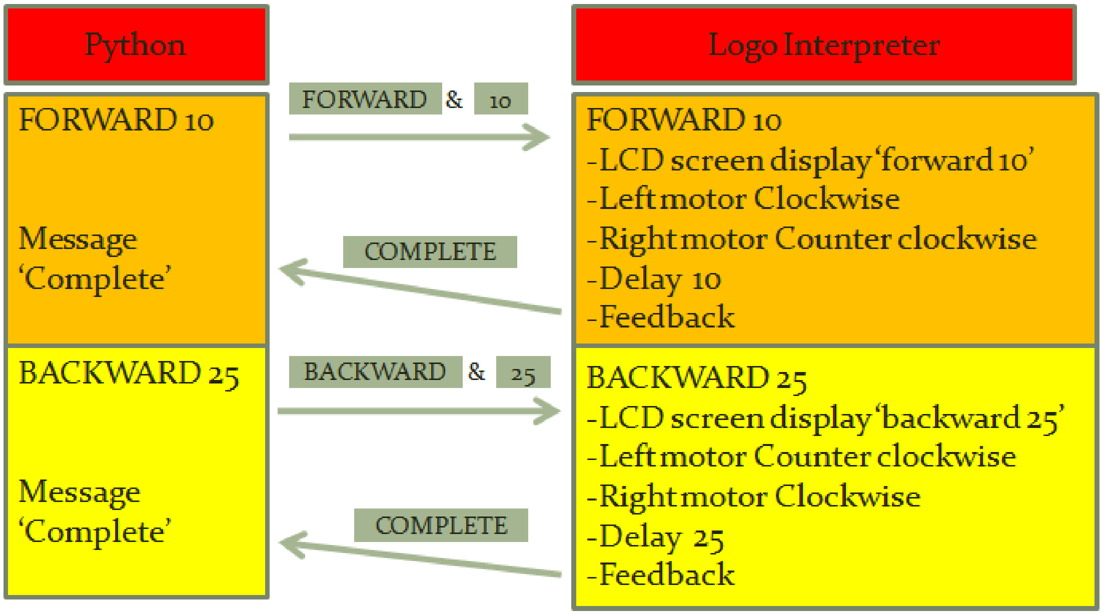


Figure 13 Turtle robot programming process

3.4 Turtle robot programming cycle

This is the overall programming cycle which shown the flow of the command. The first cycle is from the Turtle Graphic, which the command will be entered by the user. The command is in logo language and then transferred to the robot via Bluetooth. The logo interpreter will translate the command into their respective library and the robot executes the function in the library. After done executing, the robot will send feedback to the logo interpreter and the logo interpreter will translate the feedback into message and send back to the Turtle Graphic.

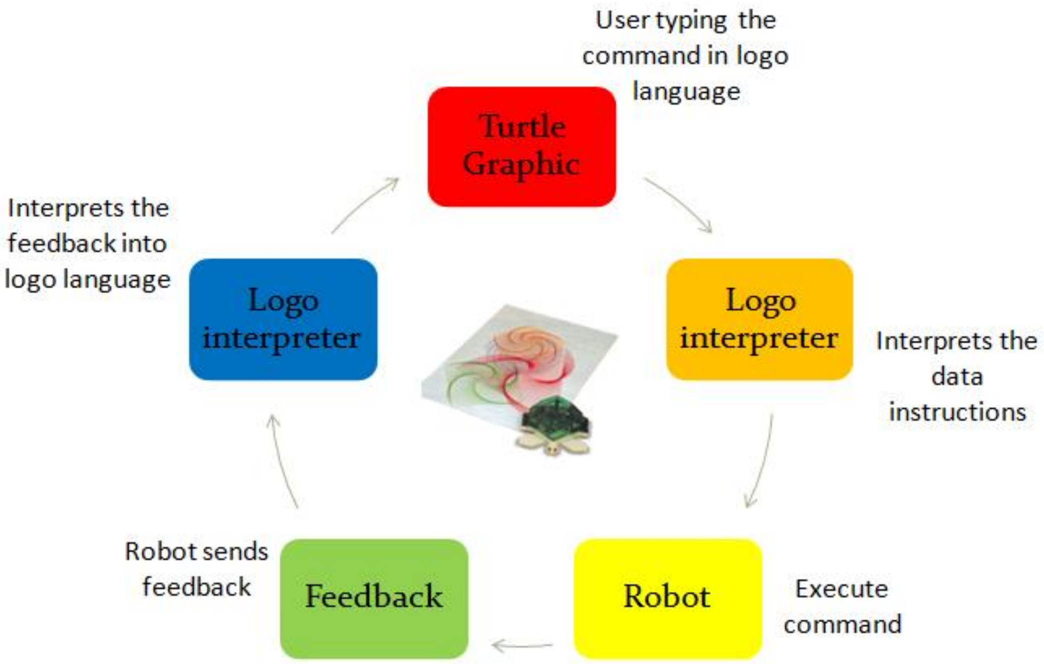

User typing the command in logo language
Turtle Graphic
Interprets the feedback into logo language
Logo interpreter
Logo interpreter
Interprets the data instructions
Robot sends feedback
Feedback
Robot
Execute command

Figure 14 Turtle robot programming cycle

3.5 Key Milestones

The general flow program of this project is done as in figure below. The user need to type in the desired sets of logo's instructions into the computer's terminal which include the path of the robot need to be taken. Then, the logo interpreter will interpret the logo's instructions into language that the robot used (C language). The instruction that have been interpreted will be send to the Turtle Robot processing unit through wireless transmission and the Turtle Robot executes the instructions and generate feedback. The turtle robot interprets the feedback and sends the feedback data to computer.

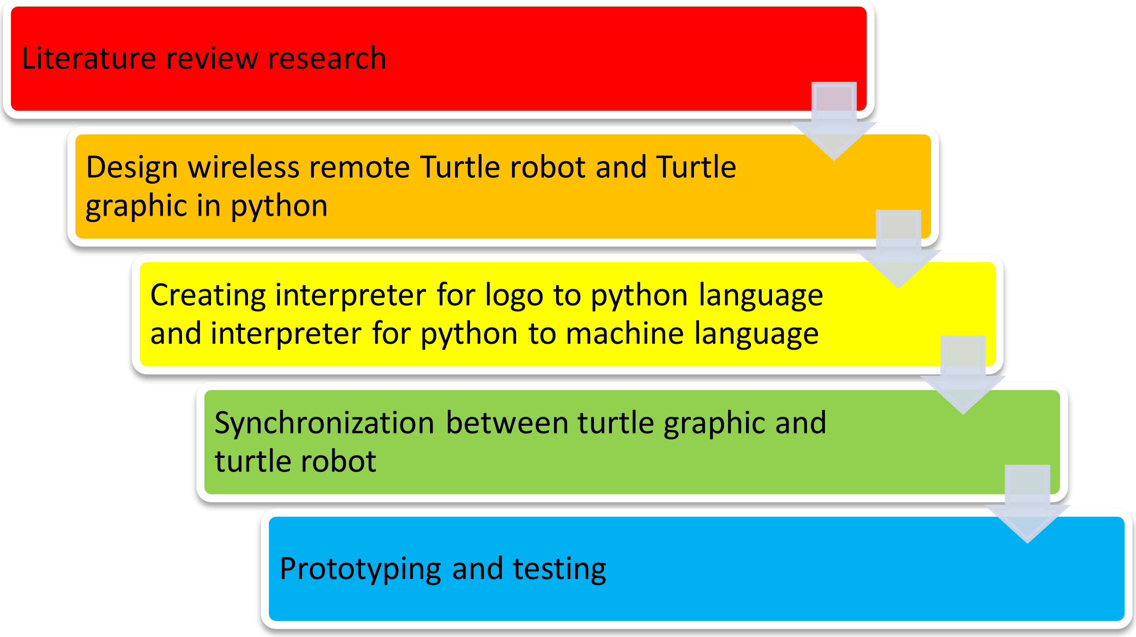

Literature review research
Design wireless remote Turtle robot and Turtle graphic in python
Creating interpreter for logo to python language and interpreter for python to machine language
Synchronization between turtle graphic and turtle robot
Prototyping and testing

Figure 15 Key milestone

3.6 Tools required

This project requires wireless transmission data between computer and in this project. There are some options for the wireless communication module. Bluetooth module is chosen as the transmission data module due to the cost price is lower compare to the Wi-Fi module and distance of transmission is further than RF module.

