

LEADER-FOLLOWING USING E-PUCK ROBOTS

by

YAN HUANG

URN: 6219669

A dissertation submitted in partial fulfilment of the
requirements for the award of

MASTER OF SCIENCE IN INTERNET COMPUTING

August 2013

Department of Computing
University of Surrey
Guildford GU2 7XH

Supervised by: Yaochu Jin

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Yan Huang
August 2013

© Copyright Yan Huang, August 2013

Abstract

Multi-robot systems have attracted increasing attention due to the increasing demand for replacing the humans with intelligent robots, especial in inhospitable environments. However, the capability of one single robot is limited and vulnerable to disturbances and failures. Multi-robot systems are therefore preferred because they can accomplish complex tasks collectively, and be less sensitive to system failures. Leader-following and avoiding obstacle are two fundamental functions of multi-robot systems. This project implements two typical behaviours of swarm robots. Webots is used as simulator software and MPLAB IDE compilation. The hardware platform of the experiments is e-puck. Several methods for system implementation are discussed, including their advantages and disadvantages. Our simulation and experimental results show that the system is robust, scalable and modular, which enable e-pucks to form a simple multi-robot system.

Acknowledgements

I am pleasure to acknowledge the support I obtained during the project. I wish to express my deepest appreciation to my supervisor Professor Yaochu Jin. His continuous support helped me finish the project. I am also extremely grateful to Dr. Hyondong Oh who shared his experience on e-puck and gave me a lot of assistance. Finally, I want to thank my wife for her help on formatting my thesis.

Contents

1	Introduction	13
1.1	Aim of the Project	15
1.2	Testing Bed	15
1.3	Stages of the Project	16
1.4	Limitations	17
1.5	Structure of the Report	17
2	Literature Review	18
2.1	Multi-robot System	18
2.2	Wheeled Mobile Robot	19
2.3	The Hardware Platform: E-puck	19
2.3.1	The Features of E-puck	20
2.3.2	Microcontroller	21
2.3.3	Sensors and Actuators	22
2.4	Differential Drive Kinematics	27
2.5	Odometry	29
2.6	Communication	30
2.7	Software	31
2.7.1	Webots	31

2.7.2	MPLAB IDE and Bootloader	33
3	Implementation	39
3.1	The Aims of Projects	39
3.2	The Abilities of the Devices of E-puck	40
3.2.1	Stepper Motors	40
3.2.2	IR Sensors	41
3.2.3	Accelerometer	44
3.2.4	Camera	44
3.3	Design of the Modules	45
3.3.1	Obstacle Avoidance	46
3.3.1.1	Method I	46
3.3.1.2	Method II	48
3.3.1.3	The Chosen Method	50
3.3.2	Leader-Following	51
3.3.2.1	Method I	51
3.3.2.2	Method II	52
3.3.2.3	The Chosen Method	53
3.3.3	Searching	54
3.3.4	The Integration of System	55
4	Experiment and Result	58
5	Conclusion and Future Work	64
5.1	Challenges of the Project	64
5.2	Strengths and Weaknesses of the System	65
5.3	Future Work	66

5.4	Conclusion	67
	Bibliography	68

List of Figures

1.1	Vacuum cleaner robot	14
1.2	SLAM robot	15
2.1	The overview of e-puck(Mondada 2009)	20
2.2	The outline of the electronic of the e-puck(Mondada 2009)	22
2.3	Sensors, LEDs and camera	23
2.4	The values of proximity sensors and distance	24
2.5	The light sensors and distance	25
2.6	Microphones	26
2.7	Differential drive kinematics	28
2.8	An example of odometry	30
2.9	Webots GUI	32
2.10	The e-puck control window	33
2.11	The interface of MPLAB IDE	34
2.12	Create a new project	35
2.13	Configuration settings	36
2.14	Compile project	36
2.15	Add e-puck into Bluetooth devices	37
2.16	E-puck's ID	38

2.17	The interface of tiny Bootloader.	38
3.1	Stepper motors of e-puck	41
3.2	Light sensors	42
3.3	Proximity sensors	43
3.4	Message sent by IR	43
3.5	Tracking line behaviour	44
3.6	Leader-following behaviour	45
3.7	Comparison of grabbed image between simulation and experiment	54
3.8	Flow chart	57
4.1	The selector position	59
4.2	Experiment of leader-following	60
4.3	Experiment of avoiding obstacle and navigation	61
4.4	Experiment of chain recovery – move follower	62
4.5	Experiment of chain recovery – move leader	63
5.1	The basic method to synchronisation	67

List of Tables

2.1	Features of the e-puck(Cyberbotics' Robot Curriculum 2009)	21
2.2	The place and orientations of IR and camera (Cyberbotics 2013)	24
3.1	IR sensors' orientations (rad)	48
4.1	Final parameters' setting	59

Glossary

$PS(i)$	The value of i^{th} proximity sensor
S_L	The speed of left wheel
S_R	The speed of right wheel
$PS_OFFSET(i)$	The noise of i^{th} proximity sensors when no obstacle around
$sensorDiecction(i)$	The orientation of i^{th} sensor
T	The height of the top red pixel
M	The average width of red pixels
N	The number of red pixels
W	The width of the camera
H	The height of the camera
R	The radius of the curve
l	The length between the two wheels
ω	The rate of rotation
D	The distance from the message sender to receiver
α	The direction from the message sender to receiver
T_D	The threshold of the change of distance to ignore noise
T_α	The threshold of the change of direction to ignore noise
$spiral$	Control the radius of spiral line

Abbreviations

SLAM	Simultaneous Localization And Mapping
IR	Infrared Sensor
EPFL	Swiss Federal Institute of Technology in Lausanne
RAM	Random Access Memory
ICC	Instantaneous Center of Curvature

Chapter 1

Introduction

According to Bonabeau et al. (1999), swarm robot has been proposed in the late 1980 from the behaviour at species of insects and birds. Today it is a rising field thanks to the apace development of computing, micro-electronics and communication technology.

In nature, some species, especially birds and insects, solve problems collectively. The swarm as a whole is able to perform tasks with the objective of implementation complicated tasks which are impossible to be done by an individual due to limited cognitive and acting abilities such as building nest, defending and finding food. In the term of exploration and navigation – it is a kind of basic activities for most animals, insects use simple interaction among individuals to emerge complex collective behaviours. For instance, Menzel et al. (2005) found that honey bees guide in the light of a abundant, map-like organisation of spatial memory, so they can plan path at any position in their familiar zone. Ants release pheromone for finding the shortest path between a destination and a nest and enabling their own kind to follow the path. (Panait & Luke 2004)

This technology inspired by insects' behaviours can use mobile-robot systems to intelligently explore and navigate to accomplish some hard tasks in complex environments which cannot be completed by human or traditional machines, for example, move heavy material (i.e. lumber, stone) in an inhospitable place, break thrombus or concretion in humanity's bodies.

Multi-robots systems imitate the social behaviours of insects. With forming swarms, the groups of robots can overwhelm the individual limitation and represent overall efficient behaviours (Sperati, Trianni & Nolfi 2011). Nowadays, mobile-robot systems improve exploration and navigation field greatly. It is applied in various areas, from vacuum robot in your home to SLAM robot in outer space.

In this project, I propose a multi-robots system to realize leader-following and obstacle avoidance behaviours by using Webots, MPLAB, Bootloader and e-puck.



Figure 1.1: Vacuum cleaner robot



Figure 1.2: SLAM robot

1.1 Aim of the Project

My project aim to make a swarm of randomly located robots shape a queue formation led by a leader in dynamic and unknown environments automatically. To make the system adapt to atrocious environments, there is no given information about their initial and present localisation. Simultaneously, all the robots can avoid obstacles and collisions with each other.

1.2 Testing Bed

In this project, I use C as the programming language and Webots as the simulation software. Webots is commercial robot simulation software which is widely used in education. It is developed by Dr. Olivier Michel at the Swiss Federal Institute of Technology in Lausanne (EPFL, for short). In addition, MPLAB IDE and Bootloader are used to program e-pucks. MPLAB IDE is free software in the purpose to develop embedded application on Microchip's PIC and dsPIC microcontrollers. Bootloader is used to upload programs to the e-pucks.

In reality, I use e-puck to validate my program. E-puck is a small mobile robot with two differential wheels and some sensors. It is widely used for educational purposes. The detailed information about Webots, MPLAB, Bootloader and e-puck are given in Chapter 2.

1.3 Stages of the Project

There are four main stages in my project.

1. **Research:** This stage includes background study and literature review. Furthermore, being familiar with Webots's interface and API and the features of e-puck are also involved.
2. **Implementation:** In this stage, I analyse the problem firstly. I design the flow chart of the system and divide it into smaller modules. Each of the modules has individual goal. Furthermore, I research how to realise these modules by e-puck's sensors and actuators.
3. **Validation:** I test my program in this stage in both simulation and reality. Actually there are many differences between simulation and reality. For example, the images captured by camera are more blurred than simulation. The light source in reality has more influence. The real background colour is not as ideal as simulation's. The readings and accuracies of sensors between reality and simulation are different. The noise is bigger and more unstable in reality than the white noise added in simulation. The parameters (e.g. IR offset values, distance threshold, colour threshold) are different. They have to be adjusted according to the reality situation.
4. **Writing:** Although some part of writing work is completed during or before the implementation stage, they have to be amended at last to maintain the integration and consistency of the dissertation.

1.4 Limitations

Most of limitations are incurred by e-puck. For example, the memory of e-puck is small, so I must optimise my code to reduce the memory requirement as possible as I can. Although the resolution of camera that e-puck has is 480x640, the maximum rate of Bluetooth bears only 2028 coloured pixels. I must adjust the width and height of image according to different requirements. Experiment light and proximity sensors' noise, infrared signal's noise also cause the weaknesses of this system. The detailed weaknesses and solutions will be discussed later.

1.5 Structure of the Report

Firstly, this chapter introduces the inspiration of swarm robot and its application nowadays. Then it describes the purpose of the project and the related hardware and software. At last, it represents the steps of the project and the main limitations.

The rest of the thesis is organised as follows. Chapter 2 demonstrates the related researches. Chapter 3 illustrates some related case studies and the implementation of the system. Chapter 4 shows the performance of the system in experiment. Chapter 5 concludes the project including successful criteria, the problems I was confronted and their solutions, analysis of advantages and disadvantages, future work.

Chapter 2

Literature Review

This project uses wheeled mobile robots to design a multi-robot system – all the robots follow one-by-one to form a queue which is led by a unique leader robot. At the same time, the robots can sense the environments to avoid collisions.

2.1 Multi-robot System

With the rapid technology progress, the demands of doing tasks in extreme environments is increasing. Obviously, reducing the requirements of human' presence can decrease people's injuries. Robots are institute substitutes for human-being. However, due to the limitation of single-robot systems, the difficult and complicated tasks are hard to achieve. In this situation, multi-robot systems are more likely to accomplish the tasks efficiently. Comparing to single-robot systems, multi-robots systems are more flexible, adaptable and robust. (Xu 2010)

An excellent multi-robot system must have following features:

- **Scalability:** The problem should be capable to adapt different numbers of robots. Increasing the number of robots does not decrease the performance greatly. The challenge is the fact that, with the increasing number of robots, the more interference on communications causes delay and the errors are magnified which result in the distortion of formation and the deviation of trajectory. Moreover, the scalability demands the system has a robust

pre-defined structure.

- **Modularity::** The role and functionality should be reusable and interchangeable in the framework. The framework should be able to combine the appropriate modules together and suitable for different environments.

A common framework of multi-robot systems is leader-following. Usually, the leader has more computational and sensing capabilities because it is responsible for navigation, path planning and obstacle avoidance. The rest robots track the leader and keep a pre-defined formation automatically for the purpose of gathering data, transportation, or communication.

2.2 Wheeled Mobile Robot

“Wheeled robots are robots that navigate around the ground using motorised wheels to propel themselves” (*Robotics* 2013) They have different topologies depending on the number and the configuration of wheels. I use e-puck as my hardware in this project. It is a small mobile robot with two differential wheels. The next paragraph will give its detailed information.

2.3 The Hardware Platform: E-puck

E-puck is a mobile robot designed by Swiss Federal Institute of Technology in Lausanne (EPFL). It is a differential wheeled robot designed for engineering education. It has following features to be a kind of exclusive products in the purpose to be either an effective mobile robot or a well-designed educational robot. (Mondada 2009)

- **Desktop size:** Its shape is a cylinder with 7.4 cm in diameter and 4.5 cm high and its weight is 15 g so that it can be developed on the table near the computer to improve the experiment efficiency.
- **Wide range:** For educational purpose, it can be applied in various fields such as signal

processing, automatic control, embedded programming, or distributed intelligent systems design (Mondada 2009).

- **User friendly:** The EPFL alleges its interface is simple, efficient, and intuitive. However, I do not agree it if I do not have a convenient simulator.
- **Low cost:** It is affordable for universities even if under the circumstances that the experiments of swarm intelligence often require a larger number of robots.
- **Open information:** The software and hardware of e-puck are open-source.