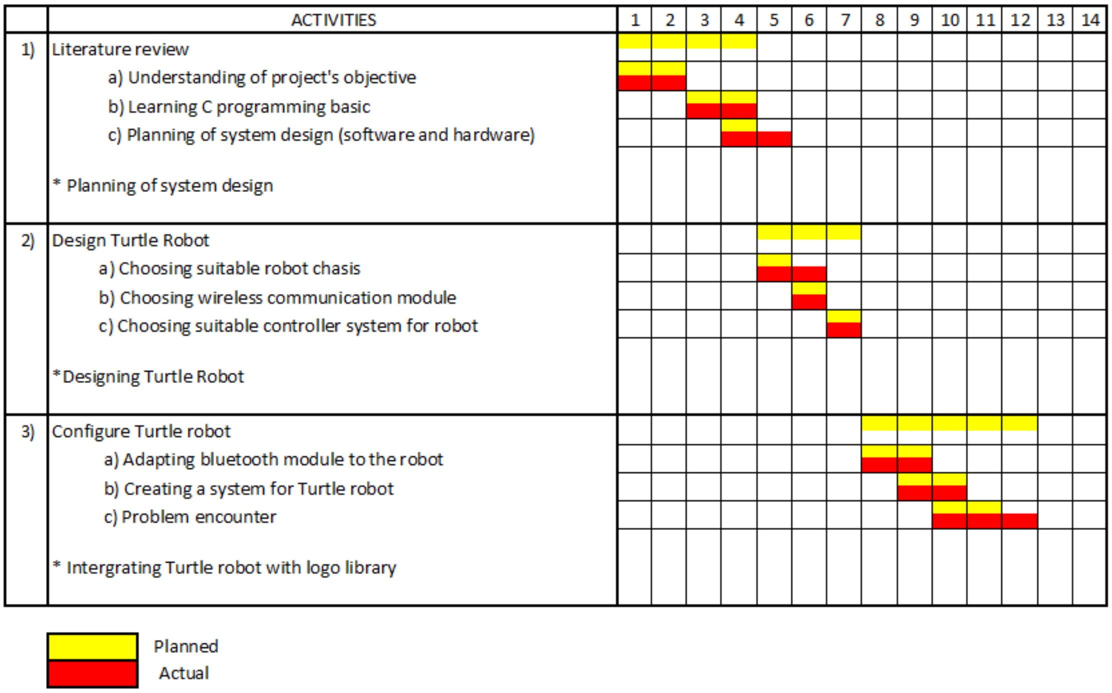
<u>Tools required</u>

1) Microcontroller board and PIC
2) Bluetooth module
3) Simple custom 2-wheel with castor robot
4) PIC Programmer
5) Software: Python v2.3

3.7 Gantt chart

The gantt chart which is used in FYP 1 and FYP 2 is shown as below :

Gantt chart for FYP1



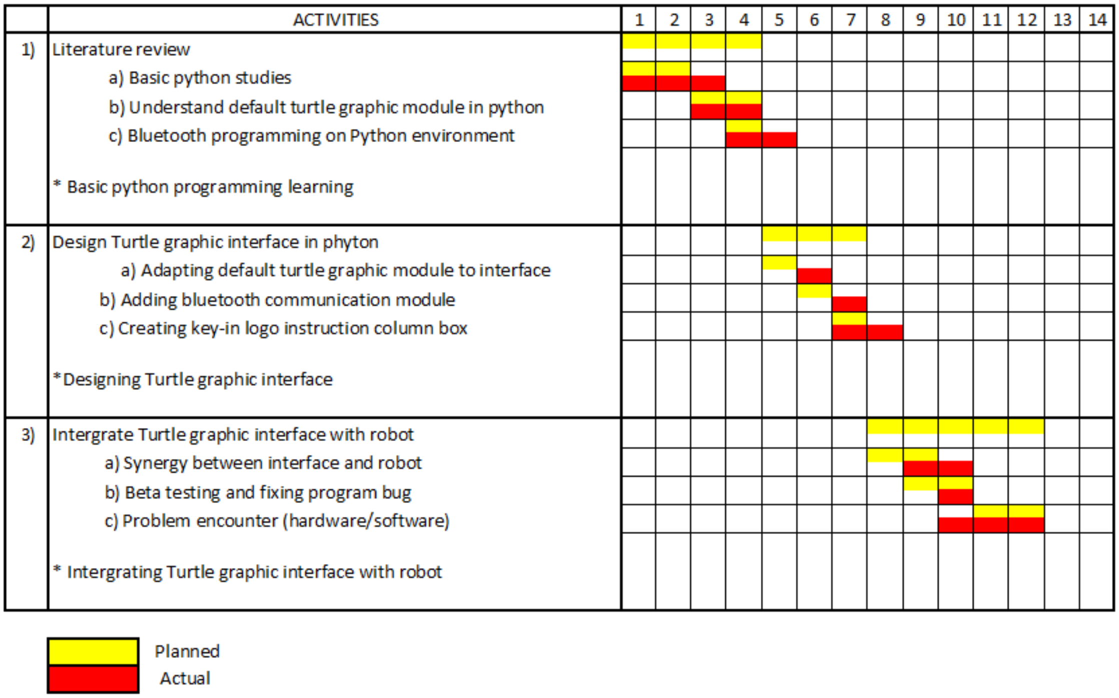
	ACTIVITIES	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1)	Literature review	🟨	🟨	🟨	🟨										
	a) Understanding of project's objective	🟨🟥	🟨🟥												
	b) Learning C programming basic			🟨🟥	🟨🟥										
	c) Planning of system design (software and hardware)				🟨🟥	🟥									
	* Planning of system design														
2)	Design Turtle Robot					🟨	🟨	🟨							
	a) Choosing suitable robot chasis					🟨🟥	🟥								
	b) Choosing wireless communication module						🟨🟥								
	c) Choosing suitable controller system for robot							🟨🟥							
	*Designing Turtle Robot														
3)	Configure Turtle robot								🟨	🟨	🟨	🟨	🟨		
	a) Adapting bluetooth module to the robot								🟨🟥	🟨🟥					
	b) Creating a system for Turtle robot									🟨🟥	🟨🟥				
	c) Problem encounter										🟨🟥	🟨🟥	🟥		
	* Intergrating Turtle robot with logo library														

🟨 Planned

🟥 Actual

Table 1 Gantt Chart for FYP 1

Gantt Chart FYP 2



	ACTIVITIES	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1)	Literature review	🟨	🟨	🟨	🟨										
	a) Basic python studies	🟨 🟥	🟨 🟥	🟥											
	b) Understand default turtle graphic module in python			🟨 🟥	🟨 🟥										
	c) Bluetooth programming on Python environment				🟨 🟥	🟥									
	* Basic python programming learning														
2)	Design Turtle graphic interface in phyton					🟨	🟨	🟨							
	a) Adapting default turtle graphic module to interface					🟨	🟥								
	b) Adding bluetooth communication module						🟨	🟥							
	c) Creating key-in logo instruction column box							🟨 🟥	🟥						
	*Designing Turtle graphic interface														
3)	Intergrate Turtle graphic interface with robot								🟨	🟨	🟨	🟨	🟨		
	a) Synergy between interface and robot								🟨	🟨 🟥	🟥				
	b) Beta testing and fixing program bug									🟨	🟨 🟥				
	c) Problem encounter (hardware/software)										🟥	🟨 🟥	🟨 🟥		
	* Intergrating Turtle graphic interface with robot														

🟨 Planned
🟥 Actual

Table 2 Gantt Chart for FYP 2

Chapter 4

RESULT AND DISCUSSION

Through FYP 1 and FYP 2, there are some results that have been obtained from the project. Basically there are two versions of turtle graphic created for this projects which have different communication approach to the robot. The results from this project can be divided into 3 parts which is:

1) Turtle Robot
2) Turtle Graphic
3) Communication between graphic and robot

4.1 Turtle Robot configuration

Educational mobile robot 2.0 from Cytron[20] Technologies is chosen to be the turtle robot in this project because of its basic functionality which have microcontroller board on it and able to move in 360 degree direction which is necessary characteristic for turtle robot in this projects besides of its elegant design of the robot' chassis. The Bluebee bluetooth module is added to the robot for communication hardware.