“The e-puck robot has already been used in a wide range of applications, including mobile robotics engineering, real-time programming, embedded systems, signal processing, image processing, sound and image feature extraction, human-machine interaction, inter-robot communication, collective systems, evolutionary robotics, bio-inspired robotics, etc.” As claimed in a wiki book. (Cyberbotics’ Robot Curriculum 2009)

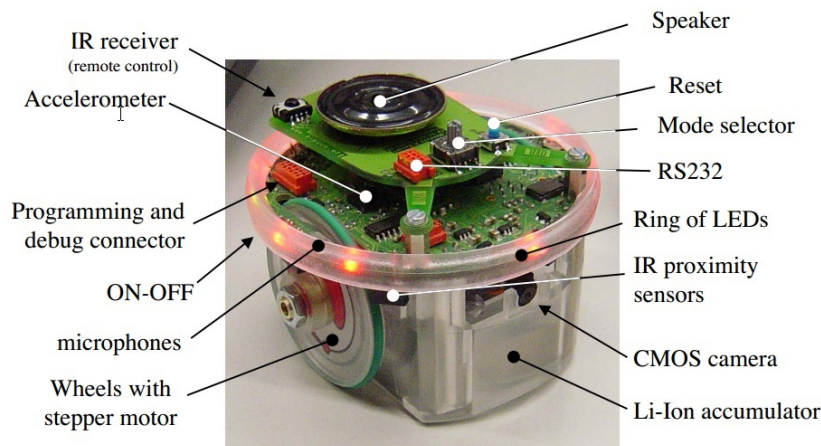


Figure 2.1: The overview of e-puck(Mondada 2009)

2.3.1 The Features of E-puck

E-puck has a dsPIC processor and many different kinds of sensors and actuators as depicted in Table 2.1 (Cyberbotics’ Robot Curriculum 2009)

Features	Technical information
Size, weight	70 mm diameter, 55 mm height, 150 g
Battery autonomy	5Wh LiION rechargeable and removable battery providing about 3 hours autonomy
Processor	dsPIC 30F6014A @ 60 MHz (15 MIPS) 16 bit microcontroller with DSP core
Memory	RAM: 8 KB; FLASH: 144 KB
Motors	2 stepper motors with a 50:1 reduction gear, resolution: 0.13 mm
Speed	Max: 15 cm/s
Mechanical structure	Transparent plastic body supporting PCBs, battery and motors
IR sensors	8 infra-red sensors measuring ambient light and proximity of objects up to 6 cm
Camera	VGA colour camera with resolution of 480x640 (typical use: 52x39 or 480x1)
Microphones	3 omni-directional microphones for sound localization
Accelerometer	3D accelerometer along the X, Y and Z axis
LEDs	8 independent red LEDs on the ring, green LEDs in the body, 1 strong red LED in front
Speaker	On-board speaker capable of WAV and tone sound playback
Switch	16 position rotating switch on the top of the robot
PC connection	Standard serial port up to 115 kbps
Wireless	Bluetooth for robot-computer and robot-robot wireless communication
Remote control	Infra-red receiver for standard remote control commands
Expansion bus	Large expansion bus designed to add new capabilities
Programming	C programming with free GNU GCC compiler. Graphical IDE (integrated development environment) provided in Webots
Simulation	Webots facilitates the use of the e-puck robot: powerful simulation, remote control, graphical and C programming systems

Table 2.1: Features of the e-puck(Cyberbotics' Robot Curriculum 2009)

2.3.2 Microcontroller

“The electronic structure of the e-puck (Figure 2.2) is built around a Microchip dsPIC microcontroller. This microcontroller complies with the educational criteria of flexibility because it embeds both a 16 bit processor with a 16 entry register file and a digital signal processor (DSP) unit. The processor is supported by a custom tailored version of the GCC C compiler. For

the e-puck, the microcontroller has 8 KB RAM and 144 KB flash memory.” As described by Mondada (2009).

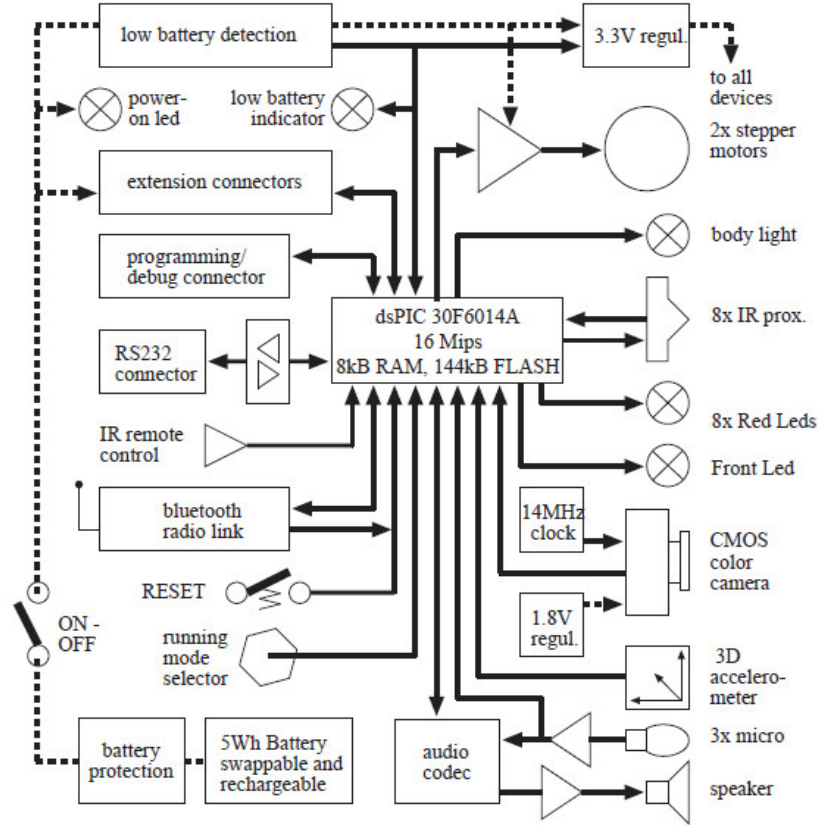


Figure 2.2: The outline of the electronic of the e-puck(Mondada 2009)

2.3.3 Sensors and Actuators

E-puck is equipped by many different kinds of sensors. They allow e-puck to possess a number of abilities.(Mondada 2009)

- **Infrared (IR) sensors:** There are eight infrared sensors around the body. (Figure 2.3 and Table 2.2) They can be used to measure the intensity of light of the nearby environment. Additionally, they can also emit infrared light and measure the received infrared light to calibrate the closeness of obstacles. So, these devices can be both proximity sensors and light sensors. These sensors can carry out simple navigation in disordered environment.

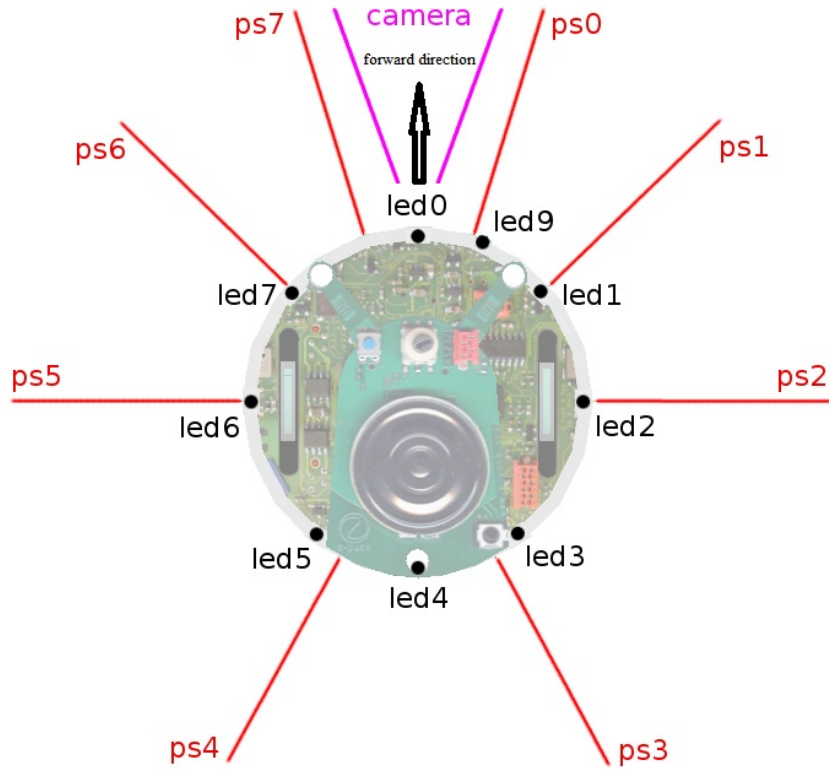


Figure 2.3: Sensors, LEDs and camera

Figure 2.4 plots the relationship between the values of proximity sensors 0, 2, 5, 7 and the distance to an obstacle and Figure 2.5 shows the values of light sensors and the distance to an LED lamp. Besides using as sensors, libIrcan library allows IR to realise local communication. The maximum bandwidth is 30 B/s and up to 25 cm distance. With receiving an infrared message, the direction and distance of a robot transmitting a message can be estimated by the receiver.

Device	x (m)	y (m)	z (m)	Orientation (rad)
IR0	0.01	0.033	-0.03	1.27
IR1	0.025	0.033	-0.022	0.77
IR2	0.031	0.033	0	0
IR3	0.015	0.033	0.03	5.21
IR4	-0.015	0.033	0.03	4.21
IR5	-0.031	0.033	0	3.14159
IR6	-0.025	0.033	-0.022	2.37
IR7	-0.01	0.033	-0.03	1.87
camera	0	0.028	-0.03	4.71239

Table 2.2: The place and orientations of IR and camera (Cyberbotics 2013)

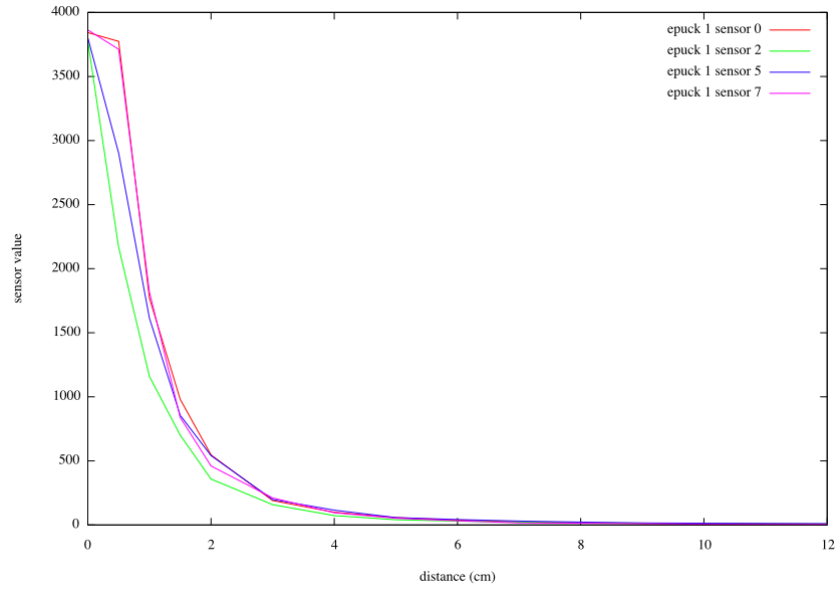


Figure 2.4: The values of proximity sensors and distance

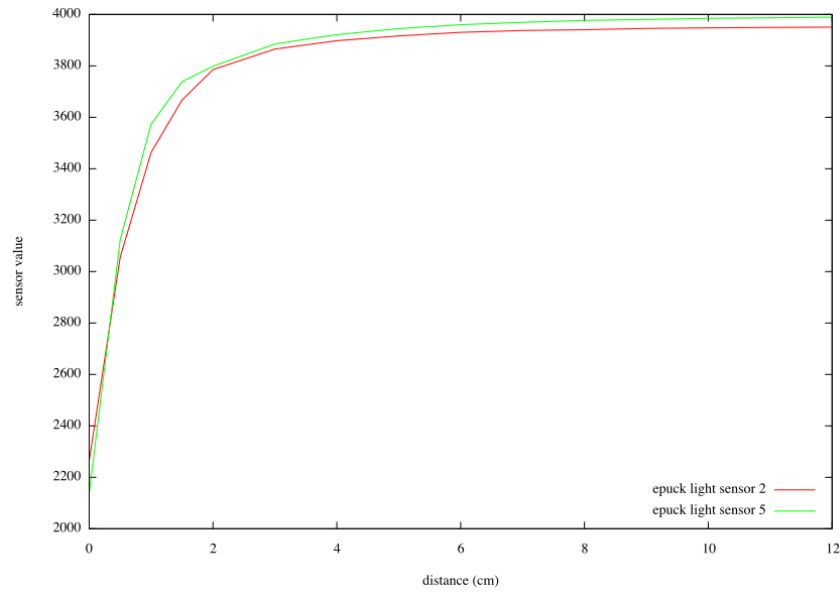


Figure 2.5: The light sensors and distance

- **Accelerometer:** Accelerometer can measure the acceleration of e-puck as a 3D vector. The unit of its reading is m/s^2 . This vector allows e-puck to estimate its inclination when it is at rest. It also can detect the collisions and falls.
- **Sound sensors:** There are three microphones capture the sound so the e-puck can sense the volume and frequency and localise the source of sound by triangulation. They are placed as illustrated in Figure 2.6.(EPFL 2013)