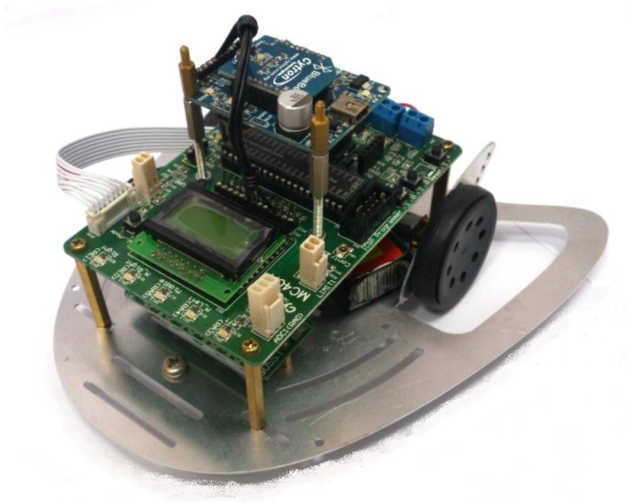
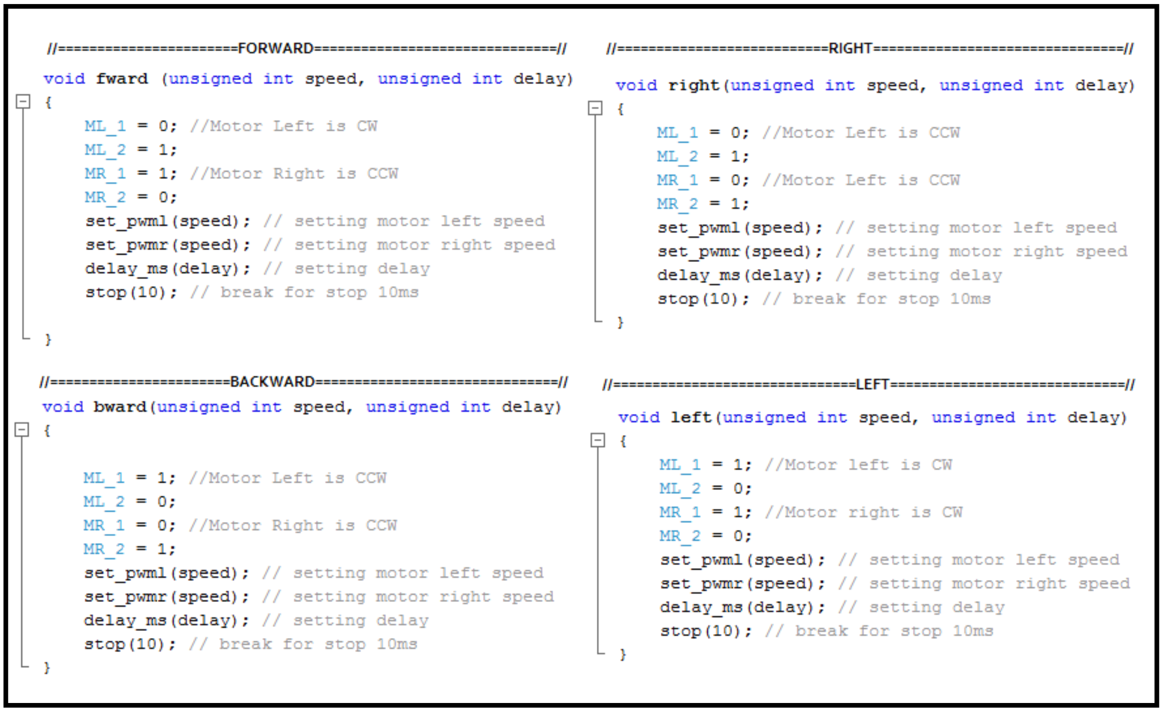


Figure 16 Educational mobile robot 2.0 from Cytron Technologies

The logo interpreter has been successfully created inside the robot during FYP 1. The logo interpreter contains 4 functions which is FORWARD, BACKWARD, LEFT and RIGHT. For every function, groups of specific command are set which are different with other functions, which is required by the robot in order to execute the commands. The input is filtered and assorted by the interpret function, and passed to the respective four functions. The programming code for the four functions is show as below:



```
//======================FORWARD==============================//
void fward (unsigned int speed, unsigned int delay)
{
    ML_1 = 0; //Motor Left is CW
    ML_2 = 1;
    MR_1 = 1; //Motor Right is CCW
    MR_2 = 0;
    set_pwml(speed); // setting motor left speed
    set_pwmr(speed); // setting motor right speed
    delay_ms(delay); // setting delay
    stop(10); // break for stop 10ms

}

//======================BACKWARD==============================//
void bward(unsigned int speed, unsigned int delay)
{

    ML_1 = 1; //Motor Left is CCW
    ML_2 = 0;
    MR_1 = 0; //Motor Right is CCW
    MR_2 = 1;
    set_pwml(speed); // setting motor left speed
    set_pwmr(speed); // setting motor right speed
    delay_ms(delay); // setting delay
    stop(10); // break for stop 10ms
}
```

```
//==========================RIGHT===============================//
void right(unsigned int speed, unsigned int delay)
{
    ML_1 = 0; //Motor Left is CCW
    ML_2 = 1;
    MR_1 = 0; //Motor Left is CCW
    MR_2 = 1;
    set_pwml(speed); // setting motor left speed
    set_pwmr(speed); // setting motor right speed
    delay_ms(delay); // setting delay
    stop(10); // break for stop 10ms
}

//==============================LEFT=============================//
void left(unsigned int speed, unsigned int delay)
{
    ML_1 = 1; //Motor left is CW
    ML_2 = 0;
    MR_1 = 1; //Motor right is CW
    MR_2 = 0;
    set_pwml(speed); // setting motor left speed
    set_pwmr(speed); // setting motor right speed
    delay_ms(delay); // setting delay
    stop(10); // break for stop 10ms
}
```

Figure 17 Logo interpreter function

There are two version of coding for turtle robot, which is Mark 1 Turtle and Mark 2 Turtle. The difference is shown inside the function interpret. The Mark 1 Turtle doesn't have fix delays but have range of delay from 1 to 9 for function forward and backward, while 1 to 8 for function right and left. The value of the delay has been predetermined from the code, such as delay 1 = 1000ms. Two characters needed as for the input. The Finite state machine and state table for interpret function is shown as below:

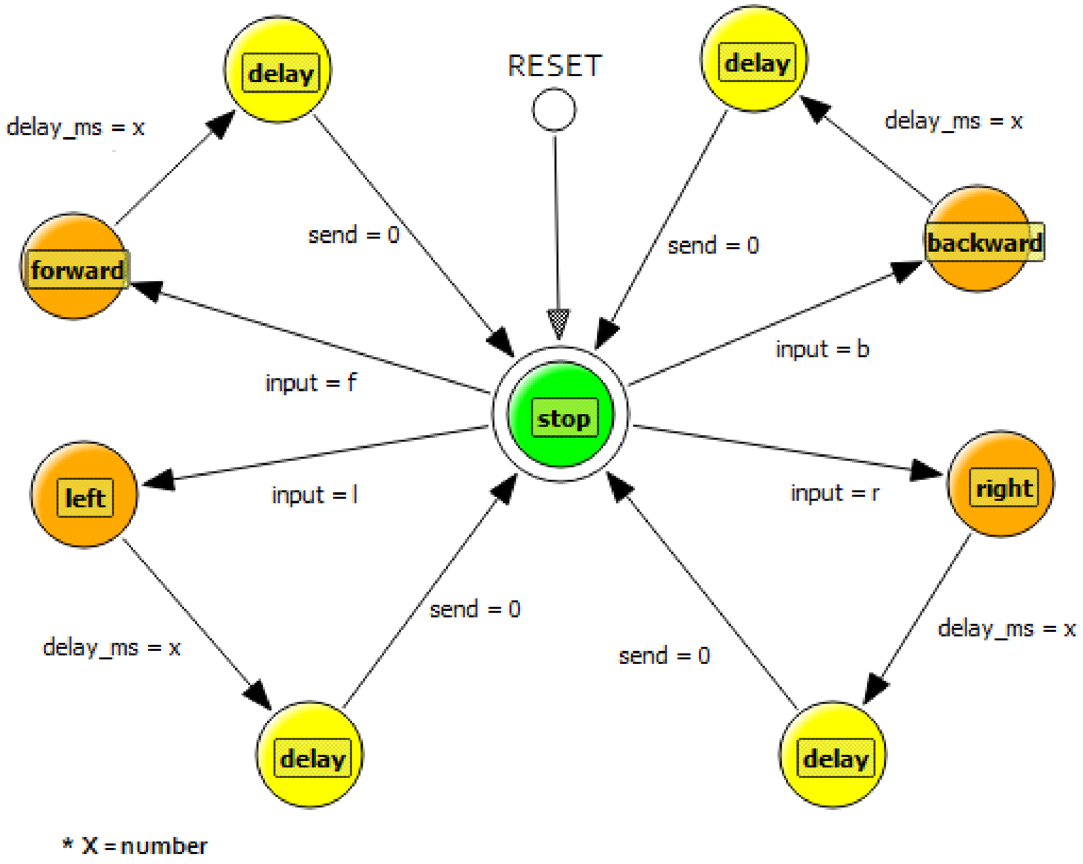


Figure 18 Mark 1 Turtle Finite State Machine

INPUT	PREVIOUS STATE	NEXT STATE	OUTPUT
Reset button	Reset	Stop	Motor Right = 0 Motor.L = 0
f	Stop	forward	Motor Right = CW Motor Left = CCW Display = “forward”
x	Forward	delay	Delay_ms = x
	Delay	Stop	Send = 0
b	stop	Backward	Motor Right= CCW Motor Left = CW Display = “backward”
x	Backward	Delay	Delay_ms = x
	delay	Stop	Send = 0
l	Stop	Left	Motor Right = CCW Motor Left = CCW Display = “left”
x	Left	Delay	Delay_ms = x
	delay	Stop	Send = 0
r	Stop	right	Motor Right = CW Motor Left = CW Display = “right”
x	right	Delay	Delay_ms = x
	Delay	stop	Send = 0

Table 3 Mark 1 Turtle State Table

Mark 2 Turtle coding has fix delays for each function. For function forward and backward, the delay is 1000ms, and for the right and left function, the delay is 5ms. With fix delay, only 1 character for input is needed. The Finite state machine and state table for interpret function is shown as below:

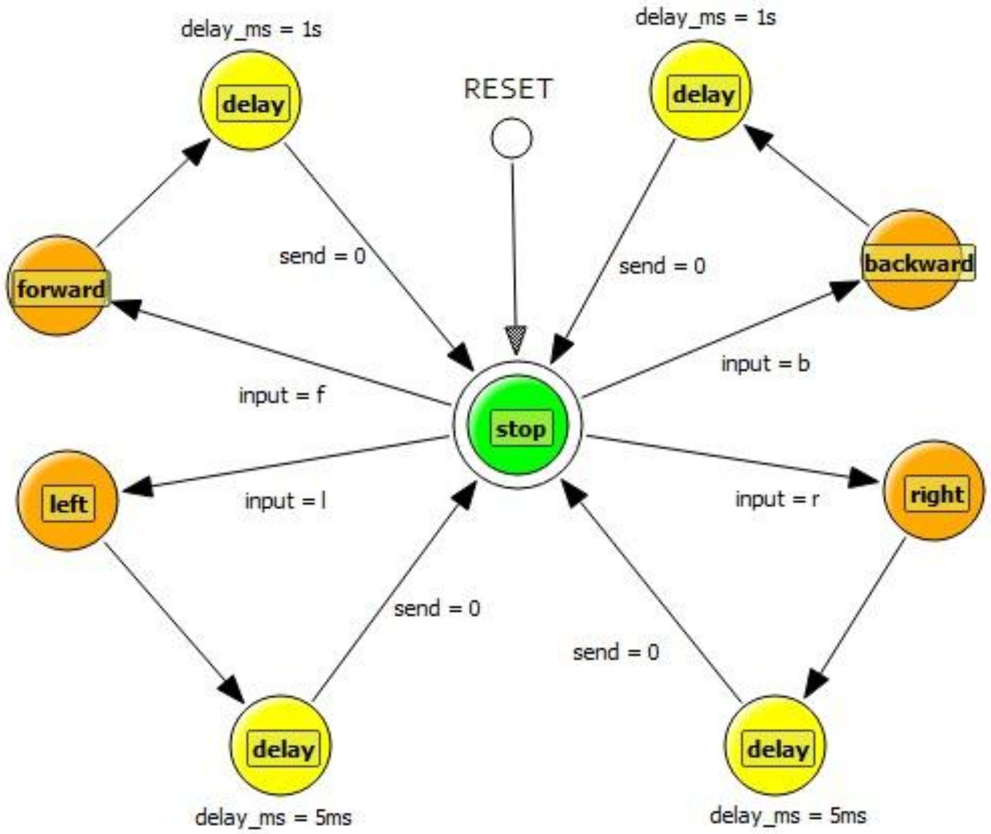


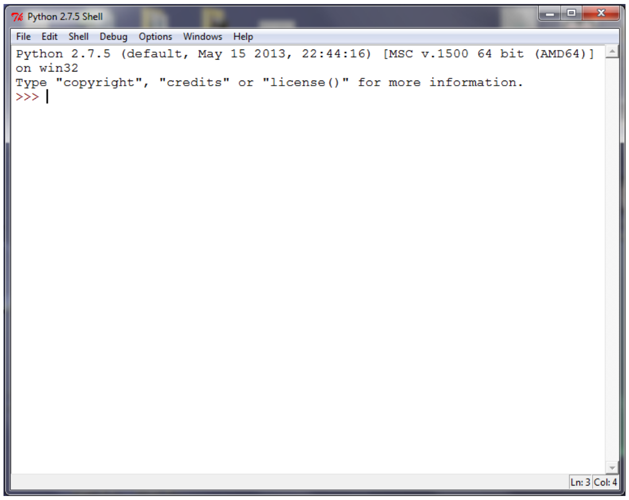
Figure 19 Mark 2 Turtle Finite State Machine

INPUT	PREVIOUS STATE	NEXT STATE	OUTPUT
Reset button	Reset	Stop	Motor Right = 0 Motor.L = 0
f	Stop	forward	Motor Right = CW Motor Left = CCW Display = “forward”
	Forward	delay	Delay_ms = 1000ms
	Delay	Stop	Send = 0
b	stop	Backward	Motor Right= CCW Motor Left = CW Display = “backward”
	Backward	Delay	Delay_ms = 1000ms
	delay	Stop	Send = 0
l	Stop	Left	Motor Right = CCW Motor Left = CCW Display = “left”
	Left	Delay	Delay_ms = 5ms
	delay	Stop	Send = 0
r	Stop	right	Motor Right = CW Motor Left = CW Display = “right”
	right	Delay	Delay_ms = 5ms
	Delay	stop	Send = 0