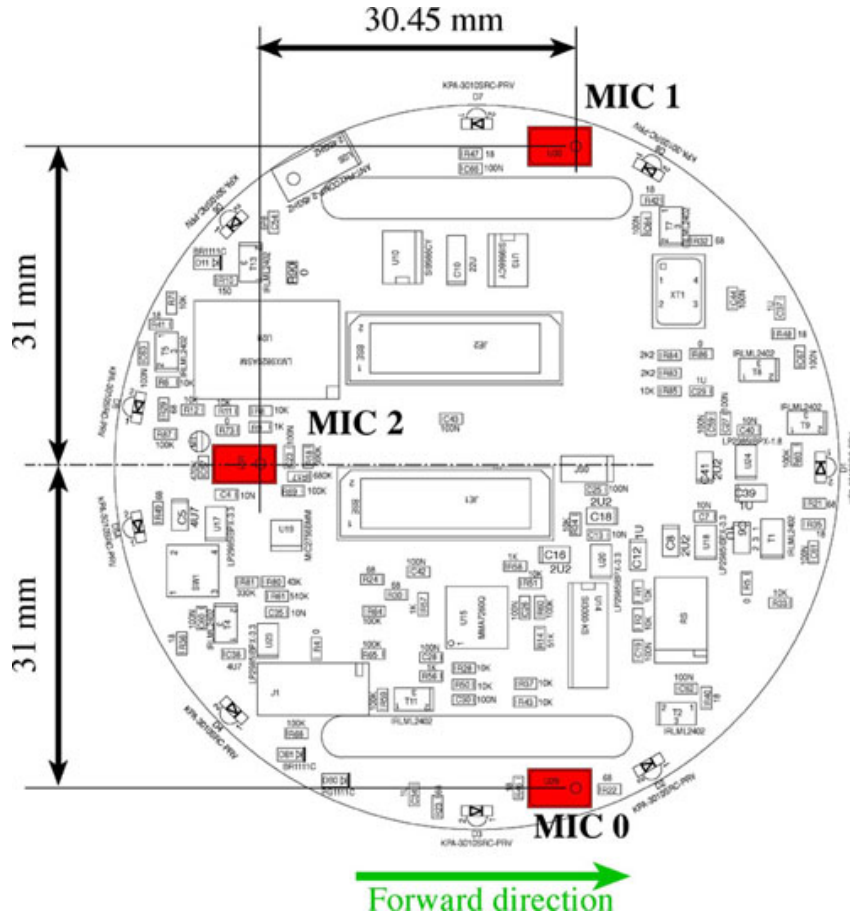


Figure 2.6: Microphones

- **Camera:** E-puck has a CMOS colourful camera with a resolution of 640*480. However, only a part of the image can be grabbed and processed because the memory is too small and the processing power is too low. Additionally, if you use remote control mode in Webots via Bluetooth communication which supports only a transmission of 2028 coloured pixel, a typical resolution is 52*39 to keep 4:3 ratio. And the field of view cannot exceed 0.7 (less than $\pi/4$).

Beside these sensors, e-puck provides the following actuators:

- **Stepper motors:** E-puck has two differential wheels controlled by stepper motors. They move independently and can go forward or backward. The resolution is 1000 steps per wheel rotation and the maximum speed is one rotation per second.
- **Speaker:** The speaker is controlled by an audio codec. Combined with the microphones,

it can communicate with another e-puck by a certain sound. It also can broadcast its own position due to the sound sensor's ability to localise the source of sound.

- **LED:** There is a LED ring around the e-puck which contains eight red LEDs. Their intensities are adjustable. Additionally, there are another red LED in the front and a green LED in the transparent body. These LEDs allow mutual visual interactions combined with the camera. They also can be used as user's interface for some certain tasks such as debugging. The front LED is special. It can emit a focused beam and project a red point. It is able to measure a longer distance to a potential obstacle by using together with the camera than the IR proximity sensors.

Other devices:

- There is a LED near the right wheel indicates low battery warning.
- A Bluetooth device allows e-puck to connect to a computer or communicate with no more than seven other e-pucks. The program is usually uploaded to e-puck via Bluetooth.
- A reset button to reset the e-puck.
- A mode selector which has 16 positions can be used to specify a four bit number. It can be used for assignment, for example, different programs, behaviours or roles.

2.4 Differential Drive Kinematics

The kinematics depends on robots. I focus on differential drive kinematics, which is suitable for e-puck. E-puck has two differential wheels and each wheel can independently be driven by a stepper motor.

When the robot is moving in a curve, the centre of the curve is known as the Instantaneous Centre of Curvature (or ICC).

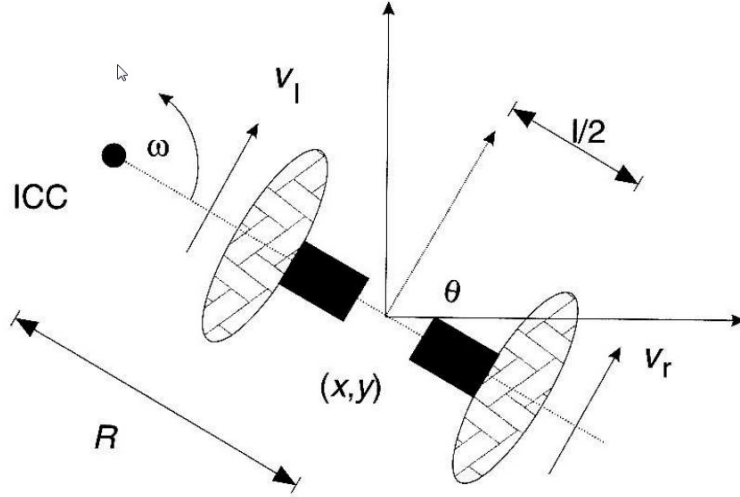


Figure 2.7: Differential drive kinematics

In Figure 2.7, R is the radius of the curve, l is the length between the wheels and the rate of rotation is ω . So, the velocities of left and right wheels are:

$$\omega(R + l/2) = V_R \quad (2.1)$$

$$\omega(R - l/2) = V_L \quad (2.2)$$

Knowing V_R, V_L , we can solve R and ω .

$$R = \frac{l}{2} \frac{V_R + V_L}{V_R - V_L} \quad (2.3)$$

$$\omega = \frac{V_R - V_L}{2} \quad (2.4)$$

Using this equation, the ICC's location is:

$$ICC = (x - R \sin(\theta), y + R \cos(\theta)) \quad (2.5)$$

And at time $t + \delta t$, the robot's position will be:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$

This is forward kinematics. The robot's position can be determined if robot's initial position and orientation, moving angular velocity and time are all known. (Dudek & Jenkin, 2010) A differential wheeled robot cannot move along the axis directly. Also it is very sensitive to slight changes in velocity in each of the wheels. Small errors can accumulate to affect the robot trajectory. They are also very sensitive to uneven or slippery ground.

2.5 Odometry

E-puck has two encoders to record the accumulated spins of left and right wheels. It can be used to estimate e-puck's position and orientation relative to the initial position and orientation. (Michel 2012)

Note S'_L and S'_R are the values of left and right encoders respectively. l is the length between the differential wheels. So the difference of orientation and position:

$$\theta' = \frac{S'_R - S'_L}{l} \quad (2.6)$$

$$x' = \frac{S'_R + S'_L}{2} \cos(\theta + \frac{S'_R + S'_L}{2b}) \quad (2.7)$$

$$y' = \frac{S'_R + S'_L}{2} \sin(\theta + \frac{S'_R + S'_L}{2b}) \quad (2.8)$$

So if the initial position is

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

the new position is:

$$p = \begin{bmatrix} x + x' \\ y + y' \\ \theta + \theta' \end{bmatrix}$$

Figure 2.8 presents an example of odometry. An e-puck goes from (0,0,0) to (0.4,0.3,0). Its position is estimated. (Xu 2010)

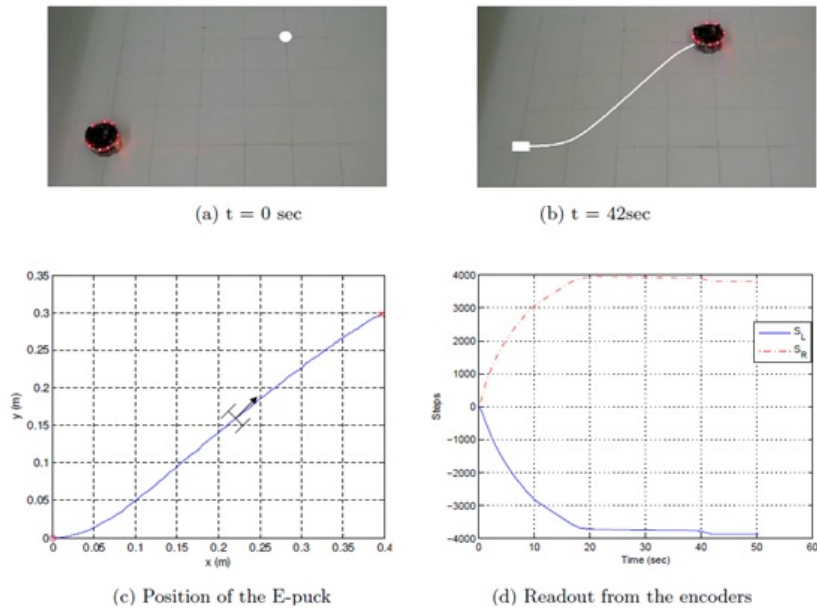


Figure 2.8: An example of odometry

2.6 Communication

Dividing by the objects of communications, there are three kinds of communications in multi-robot systems. The one is the communication between robots and environments and the other two are robot-robot, robot-human. The last one is not concerned in this paper, because this system is automatic.

In this project, e-puck use IR and camera to sense environments. IR can detect the distance to objects and camera can capture images.

In respect of robot-robot communication, e-puck can use IR and Bluetooth or radio and Wi-Fi with a proper accessory. I do not have the extension so I cannot use radio and Wi-Fi communications. Camera allows e-puck to realise visual tracking and it works perfectly in simulation. But in practise, the image is blurred and the colourful pixels are distorted seriously. So, I only use visual tracking in simulation and change to IR communication in my final design. Comparing to Bluetooth, the receivers can get the senders' distances and orientations conveniently when they get message sent by IR with the support of libIrcan library. These data are value for multi-robot system.

2.7 Software

Webots has excellent simulation interface and superior API which make implementation comfortable. I use Webots as simulation software to validate my algorithm and parameters. However, it does not support libIrcan to realise IR communication. Therefore, I compile and upload my program by MPLAB IDE which uses the standard API of e-puck. It is totally different for Webot's API, thus I spent a lot of time to re-write my code.

2.7.1 Webots

Webots provides a rapid prototyping environment for modelling, programming and simulating mobile robots. Simulation before investigation with real robots is important because simulation often means easier to setup, cost less, quicker and more comfortable development. For instance, the experiment with 100 e-pucks can be done in simulation even you do not have one. (Michel 2004)

Webots has many prototypes of mobile robots, including wheeled, legged and flying robots. Thanks to this fact, adding, removing, or moving an e-puck need only several clicks. Moreover, user can establish 3D virtual worlds with physics properties such as light source, gravitational acceleration, friction coefficients and so on. (Cyberbotics 2013) The user can add static or dynamic objects and edit their properties easily such as position, rotation, colour, texture, geometry appearance and bound.

Webots supports many languages including C, C++, Java, Python, Matlab.

Figure 2.9 is the GUI of Webots. there are four main windows: the 3D window that displays and allows to interact with the 3D simulation, the scene tree which is a hierarchical representation of the current world, the text editor that allows to edit source code and the console that displays both compilation and controller outputs. (Cyberbotics 2013)

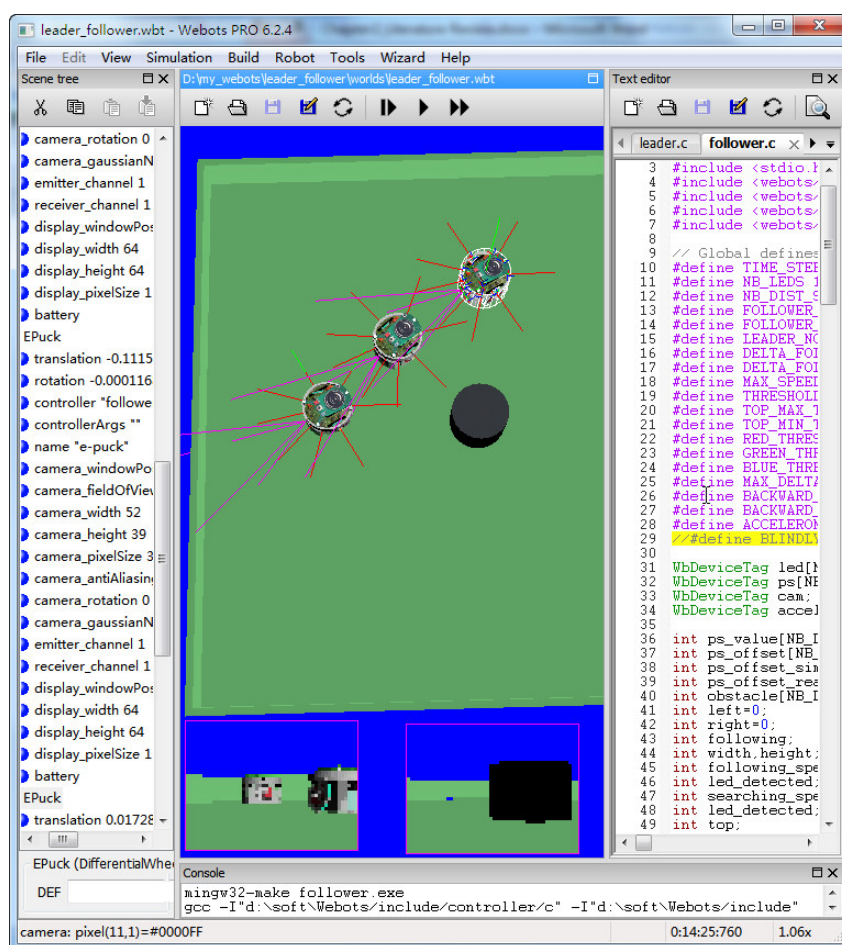


Figure 2.9: Webots GUI

After coding and setting the environment, each program will be compiled to a controller. One robot can be assigned to one and only one controller. Then users can run simulation.