Table 4 Mark 2 Turtle State Table

4.2 Turtle Graphic Interpreter

Two versions of suitable turtle graphic are created during FYP II (Mark 1 Graphic and Mark 2 Graphic) using Python programming language in Python IDLE 2.7.5 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.



```
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Figure 20 Python IDLE 2.7.5 shell

Module in python is used to provide desirable function in the program. Both versions of turtle graphics used the same library modules which are required to create a functional Turtle Graphic. The modules involved in this turtle graphic are:

1) **re.py**: provides regular expression matching operations[21]. Used to filter input and assorted to the functions assigned

2) **serial.py**: encapsulates the access for the serial port[21]. Used to communicate to the robot using COM3 port by Bluetooth devices

3) **sys.py**: provide access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter[21]. Used to close the program and turtle graphic's window.

4) **turtle.py**: provides turtle graphics primitives, in both object-oriented and procedure-oriented ways[21]. Turtle.py is a basic turtle graphic in python IDLE.

The difference of both version turtle graphic in this project is the way of processing input and delay number. Mark 1 Graphic used delay input which have predetermined values and send the converted input to the robot using Bluetooth device. Two characters are sent as the Bluetooth signal. The finite state machine and the state table are shown as below:

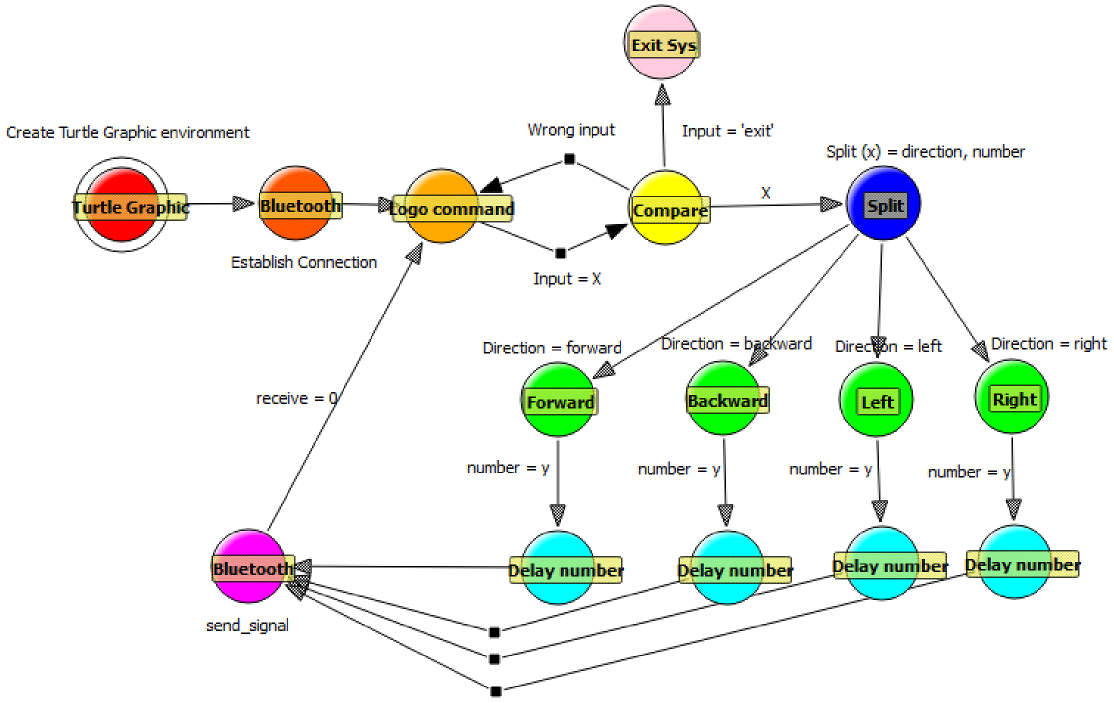


Figure 21 Mark 1 Graphic Finite State Machine

Input	Previous State	Next State	Output
Run program	-	Turtle Graphic	Turtle graphic Environment
-	Turtle Graphic	Bluetooth	Establish Bluetooth connection
-	Bluetooth	Logo command	Ask for input
Input = X	Logo command	Compare	Check input whether correct or wrong
Wrong input	Compare	Logo command	Display = "wrong input!"
Correct input	Compare	Split	Split(x) = direction, number
Direction =forward	Split	Forward	Bluetooth_signal = f
Number =y *y = 1 to 9	Forward	Delay number	Bluetooth_signal = f+y
Direction =backward	Split	Backward	Bluetooth_signal = b
Number =y *y = 1 to 9	Backward	Delay number	Bluetooth_signal = b+y
Direction =left	Split	left	Bluetooth_signal = l
Number =y *y = 1 to 8	left	Delay number	Bluetooth_signal = l+y
Direction =right	Split	right	Bluetooth_signal = r
Number =y *y = 1 to 8	right	Delay number	Bluetooth_signal = r+y
Bluetooth_signal	Delay number	Bluetooth	Send(Bluetooth_signal)
Receive= 0	bluetooth	Logo command	Ask for input

Table 5 Mark 1 Graphic State table

Mark 2 Graphic has flexible delay number due to the usage of loop for sending signal instead of predetermined values for the delay. This version graphic enable the user to key in any value without range. The number of loop is determined by the variable 'number' which is an integer value and every time Bluetooth signal is send, the value of 'number' is decrement by one. One character is send as the Bluetooth signal. The finite state machine and State Table is shown as below:

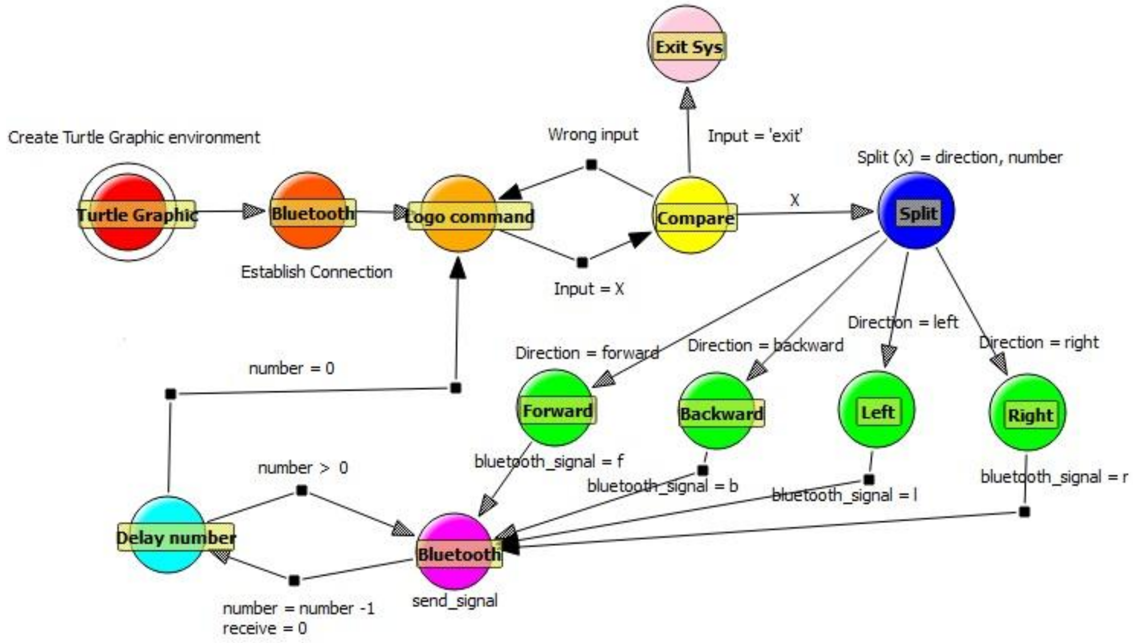


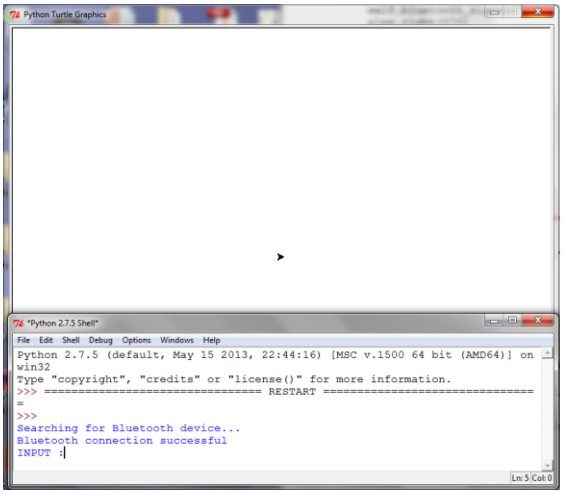
Figure 22 Mark 2 Graphic Finite State Machine

Input	**Previous State**	**Next State**	**Output**
Run program	-	Turtle Graphic	Turtle graphic Environment
-	Turtle Graphic	Bluetooth	Establish Bluetooth connection
-	Bluetooth	Logo command	Ask for input
Input = X	Logo command	Compare	Check input whether correct or wrong
Wrong input	Compare	Logo command	Display = “wrong input!”
Correct input	Compare	Split	Split(x) = direction, number
Direction =forward	Split	Forward	Bluetooth_signal = f
Direction=backward	Split	Backward	Bluetooth_signal = b
Direction =left	Split	left	Bluetooth_signal = l
Direction =right	Split	right	Bluetooth_signal = r
Bluetooth_signal	Forward Backward Left right	Bluetooth	Send(Bluetooth_signal)
Receive= 0	bluetooth	Delay number	number =number-1
number < 0	Delay number	bluetooth	Send(Bluetooth_signal)
Number = 0	Delay number	Logo command	Ask for input

Table 6 Mark 2 Graphic State Table

4.3 Communication between graphic and robot

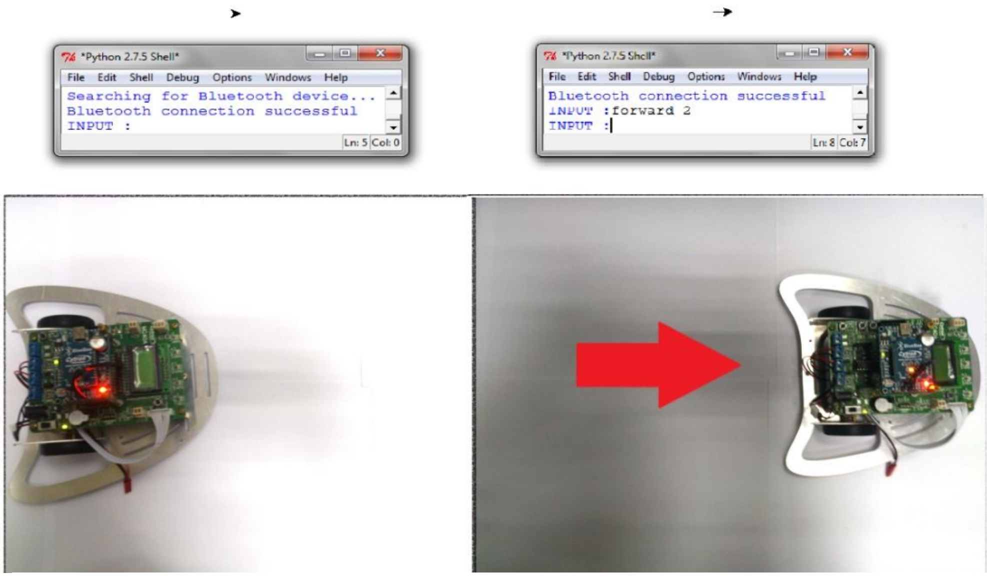
The Mark 1 Turtle and Graphic communicate each other by using series of predetermined signal, direction delay values. The advantage of this version is the execution of input command is smooth due to both variable; direction and delay is passed from graphic to robot. The disadvantage is on having short range of delays (1-9 maximum).



```
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
=
>>>
Searching for Bluetooth device...
Bluetooth connection successful
INPUT :
```

Figure 23 Mark 1 Graphic

The step by step execution for logo command “forward 2” is shown below:

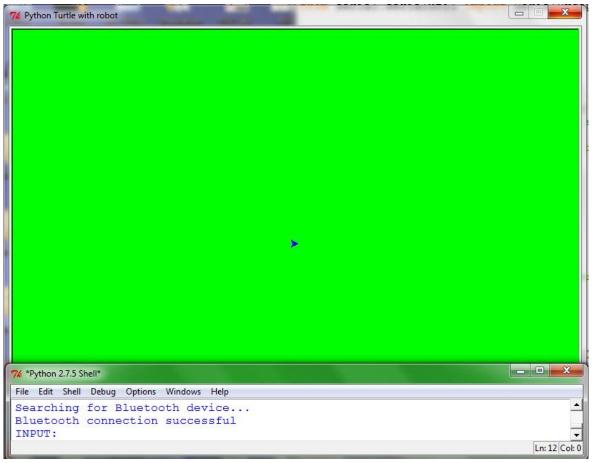

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Ln: 5 Col: 0
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Ln: 8 Col: 7

```
Searching for Bluetooth device...
Bluetooth connection successful
INPUT :
```

```
Bluetooth connection successful
INPUT :forward 2
INPUT :
```

Figure 24 Step by step execution Mark 1

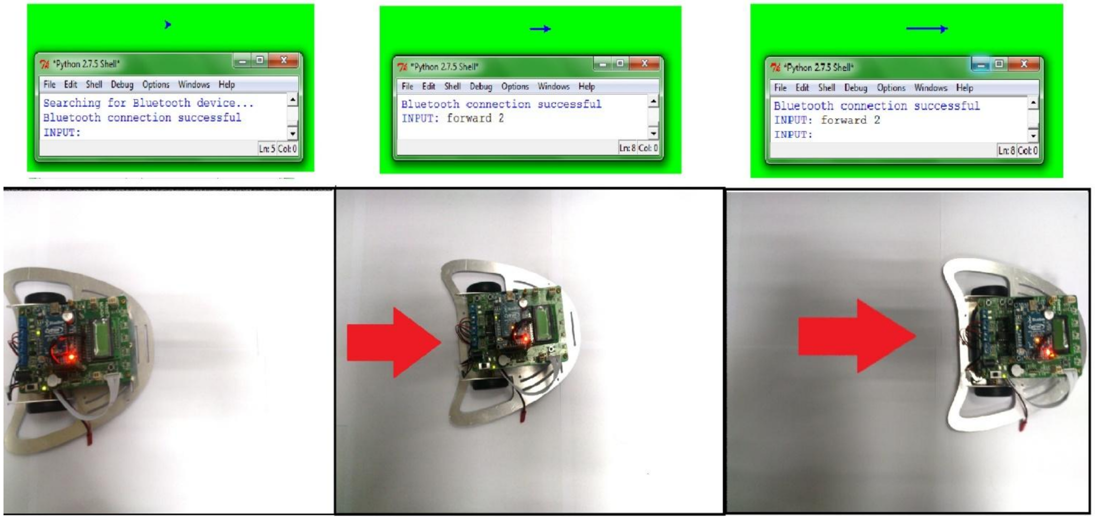
The Mark 2 Turtle and Graphic communicate between each other using predetermined direction and looping for delay. The advantage of this version is that the range of delay is flexible and has no fix range due to the only variable; direction is passed to the robot while the variable number is used as number of loop for sending signal to Bluetooth. The disadvantage of this version is on execution of input command, which is less smooth due to the robot stop each time at the starting of the loop.



```
Searching for Bluetooth device...
Bluetooth connection successful
INPUT:
```

Figure 25 Mark 2 graphic

The step by step execution for logo command "forward 2" is shown below:


Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Ln: 5 Col: 0
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Ln: 8 Col: 0
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Ln: 8 Col: 0