After simulation, Webots has two ways to program the real e-pucks.

- **Bluetooth remote control:** Webots can communicate with and remote control e-pucks via Bluetooth. Figure 2.10 shows the e-puck control window for simulation and remote control. During remote control session, Webots must keep running.

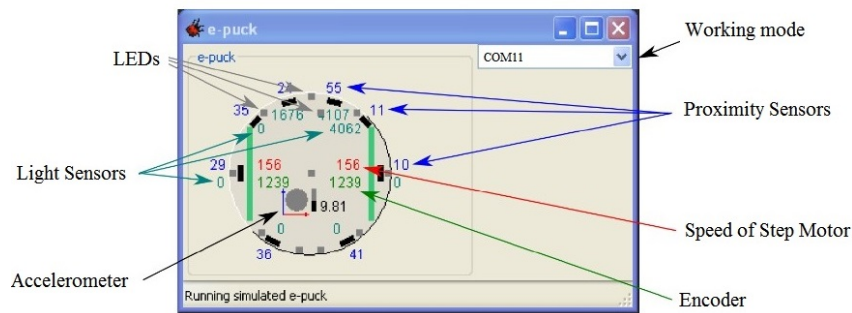


Figure 2.10: The e-puck control window

- **Cross-compilation:** An alternative way to program real e-puck is cross-compilation. Using the mode of cross-compilation, the code must be written in Webots API and C language. Cross-compilation will generate a .hex file. Then this file should be uploaded to e-puck. After that, e-puck can run the program independently, no matter that whether Webots is running or not. (Cyberbotics 2013)

2.7.2 MPLAB IDE and Bootloader

MPLAB IDE is an integrated tool set for the purpose of programming embedded devices powered by Microchip's PIC and dsPIC.

The reason of using MPLAB instead of Webots to implement the project is that libIrcorn is necessary to realise communication between e-pucks via IR. However, Webots does not support the library. If I would use that library in Webots, I have to re-write loads of API.

Figure 2.11 demonstrates the interface of MPLAB IDE.

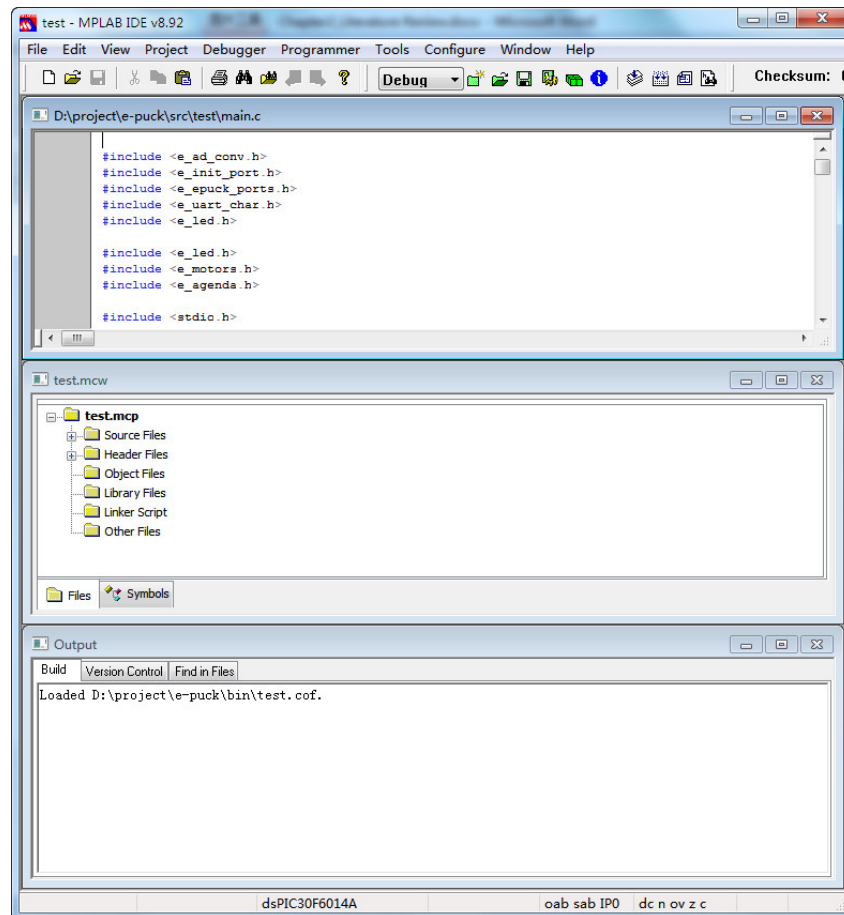


Figure 2.11: The interface of MPLAB IDE

MPLAB IDE requires a C compiler to make file. Therefore, installation of MPLAB C30 is necessary. MPLAB C30 is a full-featured ANSI compliant C compiler for the Microchip devices.

There are three steps to program the e-puck(Mondada 2006):

1. Create a new project:
 - (a) Click Project then Project Wizard.
 - (b) Select dsPIC30F6014 as device and MPLAB ASM30 Assembler as the tool suite.
 - (c) Name you project and add existing files. Figure 2.12 demonstrates these steps.

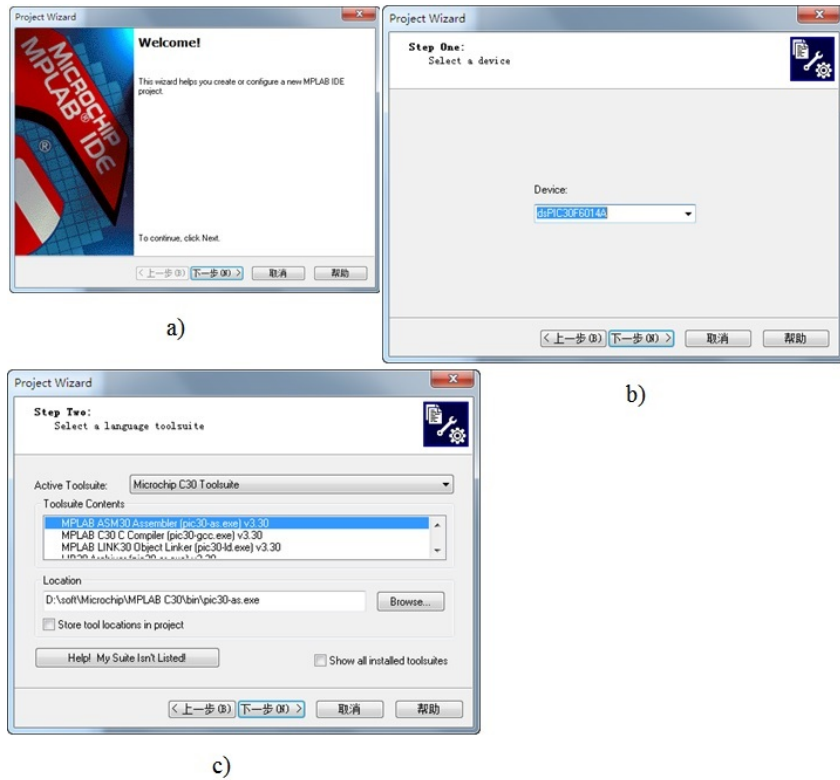


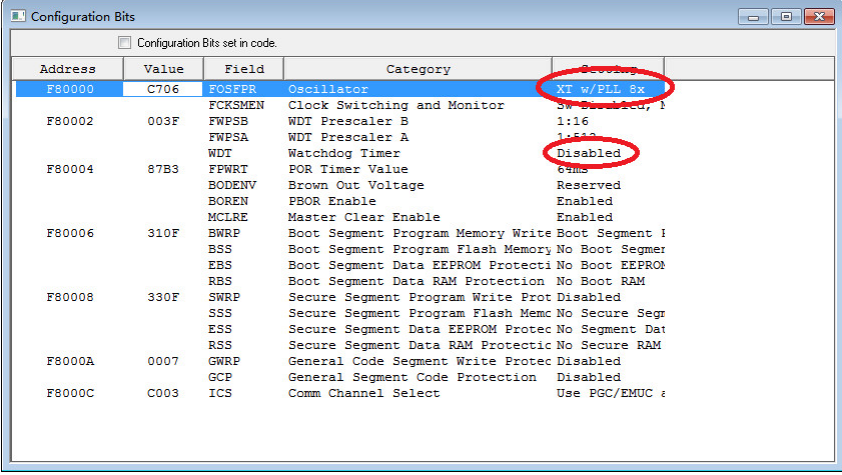
Figure 2.12: Create a new project

2. Compilation

- (a) Select Configuration and click Configuration Bits. Change following parameters as Figure 2.13:

Oscillator: XT w/PLL 8x

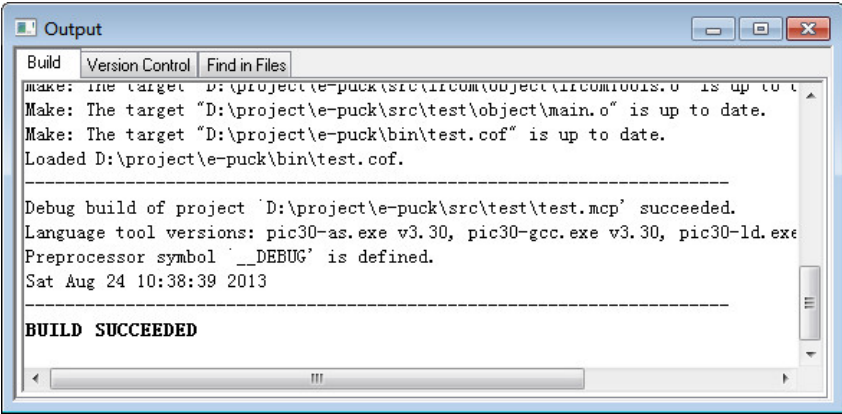
Watchdog Timer: Disabled



Address	Value	Field	Category	Setting
F80000	C706	FOSFPR	Oscillator	XT w/PLL 8x
F80002	003F	FCKSMEN	Clock Switching and Monitor	SW, SW, SW, 1
		FWPSB	WDT Prescaler B	1:16
		FWPSA	WDT Prescaler A	1:512
F80004	87B3	WDTR	Watchdog Timer	Disabled
		FWRT	POR Timer Value	67ms
		BODENV	Brown Out Voltage	Reserved
		BOREN	PBOR Enable	Enabled
		MCLR	Master Clear Enable	Enabled
F80006	310F	BWRP	Boot Segment Program Memory Write	Boot Segment i
		BSS	Boot Segment Program Flash Memory	No Boot Segmer
		EBS	Boot Segment Data EEPROM Protection	No Boot EEPROM
		RBS	Boot Segment Data RAM Protection	No Boot RAM
F80008	330F	SWRP	Secure Segment Program Write Prot	Disabled
		SSS	Secure Segment Program Flash Memc	No Secure Segr
		ESS	Secure Segment Data EEPROM Protec	No Segment Dat
		RSS	Secure Segment Data RAM Protection	No Secure RAM
F8000A	0007	GWRT	General Code Segment Write Protec	Disabled
		GCP	General Segment Code Protection	Disabled
F8000C	C003	ICS	Comm Channel Select	Use PGC/EMUC

Figure 2.13: Configuration settings

- (b) Select Project and click Make or input F10 to compile project. (Figure 2.14) It will generate a .hex file.



```

Output
Build Version Control Find in Files
make: the target "D:\project\e-puck\src\test\object\main.o" is up to date.
Make: The target "D:\project\e-puck\bin\test.cof" is up to date.
Loaded D:\project\e-puck\bin\test.cof.

-----
Debug build of project 'D:\project\e-puck\src\test\test.mcp' succeeded.
Language tool versions: pic30-as.exe v3.30, pic30-gcc.exe v3.30, pic30-ld.exe
Preprocessor symbol '__DEBUG' is defined.
Sat Aug 24 10:38:39 2013

-----
BUILD SUCCEEDED

```

Figure 2.14: Compile project

3. Program e-puck

The final step is using Bootloader to upload .hex file to e-puck via Bluetooth, so the computer must have Bluetooth adapter. Firstly, click the Bluetooth icon then add a new device. Secondly, enter the pairing code. Then windows will install the driver automatically. (Figure 2.15) The paring code is the e-puck's ID printed on the body. (Figure 2.16)

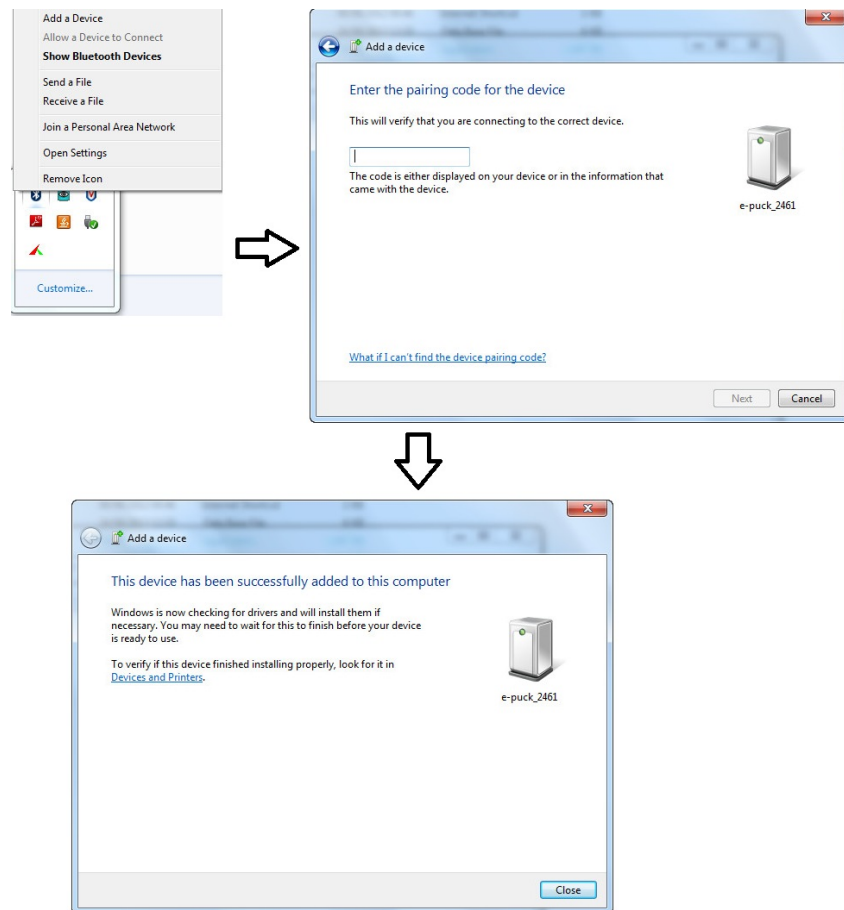


Figure 2.15: Add e-puck into Bluetooth devices



Figure 2.16: E-puck's ID

The final step is using Bootloader to upload .hex file to e-puck. Figure 2.17 is the interface of tiny Bootloader.

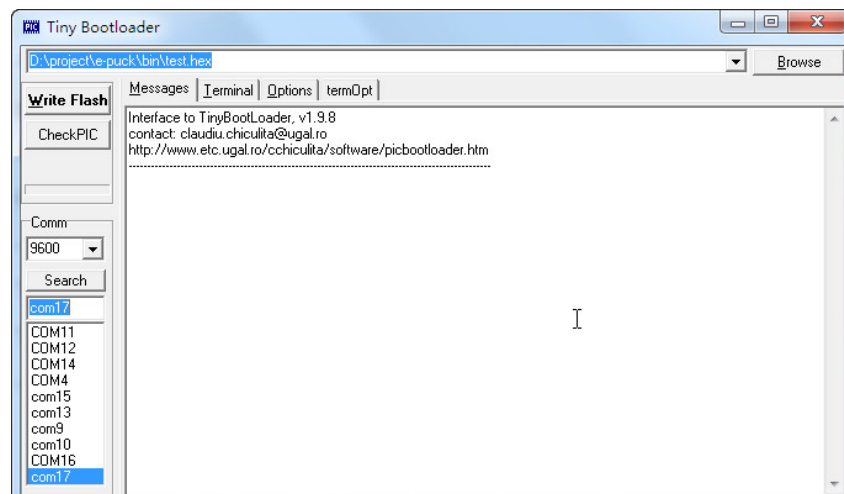


Figure 2.17: The interface of tiny Bootloader.

Chapter 3

Implementation

This chapter discusses the system design and the methods to realise the leader-following and obstacle avoidance behaviours.

As mentioned in Chapter 2, there are many different methods for e-puck to sense the environments and other e-pucks. The choice of methods and system design is chosen according to real situation.

3.1 The Aims of Projects

The main object of the project is making a multi-robot system. In the system, a unique robot is considered as the leader, which is responsible to navigate all the other robots. The other robots follow the leader and maintain a chain. Additionally, there is no collision either between robots or between robots and obstacles. The system can adapt to unknown and dynamic environments and it is de-centralised which means all the robots have the abilities of self-adaption and self-organising.

The first requirement asks the followers can find and track the leader. They are also able to maintain the correct relative position to the leader.

The second requirement demands the robots that sense the environment and predict an appropriate trajectory to avoid static or moving potential obstacles as well as other robots.

3.2 The Abilities of the Devices of E-puck

For the purpose of knowing how to control the devices and how they perform in a certain environment, it is necessary to research some abilities of e-puck's devices before implementing the project.

3.2.1 Stepper Motors

The two stepper motors control the left and right differential wheels of e-puck respectively. They divide a full rotation into 1000 equal steps and the maximum speed is 1000 steps per second. E-puck cannot move along axis directly. However, each stepper motor has an encoder which counts the steps from the last reset of the encoder. The relative position and orientation can be calculated by the two values of encoders according to the odometry mentioned in the previous chapter.

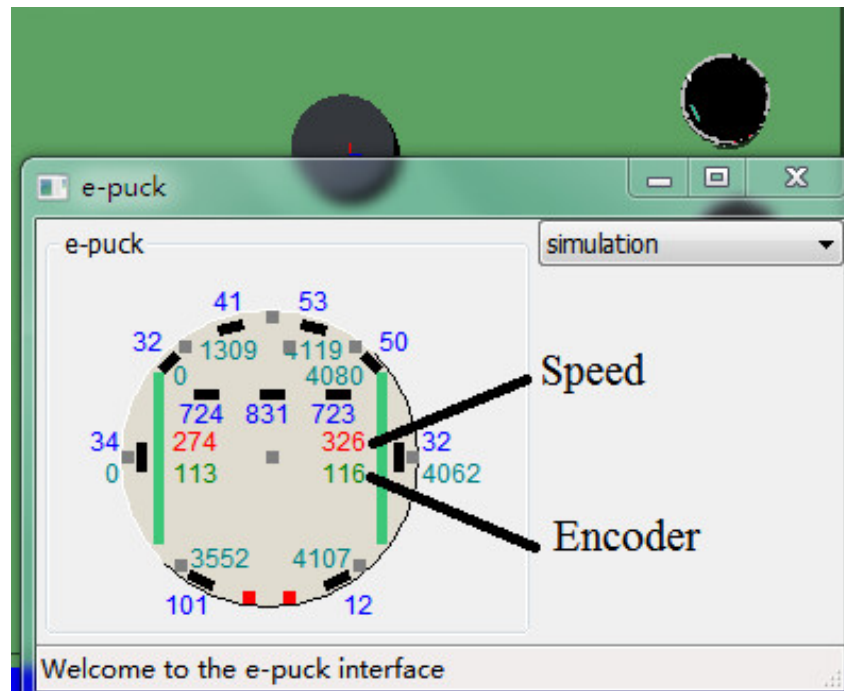


Figure 3.1: Stepper motors of e-puck

3.2.2 IR Sensors

As the light sensor, IR sensors get the intensity of received light. The more intense light they receive, the less the values are. Figure 3.2 represents that the values of sensors back to light are bigger than the sensors' face to light.

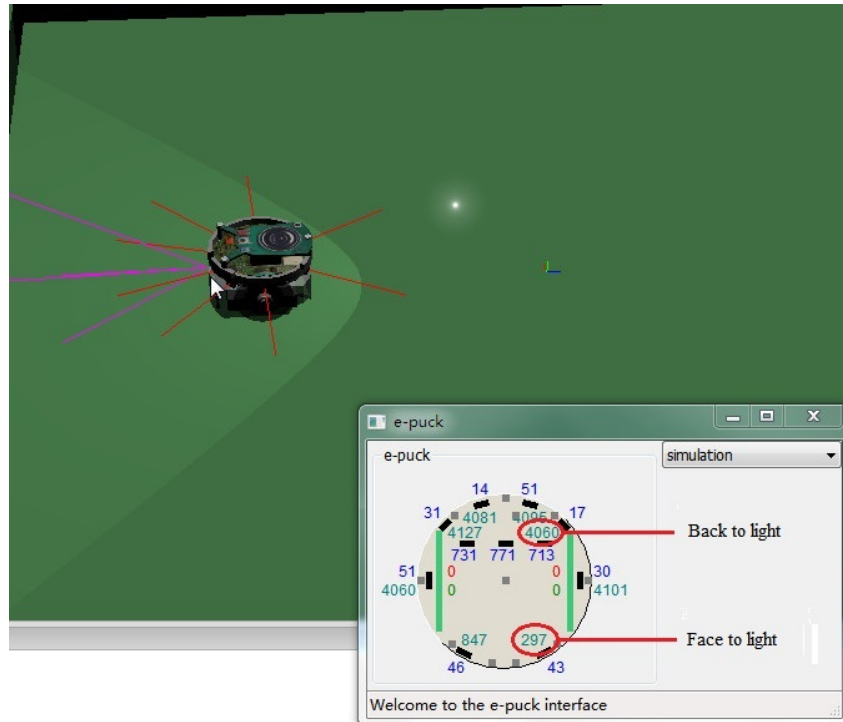


Figure 3.2: Light sensors

IR sensors can also be used as proximity sensors. They emit infrared light and measure the received infrared light which bounces on obstacles. The bigger the values are, the closer the obstacle is. Neither the values of light sensors nor proximity sensors are proportional to the light's intensity or distance. Figure 2.4 and Figure 2.5 in last chapter plot them.

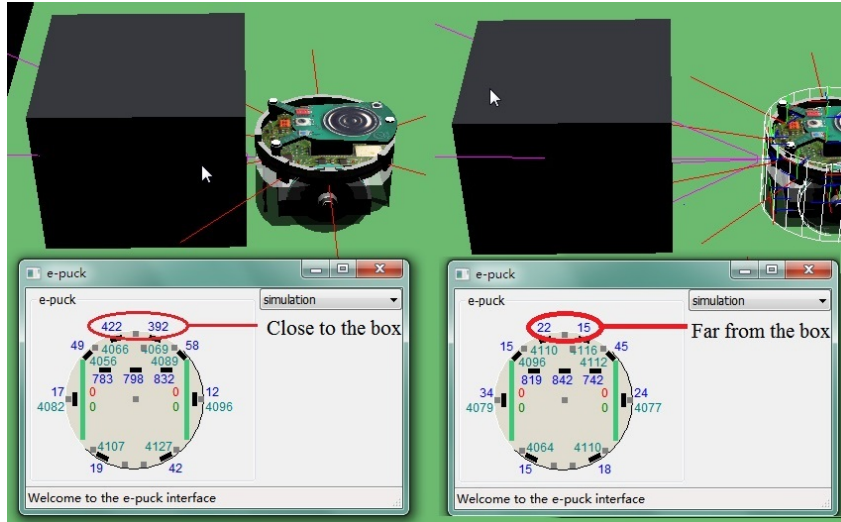


Figure 3.3: Proximity sensors

Besides, IR can be used to realise communication between e-pucks with the support of libIrcom. Its maximum rate is 30 B/s and maximum range is 25 cm. The receiver can measure the angle and distance from the sender's centre to the receiver's centre.

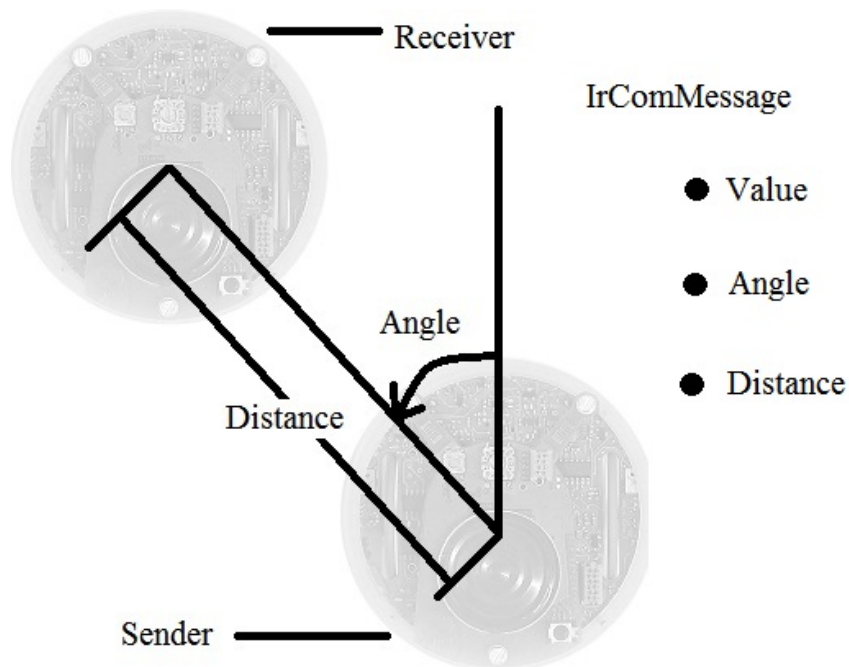


Figure 3.4: Message sent by IR

3.2.3 Accelerometer

Accelerometer measures the acceleration of e-puck as a 3D vector. It equals to gravitational acceleration when the e-puck is at rest on level ground and equals to zero if it is falling freely, so it is easy to detect a fall according to the z-axis value. When an e-puck is at rest on a slope, accelerometer can be used to measure the inclination and orientation of the slope by using the trigonometry. In addition, an unexpected violent change of acceleration usually means a collision happens.