```
Searching for Bluetooth device...
Bluetooth connection successful
INPUT:
```

```
Bluetooth connection successful
INPUT: forward 2
```

```
Bluetooth connection successful
INPUT: forward 2
INPUT:
```

Figure 26 Step by step execution Mark 2

4.4 Chosen turtle graphic interpreter and command list

Mark 1 interpreter is chosen as Turtle Graphic Interpreter in this project based on the advantages of smooth execution and reliable system. A list of logo command is created based on this interpreter. The list is shown as below:

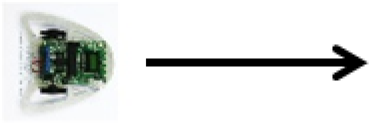
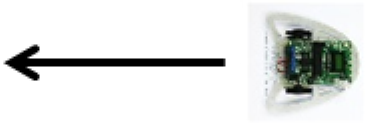
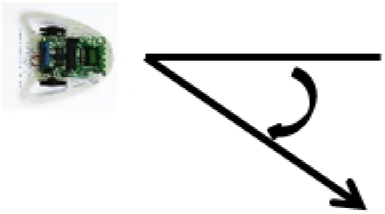
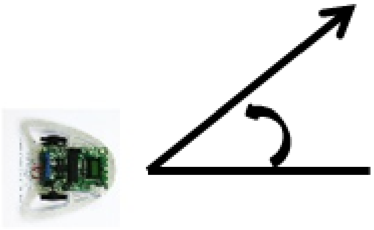
Logo command	Execution of robot
1) forward X i) Robot run forward for a certain time ii) Total time = X * 1sec iii) The number range of X is from 1-9	
2) backward X i) Robot run backward for a certain time ii) Total time = X * 1sec iii) The number range of X is from 1-9	
3) right X i) Robot turn right for a certain angle ii) Total angle = X * 45° iii) The number range of X is from 1-8	
4) left X i) Robot turn left for a certain angle ii) Total angle = X *45° iii) The number range of X is from 1-8	

Table 7 Logo programming command list

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Both version of turtle graphic, Mark 1 and Mark 2 achieve the objective of this project which is to create a simple interactive logo interpreter, capable of processing logo instructions from the user and sending data instruction to turtle robot wirelessly. With this Turtle graphic, children can learn programming language using logo command. The graphic is user-friendly as the user can insert and run the input from standalone graphic, instead in python shell.

In view of advantages and disadvantages of both versions, Mark 1 is more favorable than Mark 2 due to speed of executing input by the robot. But the range of delay number is limited, which the effect can be seen when turning right and left. Due to limited delay range, each delay numbers for right and left is only for 45 degree worth step. Instead of using sensors, both interpreter use delay as distance and angle measurement which is not very accurate but acceptable, as the measurement can be vary depending on the surface where Turtle Robot is on.

5.2 Recommendation and future work

There are two points of view for recommendation and future work for this project which is an applicable upgrade for this turtle logo. From the view of hardware (turtle robot), using timing delay as a measurement of distance and angle is not very accurate and hard to implement. However, using sensors such as encoder, line following sensor, range finder can be an upgrade for the Turtle Robot for accurate movement and turning. Other than that, a marker can be put to the robot so that it can actually draw if it is move on paper. With marker, turtle logo can be more interesting to the children as they can draw something instead of looking at the graphics from the software.

From the view of software (turtle graphic), the graphical interface can be enhance by putting a lot of examples and picture for the children to draw and also tutorial for the beginners. The graphical interface can be upgrade into more user-friendly by adding cool turtle picture instead of cursor and more functional button to the interface such as help button, tutorial button and others. Besides that, instead of 4 logo command, more complex command can be develop by putting conditional command (if else) and loop (while) for more challenging learning for children.

REFERENCES

1. Rachum, R. *Python Turtle*. 1986 [cited 2013 25/6/2013]; Available from: http://pythonturtle.org/screenshot.gif.
2. AUVAC. *SAUVIM configuration*. AUV System Spec Sheet [cited 2013 25/6/2013]; Available from: http://auvac.org/uploads/configuration/SAUVIM%202008%20[Hanging].jpg.
3. Eveland, E. *win32 F/OSS development - Logo*. 12/3/2008 [cited 2013 25/6/2013]; Available from: http://evelands.net/evan/fossDevel_logo.php.
4. Fulcher, J.A., *Fun and games and microcomputer interfacing (laboratory exercises)*. Micro, IEEE, 1991. **11**(1): p. 18-21.
5. Sheehan, R. *Lower floor, lower ceiling: easily programming turtle-graphics*. in *Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on*. 2000.
6. Demo, G.B., G. Marciano, and S. Siega. *Concrete Programming: Using Small Robots in Primary Schools*. in *Advanced Learning Technologies, 2008. ICALT '08. Eighth IEEE International Conference on*. 2008.
7. Turbak, F., et al. *Blocks languages for creating tangible artifacts*. in *Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on*. 2012.
8. Wurman, R.S., et al., *Information anxiety 2*. 2001: Que. 308.
9. Bowker, R.R., *American Men and Women of Science: The physical and biological sciences*. 20th ed. 1998: R.R. Bowker.
10. Papert, S., *MINDSTORM : Childern, Computers, and Powerful Ideas*. 1980, New York: Basic Books, Inc., Publishers. 230.
11. Dictionary.com, in *Collins English Dictionary - Complete & Unabridged*, HarperCollins Publishers.
12. Dictionary.com, in *Online Etymology Dictionary*, D. Harper, Editor: Historian.
13. Harvey, B., *Symbolic Computing*. Computer Science Logo Style. Vol. 1. 1997. xv-xvi.
14. Harvey, B., *Beyond Programming*. Computer Science Logo Style. Vol. 3. 1997.
15. Kim, T.W. and J. Yuh. *Task description language for underwater robots*. in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. 2003.
16. Biswas, J. and M. Veloso. *WiFi localization and navigation for autonomous indoor mobile robots*. in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010.
17. Yeong Che Fai, S.H.M.A., Norsheila bt Fisal, J. Abu Bakar, *Bluetooth Enabled Mobile Robot*, in *Faculty of Electrical Engineering*2002, Universiti Teknologi Malaysia.
18. Owayjan, M. and A.A. Ammo. *Design and development of a hybrid feedback control system for an RF remote-controlled robot*. in *Advances in Computational Tools for Engineering Applications, 2009. ACTEA '09. International Conference on*. 2009.

19. 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.

20. Technologies, C., *Educational Mobile Robot 2.0*.

21. *The Python Standard Library*, 1990, Python Software Foundation.

APPENDICES

APPENDIX A

TURTLE ROBOT LIBRARY FUNCTION C CODE

```
void interpret (char a[2])
{
 switch(a[0])// filtering first array element for forward, backward, left and right
 {
   case 'f' : lcd_putstr("fwd");
         switch(a[1])// filter the delay number
         {
          case '1' : lcd_goto(0x04);lcd_putstr("1");fward(255,1000);uart_putstr("0");break;
          case '2' : lcd_goto(0x04);lcd_putstr("2");fward(225,2000);uart_putstr("0");break;
          case '3' : lcd_goto(0x04);lcd_putstr("3");fward(225,3000);uart_putstr("0");break;
          case '4' : lcd_goto(0x04);lcd_putstr("4");fward(225,4000);uart_putstr("0");break;
          case '5' : lcd_goto(0x04);lcd_putstr("5");fward(225,5000);uart_putstr("0");break;
          case '6' : lcd_goto(0x04);lcd_putstr("6");fward(225,6000);uart_putstr("0");break;
          case '7' : lcd_goto(0x04);lcd_putstr("7");fward(225,7000);uart_putstr("0");break;
          case '8' : lcd_goto(0x04);lcd_putstr("8");fward(225,8000);uart_putstr("0");break;
          case '9' : lcd_goto(0x04);lcd_putstr("9");fward(225,9000);uart_putstr("0");break;
         }break;

    case 'b' : lcd_putstr("bwd");
          switch(a[1])// filter the delay number
          {
          case '1' : lcd_goto(0x04);lcd_putstr("1");bward(255,1000);uart_putstr("0");break;
          case '2' : lcd_goto(0x04);lcd_putstr("2");bward(225,2000);uart_putstr("0");break;
          case '3' : lcd_goto(0x04);lcd_putstr("3");bward(225,3000);uart_putstr("0");break;
          case '4' : lcd_goto(0x04);lcd_putstr("4");bward(225,4000);uart_putstr("0");break;
          case '5' : lcd_goto(0x04);lcd_putstr("5");bward(225,5000);uart_putstr("0");break;
          case '6' : lcd_goto(0x04);lcd_putstr("6");bward(225,6000);uart_putstr("0");break;
          case '7' : lcd_goto(0x04);lcd_putstr("7");bward(225,7000);uart_putstr("0");break;
          case '8' : lcd_goto(0x04);lcd_putstr("8");bward(225,8000);uart_putstr("0");break;
          case '9' : lcd_goto(0x04);lcd_putstr("9");bward(225,9000);uart_putstr("0");break;
         }break;