3.2.4 Camera

The e-puck's visual sensor is a colour camera with the resolution of 480*640 in RGB mode. However it cannot grab images with this resolution due to the limitation of memory and processor's power. The bandwidth of Bluetooth also constrains the maximum resolution if e-puck is used in remote control mode.

Setting the height of resolution equal to 1, it can be used as a linear camera. In addition set the field of view equal to $\pi/4$ (the camera will point to the ground in front of the e-puck), e-puck can track the line with a certain colour. (Cyberbotics' Robot Curriculum 2009)

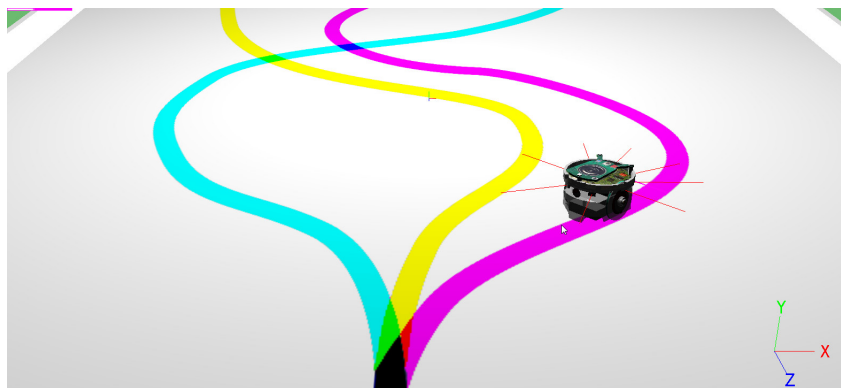


Figure 3.5: Tracking line behaviour

Camera also allows e-puck to track an object with a certain colour in a fixed distance. In the

example demonstrated in Figure 3.6, an e-puck is following another one with the rear red LED on. The follower changes its speed to keep the distance to the leader according to the location of the top side of LEDs in camera. (The top side higher than the threshold means the distance is too close, the follower will decrease the speed and vice versa.) At the same time, the difference between left and right wheel's speeds is changed to keep the LEDs in the middle of the image so the follower can keep facing to the leader. This method allows e-puck to have a simple leader-following behaviour.

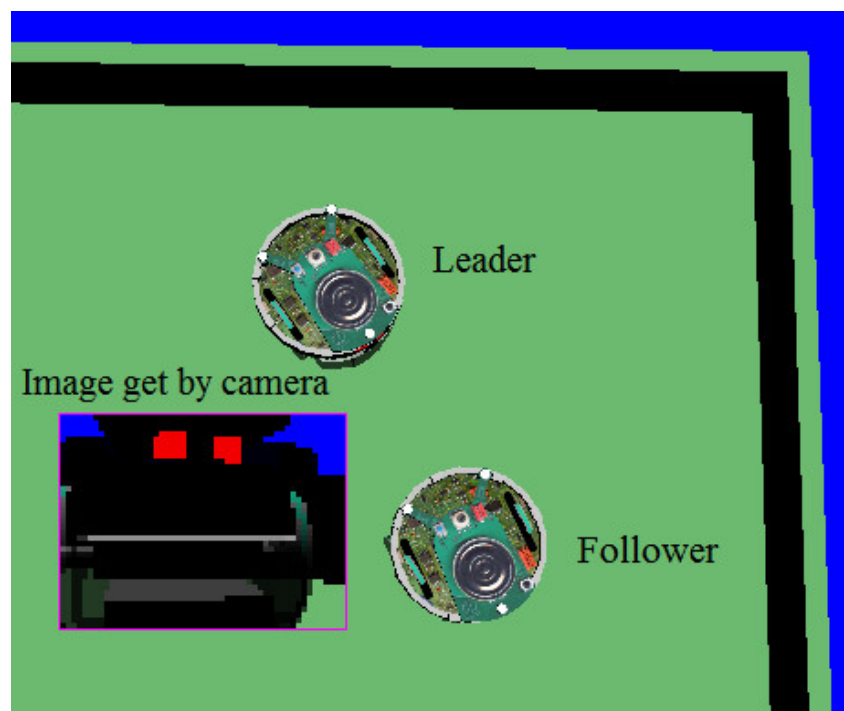


Figure 3.6: Leader-following behaviour

3.3 Design of the Modules

For the modularity of the multi-robot systems, my system is divided into several modules, including obstacle avoidance, leader-following and searching.

3.3.1 Obstacle Avoidance

This module enables e-pucks to detect the distance and direction to obstacles which are close. The speed of stepper motors are changed according to the values of proximity sensors. There are two methods to achieve its goal.

3.3.1.1 Method I

The first method is giving different weights to the proximity sensors and adjusting the speeds of left and right differential wheels. Denote the vector with proximity sensors' values by PS , the vector with weights by W , the speed of left wheel S_L and right S_R . S is the speed which e-pucks run at without detecting any obstacle, S_{max} is 1000 which is the maximum speed of e-puck. θ is the threshold of distance. Δ_{max} controls the maximum angular velocity during avoiding obstacles. Let Δ equal to:

$$\Delta = \min \left(\sum_{i=0}^7 W_i * PS_i (if PS_i > \theta), \Delta_{max} \right) \quad (3.1)$$

Then,

$$S_L = \min (S - \Delta, -S_{max}) \quad (3.2)$$

$$S_R = \min (S + \Delta, S_{max}) \quad (3.3)$$

The algorithm is described by pseudo-code below:

Algorithm 1

```
1: //read PS
2:  $\Delta = 0$ 
3: for  $i = 0 \rightarrow 7$  do
4:   read  $PS(i)$ 
5:   if  $PS(i) > \theta$  then
6:      $\Delta \leftarrow \Delta + w(i) * PS(i)$ 
7:   end if
8: end for
9: if  $\Delta > \Delta_{max}$  then
10:    $\Delta \leftarrow \Delta_{max}$ 
11: end if
12: //set speed
13:  $S_L = S - \Delta$ 
14:  $S_R = S + \Delta$ 
15: if  $S_L < -S_{max}$  then
16:    $S_L \leftarrow -S_{max}$ 
17: end if
18: if  $S_R > S_{max}$  then
19:    $S_R \leftarrow S_{max}$ 
20: end if
21: return  $S_L, S_R$ 
```

Considering the PS is more than zero even if there is no obstacles close to the e-puck, a vector with offset values should be subtracted from PS . The offset vector is an adjustable parameter based on environment condition including light intensity and background colour.

Therefore, the equation and statements which calculate Δ are changed to Equation 3.4 and Algorithm 2:

$$\Delta = \min \left(\sum_{i=0}^7 W_i * (PS_i - PS_OFFSET_i) (if \ PS_i - PS_OFFSET_i > \theta), \Delta_{max} \right) \quad (3.4)$$

Algorithm 2

```
1: if  $PS(i) - PS\_OFFSET(i) > \theta$  then  
2:    $\Delta \leftarrow \Delta + w(i) * (PS(i) - PS\_OFFSET(i))$   
3: end if
```

3.3.1.2 Method II

The second method is based on the orientations of proximity sensors and odometry. Assuming the forward direction is 0, clockwise rotation is positive direction, and the orientations of proximity sensors are shown below (rad):

IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
0.2967	0.8727	1.5708	2.618	3.6652	4.7124	5.4105	5.986

Table 3.1: IR sensors' orientations (rad)

Denote the angular velocity to avoid the obstacle as α and assume that:

$$\alpha = \tan^{-1} \frac{y}{x} \quad (3.5)$$

On the basis of odometry, to avoid an obstacle with the orientation of R , the relative displacement (x', y') is:

$$x' = \cos(R) \quad (3.6)$$

$$y' = \sin(R) \quad (3.7)$$

Therefore, x and y should be (Offset values also need to be subtracted here and denote orientation of i^{th} sensor as ω_i):

$$x = - \sum_{i=0}^7 \cos(\omega_i) * (PS_i - PS_OFFSET_i) (if PS_i > \theta) \quad (3.8)$$

$$y = \sum_{i=0}^7 \sin(\omega_i) * (PS_i - PS_OFFSET_i) (if PS_i > \theta) \quad (3.9)$$

Then α can be calculated. After that the speed can be set as below: (A is a scale factor) If $\alpha \geq 0$,

$$S_L = \cos(\alpha) * A \quad (3.10)$$

$$S_R = A \quad (3.11)$$

If $\alpha < 0$,

$$S_L = A \quad (3.12)$$

$$S_R = \cos(\alpha) * A \quad (3.13)$$

The pseudo-code is in Algorithm 3:

Algorithm 3

```
1: //read PS and calculate  $x$  and  $y$ 
2:  $x = 0$ 
3:  $y = 0$ 
4: for  $i = 0 \rightarrow 7$  do
5:   read  $PS(i)$ 
6:   if  $PS(i) - PS\_OFFSET(i) > \theta$  then
7:      $x \leftarrow x - \cos(sensorDirection(i)) * (PS(i) - PS\_OFFSET(i))$ 
8:      $y \leftarrow y + \sin(sensorDirection(i)) * (PS(i) - PS\_OFFSET(i))$ 
9:   end if
10: end for
11: //get the expected angular velocity
12:  $angle = \arctan(y, x)$ 
13: //get the linear speed according to odometry
14: if  $angle \geq 0$  then
15:    $S_L = 1$ 
16:    $S_R = \cos(angle)$ 
17: else
18:    $S_L = \cos(angle)$ 
19:    $S_R = 1$ 
20: end if
21: //set speed
22:  $S_L = S_L * A$ 
23:  $S_R = S_R * A$ 
24: return  $S_L, S_R$ 
```

3.3.1.3 The Chosen Method

My final choice to avoid obstacle is the second one. It is based on the two following reasons. Firstly, the first method sum up the proximity sensors' weighted values directly. It also accumulates the noise of proximity sensors and decreases the performance greatly. Accordingly, the second method use the summation to calculate the instantaneous angular velocity instead of the

absolute speed. This fact neutralises the noise so the interference is smaller. Secondly, I draw the trajectory in simulation when an e-puck is avoiding an obstacle. The second method has shorter travelling distance in the most cases.

3.3.2 Leader-Following

The method to follow the leader depends on the choice of sensors. Basically, the follower can follow the leader by visual tracking based on camera or by odometry based on IR communication.

3.3.2.1 Method I

Visual tracking is similar to the problem discussed in 3.2.4. Follower adjusts its speed to keep a fixed distance to the leader and facing to the leader. W is denoted the width of camera and H the height. Assuming there are N red pixels exist and their position are $P(x, y)$ between $(0, 0)$ and $(W - 1, H - 1)$ from top left to bottom right. Note that the images captured by e-puck's camera are upside down and left-right reversed in practice. Therefore, the topside of LEDs is the lowest red pixel (denote as T) in the images and followers should turn left if the average width (denote as M) of red pixels is in the right half.

$$T = \max_{i=0 \dots N-1} \{y_i\} \quad (3.14)$$

$$M = \frac{\sum_{i=0}^{N-1} x_i}{N} \quad (3.15)$$

If I want to fix the topside in the height of H' ,

$$S_L = V + ((H - H') - T) * A - (M - W/2) * B \quad (3.16)$$

$$S_R = V + ((H - H') - T) * A + (M - W/2) * B \quad (3.17)$$

Denote the follower's basic speed as V . In addition A and B are the coefficients to tune the

magnitude of the speed changed due to turning to the leader and maintaining the distance respectively.

The pseudo-code describes this action is:

Algorithm 4

```

1:  $T = 0$ 
2:  $M = 0$ 
3:  $N = 0$ 
4: for each pixel in image do
5:   if pixel( $x_i, y_i$ ) is red then
6:     if  $T > y_i$  then
7:        $T \leftarrow y_i$ 
8:     end if
9:      $M \leftarrow M + x_i$ 
10:     $N \leftarrow N + 1$ 
11:   end if
12: end for
13:  $M \leftarrow M/N$ 
14:  $S_L \leftarrow V + ((H - H') - T) * A - (M - W/2) * B$ 
15:  $S_R \leftarrow V + ((H - H') - T) * A + (M - W/2) * B$ 
16: return  $S_L, S_R$ 

```

3.3.2.2 Method II

Another way to track leader is IR communication. As mentioned before, with the support of libIrcom, e-pucks can exchange information as well as their relative distance and angle. Assuming that e-pucks plan to keep the distance of N cm to each other and form a chain (so, target angle equals to π) and current distance is D , angle is α . (Note that the distance is from centre to centre and the diameter of e-puck is 7 cm, so the actual distance is $D-7$.) Similar to visual tracking:

$$S_L = (D - 7 - N) * A - \alpha * B \quad (3.18)$$

$$S_R = (D - 7 - N) * A + \alpha * B \quad (3.19)$$

Here, A and B are the similar coefficients in visual tracking. Besides, the threshold of distance and angle is necessary because of the noise (denote as T_D and T_α , otherwise e-puck will keep trembling even if it is in the target position. The pseudo-code is following (ID is the assigned leader's ID):

Algorithm 5

```

1: get message ( $id, d, a$ )
2: if  $id = ID$  then
3:    $D \leftarrow d$ 
4:    $\alpha \leftarrow a$ 
5: end if
6: if  $D - 7 - N > T_D$  then
7:    $D' \leftarrow D - 7 - N$ 
8: else
9:    $D' \leftarrow 0$ 
10: end if
11: if  $\alpha > T_\alpha$  then
12:    $\alpha' \leftarrow \alpha$ 
13: else
14:    $\alpha' \leftarrow 0$ 
15: end if
16:  $S_L \leftarrow D' * A - \alpha * B$ 
17:  $S_R \leftarrow D' * A + \alpha * B$ 
18: return  $S_L, S_R$ 

```

3.3.2.3 The Chosen Method

I choose IR communication to implement the system. Intuitively, that method is simpler. In addition, because of the constraint of the field of view, visual tracking is only capable to sense the leader in front of the follower. However, IR can sense all the direction as long as the leader

and the follower are in the same plain. Although IR communication's range is smaller than camera, it is enough for this project.

Actually, I tried camera firstly and I found it was not feasible when I did the experiment. The captured image is blurred and the colour is distorted severely. Figure 3.7 shows the contrast of the images captured by e-puck between simulation and experiment. Both of the two images are taken by an e-puck which is following another one with the rear red LED on in dark environment. It is obviously that the one taken in experiment is not only blurred but also has colour distortion.

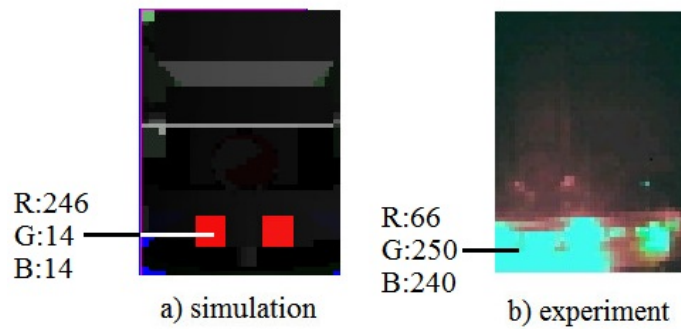


Figure 3.7: Comparison of grabbed image between simulation and experiment

3.3.3 Searching

Due to the limited IR communication's range, follower needs a searching method to get in the communication range. Intuitionally, using camera to find leader LEDs is convenient, but this method is not scalable. If there are more than two e-pucks in the system, each of followers has an assigned leader (e.g. No.1 follows No. 0 and No.2 follows No.1), it is hard to differentiate whether the detected LEDs belong to the correct leader.

Considering the experiment field is not spacious, another way is let lost followers go randomly. However, this method is unstable and blind. A more intelligent way is inspired by vacuum

cleaner robot – let the lost followers go in spiral. In addition, the obstacle avoidance behaviour is also applied. (Δ is the difference in diameter of spiral)

Algorithm 6

```

1: spiral  $\leftarrow$  0
2: if obstacle detected then
3:   obstacle_avoidance()
4: else
5:   spiral  $\leftarrow$  spiral +  $\Delta$ 
6:   if spiral > MAX_SPIRAL then
7:     spiral  $\leftarrow$   $\Delta$ 
8:   end if
9:   SR  $\leftarrow$  spiral
10:  SL  $\leftarrow$  MAX_SPIRAL
11: end if

```

3.3.4 The Integration of System

Integrating modules makes the final system. The system is designed based on rules. Algorithm 7 shows the integrated system. Figure 3.8 shows the flow chart.

Algorithm 7

```
1: if get communication then
2:   if no obstacle then
3:     if I am leader then
4:       leader's leading behaviour
5:     else
6:       follow leader
7:     end if
8:   else
9:     avoid obstacle
10:  end if
11: else
12:   if no obstacle then
13:     if I am leader then
14:       wait follower
15:     else
16:       search leader
17:     end if
18:   else
19:     avoid obstacle
20:   end if
21: end if
```

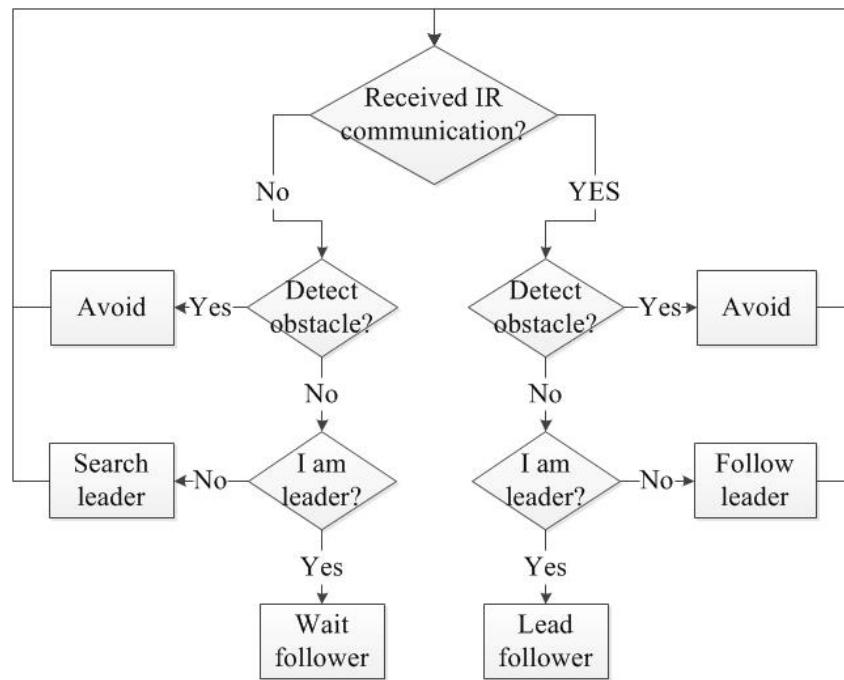


Figure 3.8: Flow chart

Chapter 4

Experiment and Result

This chapter concerns the procedure of experiment and the result. There are following parameters need to be tuned before the final experiment:

- COM_CYCLE: Maximum round-trip delay time of IR communication.
- NEIGHBOUR_TTL: TTL of the messages of neighbours
- LEADER_THRESHOLD: The distance threshold of leader's obstacle avoidance behaviour
- ANGLE_THRESHOLD and DISTANCE_THRESHOLD: The angle and distance whose absolute values are smaller than these two thresholds respectively will be treated as noise.
- leader_speed: The speed of leader's go forward behaviour
- obstacleAvoidanceThreshold: The threshold of distance to avoid obstacle for followers
- obstacleAvoidanceSpeed: The scale factor for obstacle avoidance behaviour

Thanks to the robustness of IR communication, these parameters are not changed distinctly while the environment varies.

After a number of times of experiments, the tuned parameters' setting is Table 4.1:

COM_CYCLE	1000
NEIGHBOUR_TTL:	10000
LEADER_THRESHOLD	300
ANGLE_THRESHOLD	30
DISTANCE_THRESHOLD:	1
leader_speed	200
obstacleAvoidanceThreshold	300
obstacleAvoidanceSpeed	200

Table 4.1: Final parameters' setting

In the experiment, four e-pucks are used to validate the system. Before testing, all the e-puck's IDs are set by selectors. Leader's ID is 0 and the rest e-pucks' are 1, 2, 3. Figure 4.1 shows the selector position.

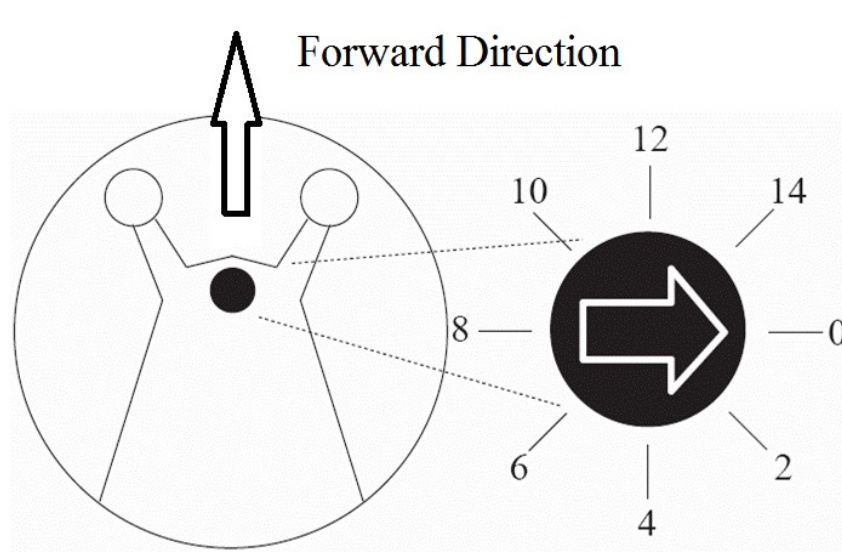


Figure 4.1: The selector position

In the whole procedure, each of them follows the previous e-puck one-by-one and all the e-pucks are able to avoid obstacles. There is a video in the CD-ROM shows the detailed performance of the system. The green blinking body LED indicates the e-puck is communicating with another one. (Note that not the green LED on the ring, the green LED on the ring indicates switch on or off.) The red LEDs on the ring indicate that obstacles are detected in that direction.

At the beginning, all the e-pucks are located in random position. Then, the followers start to search their assigned leader firstly. (No. 1's leader is No. 0. No. 2's leader is No. 1 and so on.) After several seconds, the e-pucks form a chain. Figure 4.2 demonstrates that the e-pucks form a chain from the random initial positions.

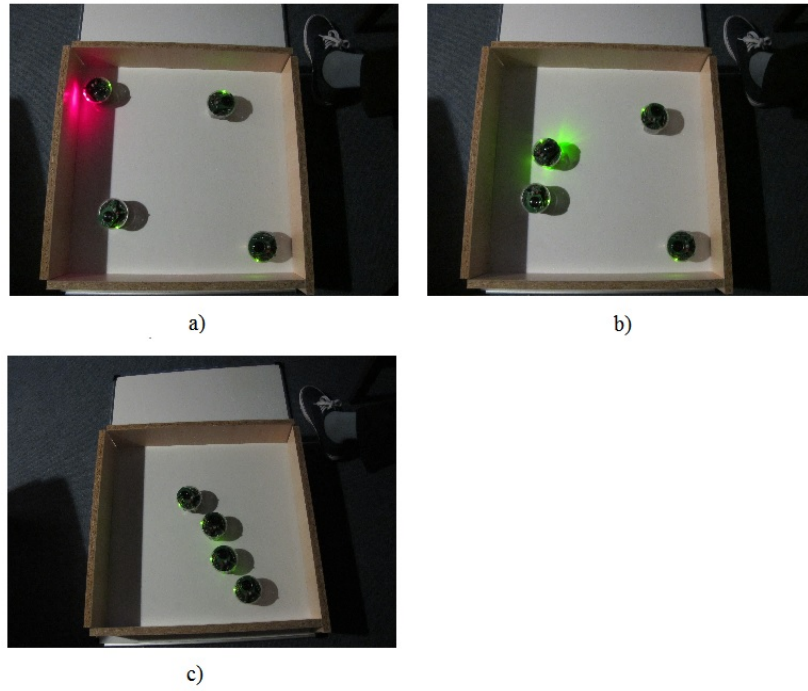


Figure 4.2: Experiment of leader-following

a): At start, e-pucks are put randomly.

b): They start to search assigned leader. Followers can avoid collision between each other during searching. If e-puck gets the communication, the green body LED will blink. But they only follow the assigned leader. The messages are sent via IR including the sender's ID, distance and direction from the receiver to the sender.

c): E-pucks follow their assigned leaders according to the messages and form a chain.

The whole procedure is collision-free. Figure 4.3 shows the leader's navigation behaviour can detect and avoid obstacles.

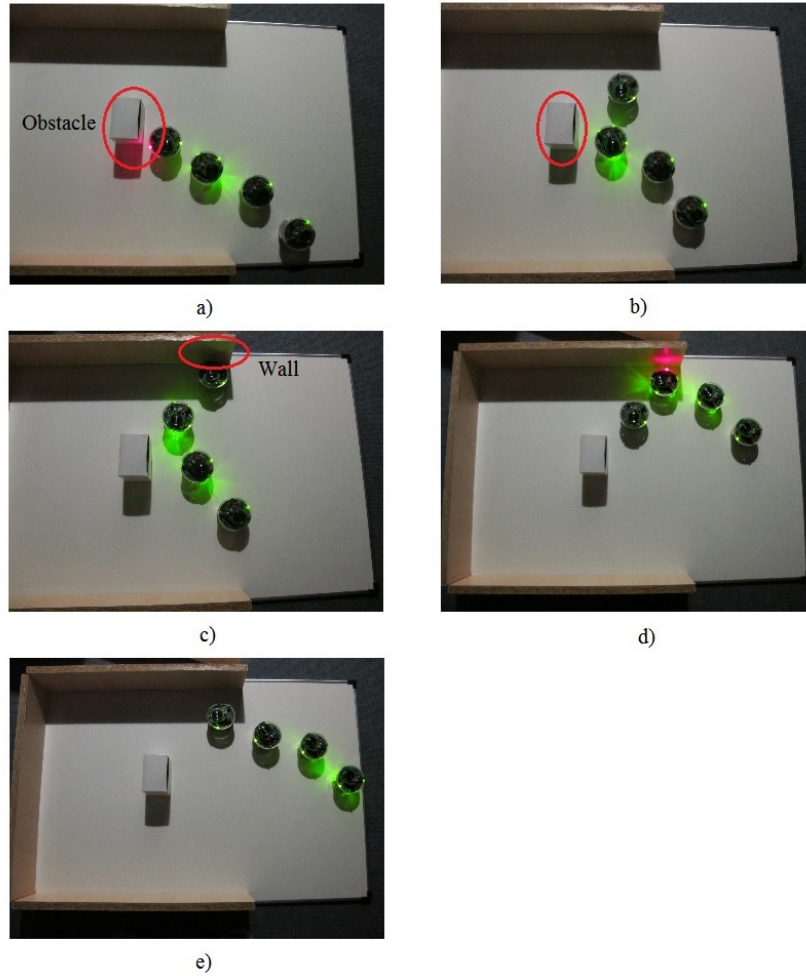


Figure 4.3: Experiment of avoiding obstacle and navigation

a) and b): The leader detects and avoids the white box (The red LED indicates the obstacle) then navigates.

c) and d): The leader detects and avoids the wall then navigates.

e): The leader goes forward without obstacle detected.