     case 'r': lcd_putstr("rght");
          switch(a[1])// filter the delay number
          {
          case '1' : lcd_goto(0x05);lcd_putstr("1");right(250,225);uart_putstr("0");break;
          case '2' : lcd_goto(0x05);lcd_putstr("2");right(225,450);uart_putstr("0");break;
          case '3' : lcd_goto(0x05);lcd_putstr("3");right(225,675);uart_putstr("0");break;
          case '4' : lcd_goto(0x05);lcd_putstr("4");right(225,900);uart_putstr("0");break;
          case '5' : lcd_goto(0x05);lcd_putstr("5");right(225,1125);uart_putstr("0");break;
          case '6' : lcd_goto(0x05);lcd_putstr("6");right(225,1350);uart_putstr("0");break;
          case '7' : lcd_goto(0x05);lcd_putstr("7");right(225,1575);uart_putstr("0");break;
          case '8' : lcd_goto(0x05);lcd_putstr("8");right(225,1800);uart_putstr("0");break;
          }break;
    case 'l': lcd_putstr("left");
         switch(a[1])// filter the delay number
         {
          case '1' :lcd_goto(0x05);lcd_putstr("1");left(250,255);uart_putstr("0");break;
          case '2' :lcd_goto(0x05);lcd_putstr("2");left(225,450);uart_putstr("0");break;
          case '3' :lcd_goto(0x05);lcd_putstr("3");left(225,675);uart_putstr("0");break;
          case '4' :lcd_goto(0x05);lcd_putstr("4");left(225,900);uart_putstr("0");break;
          case '5' :lcd_goto(0x05);lcd_putstr("5");left(225,1125);uart_putstr("0");break;
          case '6' :lcd_goto(0x05);lcd_putstr("6");left(225,1350);uart_putstr("0");break;
          case '7' :lcd_goto(0x05);lcd_putstr("7");left(225,1575);uart_putstr("0");break;
          case '8' :lcd_goto(0x05);lcd_putstr("8");left(225,1800);uart_putstr("0");break;
         }break;
 }
}
```

APPENDIX B

Turtle Graphic Interpreter Pyhton Code

```
import os
import re
import sys
import serial
from serial.serialutil import SerialException
from PyQt4 import QtGui
from PyQt4 import QtCore
import turtle
from turtle import reset, fill, tracer, speed, forward, left, backward, right

class bluetooth(object):

    def __init__(self):
        self.serial = None
        try:
            self.serial = serial.Serial('COM3', 9600)
            print "Bluetooth connection successful"
        except SerialException, e:
            print "Bluetooth connection fail, try again"
            sys.exit()

    def send_cmd(self,blue_in):
        self.blue_in = blue_in
        if self.serial:
            self.serial.flushOutput()
            send_str = self.blue_in
            self.serial.write(send_str)
            print "Sending: " + send_str
        else:
            print "Can't send without serial connection"


class test(object):

    def __init__(self, input_string_test):
        self.input_string_test = input_string_test

    def test_string(self):
        if re.match('(\w+\s\d)', self.input_string_test):
            return self.input_string_test
        elif self.input_string_test == "exit":
            print "exit"
            sys.exit()
        else:
            print "wrong command input!"
            self.input_string_test = "1"
            return self.input_string_test

class interprete(object):

    def __init__(self,true_input):
        self.true_input = true_input
        self.direction,self.number = self.true_input.split()
        if self.direction == "forward":
            self.forward()
        elif self.direction == "backward":
            self.backward()
    elif self.direction == "left":
        self.left()
    elif self.direction == "right":
        self.right()
    else :
        self.false_direction()

def false_direction(self):
    print "wrong direction command!"
    self.true_input = "1"
    return self.true_input

def false_number(self):
    print "value is not in range!"
    self.true_input = "1"
    return self.true_input

def forward(self):
    if self.number == "1":
        self.bluetooth_signal('f1')
        alex.forward(10)
    elif self.number == "2":
        self.bluetooth_signal("f2")
        alex.forward(20)
    elif self.number == "3":
        self.bluetooth_signal("f3")
        alex.forward(30)
    elif self.number == "4":
        self.bluetooth_signal("f4")
        alex.forward(40)
    elif self.number == "5":
        self.bluetooth_signal("f5")
        alex.forward(50)
    elif self.number == "6":
        self.bluetooth_signal("f6")
        alex.forward(60)
    elif self.number == "7":
        self.bluetooth_signal("f7")
        alex.forward(70)
    elif self.number == "8":
        self.bluetooth_signal("f8")
        alex.forward(80)
    elif self.number == "9":
        self.bluetooth_signal("f9")
        alex.forward(90)
    else :
        self.false_number()

def backward(self):
    if self.number == "1":
        self.bluetooth_signal("b1")
        alex.backward(10)
    elif self.number == "2":
        self.bluetooth_signal("b2")
        alex.backward(20)
    elif self.number == "3":
        self.bluetooth_signal("b3")
        alex.backward(30)
    elif self.number == "4":
        self.bluetooth_signal("b4")
        alex.backward(40)
    elif self.number == "5":
        self.bluetooth_signal("b5")
```

```
            alex.backward(50)
        elif self.number == "6":
            self.bluetooth_signal("b6")
            alex.backward(60)
        elif self.number == "7":
            self.bluetooth_signal("b7")
            alex.backward(70)
        elif self.number == "8":
            self.bluetooth_signal("b8")
            alex.backward(80)
        elif self.number == "9":
            self.bluetooth_signal("b9")
            alex.backward(90)
        else :
            self.false_number()

    def left(self):
        if self.number == "1":
            self.bluetooth_signal("l1")
            alex.left(45)
        elif self.number == "2":
            self.bluetooth_signal("l2")
            alex.left(90)
        elif self.number == "3":
            self.bluetooth_signal("l3")
            alex.left(135)
        elif self.number == "4":
            self.bluetooth_signal("l4")
            alex.left(180)
        elif self.number == "5":
            self.bluetooth_signal("l5")
            alex.left(225)
        elif self.number == "6":
            self.bluetooth_signal("l6")
            alex.left(270)
        elif self.number == "7":
            self.bluetooth_signal("l7")
            alex.left(315)
        elif self.number == "8":
            self.bluetooth_signal("l8")
            alex.left(360)
        else :
            self.false_number()

    def right(self):
        if self.number == "1":
            self.bluetooth_signal("r1")
            alex.right(45)
        elif self.number == "2":
            self.bluetooth_signal("r2")
            alex.right(90)
        elif self.number == "3":
            self.bluetooth_signal("r3")
            alex.right(135)
        elif self.number == "4":
            self.bluetooth_signal("r4")
            alex.right(180)
        elif self.number == "5":
            self.bluetooth_signal("r5")
            alex.right(225)
        elif self.number == "6":
            self.bluetooth_signal("r6")
            alex.right(270)
        elif self.number == "7":
            self.bluetooth_signal("r7")
            alex.right(315)
        elif self.number == "8":
            self.bluetooth_signal("r8")
            alex.right(360)
        else :
            self.false_number()

    def bluetooth_signal(self,signal):
        self.true_input = signal
        return self.true_input



#================== MAIN STARTS HERE===================#
wn = turtle.Screen()
alex = turtle.Turtle()
print "Searching for Bluetooth device..."
bluetooth_protocol = bluetooth()
loop = 1
while loop == 1:

    blessed_input = "1"
    while blessed_input == "1":
        input_string = raw_input("Enter command :")
        input_user = test(input_string)
        blessed_input = input_user.test_string()

    bluetooth_input = interprete(blessed_input).true_input
    if bluetooth_input != "1":
       print bluetooth_input
       bluetooth_protocol.send_cmd(bluetooth_input)
```

APPENDIX C

Bluetooth module specifications

Specifications

- Bluetooth chip: CSR BC04 Chipset
- Bluetooth protocol: Bluetooth Specification v2.0 + EDR
- Operating frequency: 2.4 ~ 2.48GHz unlicensed ISM band
- Modulation: GFSK (Gaussian Frequency Shift Keying)
- Transmit Power: ≤ 4dBm, Class 2
- Transmission distance: 20 ~ 30m in free space
- Sensitivity: ≤-84dBm at 0.1% BER
- Transfer rate: Asynchronous: 2.1Mbps (Max) / 160 kbps; Synchronous: 1Mbps/1Mbps
- Safety features: Authentication and encryption
- Support profiles: Bluetooth serial port
- Serial port settings: 1200 ~ 1382400 / N / 8 / 1
- Baud rate default: 9600 bps(Serial Port Profile, transparent mode)
- Baud rate default: 38400 bps in AT mode.
- Pair Number/ID: 1234
- Input Voltage: +3.3 DC/50mA
- Operating temperature: -20 ℃ ~ +55 ℃
- Module Size: 32 × 24 × 9mm