E-pucks can recover the chain if it is broken forcedly. (Figure 4.4 and Figure 4.5)

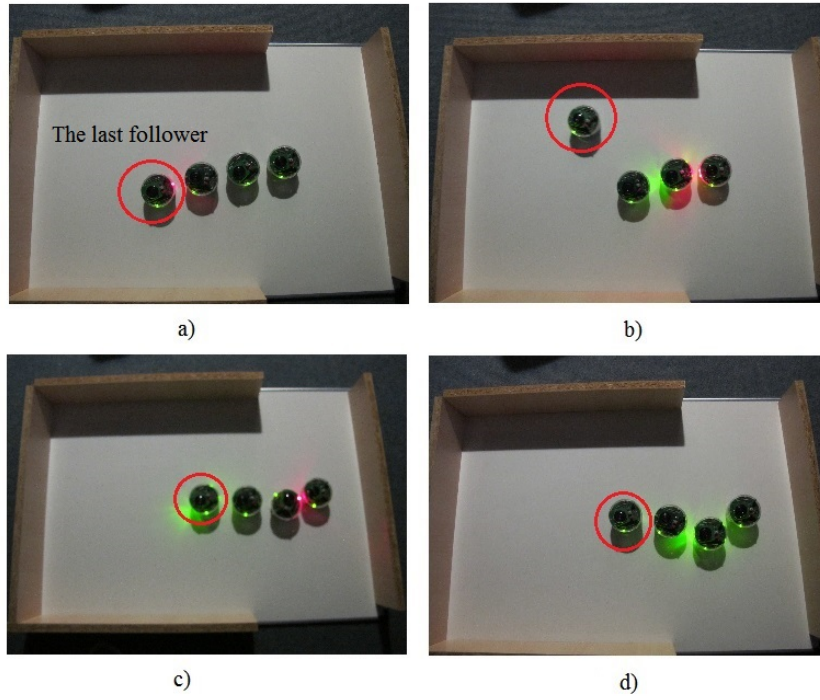


Figure 4.4: Experiment of chain recovery – move follower

- a): At start, e-pucks go in queue.
- b): Move away the last follower (No.3) by hand.
- c): No.3 finds and follows No.2.
- d): Chain is reformed.

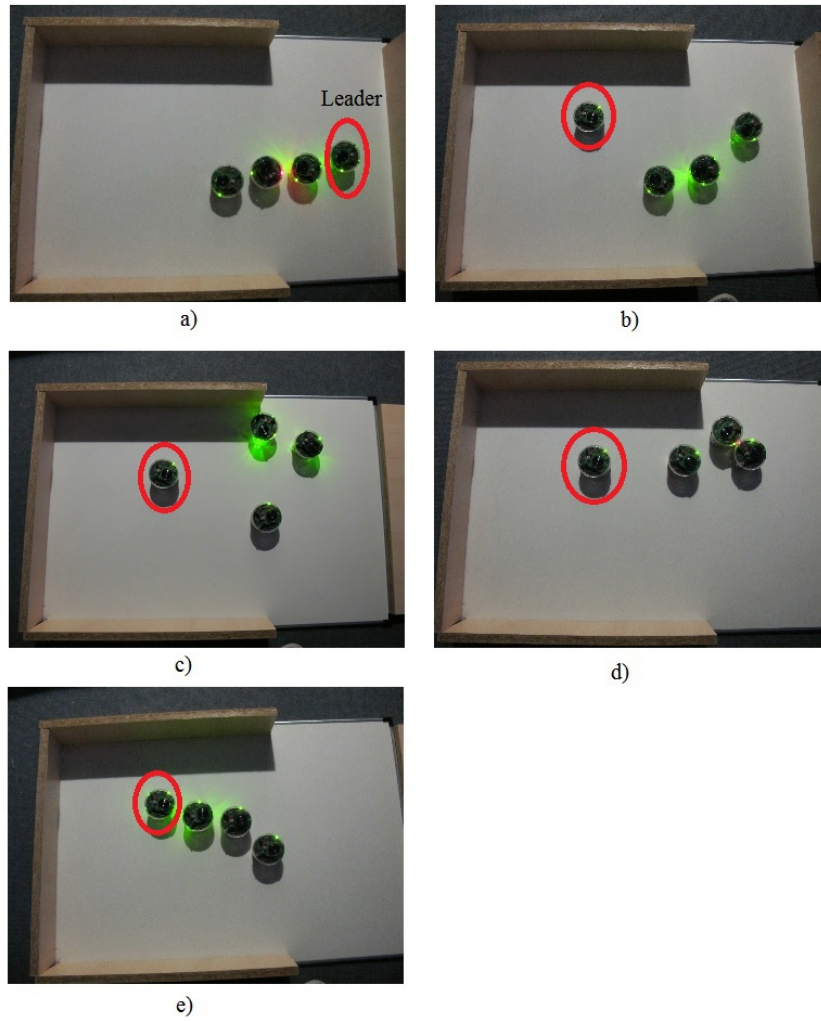


Figure 4.5: Experiment of chain recovery – move leader

- a): Initially, E-pucks go in chain.
- b): Move away the leader (No.0) by hand
- c): No.1 loses the communication from the leader and searches.
- d): The chain is recovering.
- e): Recovery is finished.

In summary, the system successfully achieves the goal. Leader-following and obstacle avoidance behaviour are integrally implemented.

Chapter 5

Conclusion and Future Work

Generally speaking, I achieve the planned goals. The System can act leader-following behaviour and avoid obstacle in the meantime. Because of the communication via IR, the system is robust. The environment light, background colour can hardly influence the performance of the system.

5.1 Challenges of the Project

Before this project, multi-robot is a totally unfamiliar field for me. Especially, the reactive control of e-puck is complex due to the physic limitation. E-puck cannot go along any axis. All the movement is led by the differential wheels. Odometry is necessary to estimate current position. It complicates the program evidently.

Another complex problem is communication which is solved properly. IR is a proper solution and it takes me a long time. At the beginning, I plan to use visual tracking and Webots to simulate and compile the project. However, as mentioned before, the camera performs badly even if I made great effort to tune it. Then, I took a long time to research the other ways to communicate. It is a dilemma for me to decide to choose IR communication or not, because I have to give up Webots which I spent a lot of time to study and research MPLAB which is also new software for me. In respect of programming e-pucks, Webots and MPLAB have totally different API which means that I have to rewrite most of my code. In addition, unlike Webots, MPLAB cannot simulate, so it is tedious to check the feasibility of my idea and debug my code.

Furthermore, libIrcan library does not have any support documentation. Combined with the MPLAB without the ability of simulation, it makes coding extremely hard. On the other hand, IR can send not only the message but also the sender's position. It makes reactive control convenient and effective. The performance of the final system indicates that the effort incurred by this decision is worthy.

Another kind of problems is the hardware's problem. E-pucks sometimes perform unexpectedly. Firstly, occasionally, the proximity sensors indicate that there is an obstacle even if there is no obstacle around the e-puck at all.(It is not noise obviously.) I do not find any information about this phenomenon and cannot differentiate it is the common problems or it only happens in my e-pucks. Secondly, the battery contacts of e-puck that I have do not work well. The shake caused by moving sometimes disconnects the batteries and the e-pucks reset. The appearance of the e-pucks which just reset is nearly the same as the e-pucks which throw error exception. It confused me and I spent several days to check my code. At last, I confirmed it was the problem of battery contacts.

5.2 Strengths and Weaknesses of the System

The system has following strengths:

- **Easy to use:** Thanks to the features of e-puck, the system is easy to use, To set up the experiment, all I need to do are uploading program to the e-pucks and switching them on.
- **Robustness:** Due to the application of IR communication, the system is robust for the interference. Unlike the visual tracking, the system with IR communication can perform excellently in environments with different lightness, brightness and background colours.
- **Scalability:** It is easy to increase the number of e-pucks. The only necessary action to add a new e-puck is uploading the program to it, setting its selector correctly and putting it in the environment. The system is convincing scalable.
- **Modularity:** All the e-pucks are using the same program whatever their roles are. The system is divided into several modules appropriately.

The weaknesses are:

- **Limited communication range and blind searching method:** The alleged range of IR communication is 25 cm. However it is smaller in reality. In this fact, an effective method is necessary to lead the follower to move into the leader's communication range. The basic vacuum robot behaviour is applied — the e-puck searches the signal in a spiral line. But the method is not intelligent and the performance is random although it is acceptable in this system.
- **Hardware failure:** As mentioned before the hardware failure decreases the system's performance seriously. It is hard to find and avoid. In respect of hardware, I cannot discover e-puck deeply in several months.

5.3 Future Work

The first future work is improving the search method. Use visual tracking to find the LEDs is a potential way to get into the communication range of the leader, but the problem is that e-puck cannot differentiate the ID of the owner of the LEDs. Comparing to IR, the range of Bluetooth is much bigger. The libIrcom also allows e-pucks to communicate with each other via Bluetooth. It can solve the problem. The combination will improve the efficiency of searching.

The second one is optimisation of the travel distance and time to form a chain and avoid obstacle. They are not optimised in my existing system.

The third one is formation control which is the most important. With a slight modification of my current system, e-pucks can form a simple pre-defined formation. But it is unstable, because e-pucks have to be able to maintain the same orientation with the leader. I have the idea but it is not implemented. Fig. 5.1 shows the method which can realise the synchronisation.

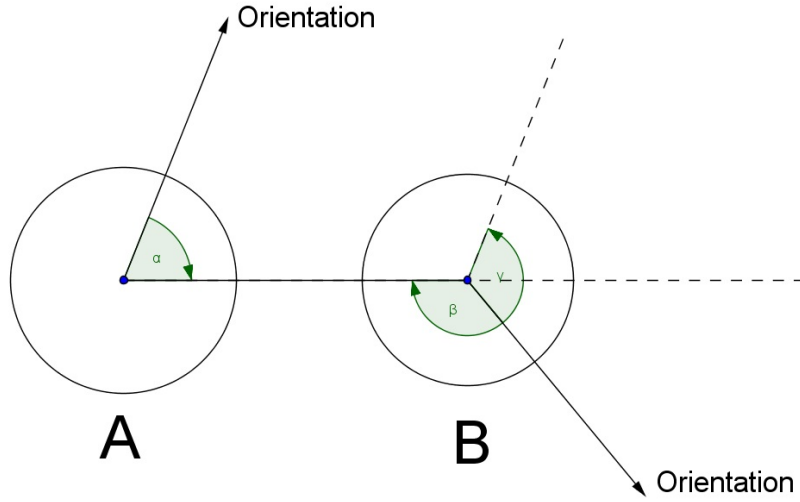


Figure 5.1: The basic method to synchronisation

In this case, it is assumed that B wants to rotate to the same orientation with A. Note that IR communication only allows receiver to measure the direction and distance to sender. When A receives message from B (message includes B's ID), A can detect B's relative direction α . After that A puts α and A's ID in the message and send it to B, B can detect β . This procedure allows B to know α and β . Therefore, B can calculate the transposed angle γ :

$$\gamma = \pi - \beta + \alpha \quad (5.1)$$

5.4 Conclusion

In this project the planned tasks are finished. The system is robust, scalable and modular. The e-puck can find their assigned leader and avoid obstacles at the same time. Leader can wait and navigate followers and the trajectory is smooth. When I am doing the project, I learned a lot about robotics and multi-robot systems. In addition, I improve my ability to discovery a new field.

Bibliography

Bonabeau, E., Dorigo, M. & Theraulaz, G. (1999), *Swarm intelligence: From Natural to Artificial Systems*, New York: Oxford University Press Inc.

Cyberbotics_Ltd. (2013), *Webots Robot Simulator – Documentation – User guide*.

URL: <http://www.cyberbotics.com/guide/>

Cyberbotics' Robot Curriculum (2009). Accessed: 2013-08-01.

URL: http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum

Dudek, G. & Jenkin, M. (2010), *Computational principles of mobile robotics*, Cambridge university press, New York, NY.

EPFL (2013), 'E-puck education robot.'. Accessed: 2013-08-01.

URL: <http://www.e-puck.org>

Menzel, R., Greggers, U., Smith, A., Berger, S., Brandt, R., Brunke, S., Bundrock, G., Hülse, S., Plümpe, T., Schaupp, F., Schüttler, E., Stach, S., Stindt, J., Stollhoff, N. & Watzl, S. (2005), 'Honey bees navigate according to a map-like spatial memory', *Proceedings of the National Academy of Sciences of the United States of America* **102**(8), 3040–5.

Michel, O. (2004), 'WebotsTM: Professional mobile robot simulation', *arXiv preprint cs/0412052*

Microchip (2006), *MPLAB® IDE Users Guide*. Accessed: 2013-08-01.

URL: http://www.cis.upenn.edu/~lee/06cse480/data/MPLAB_IDE_User_guide.pdf

Mondada, F. (2009), "introduction to the e-puck robot". Accessed: 2013-08-01.

URL: <http://eai07.di.fc.ul.pt/docs/Mondada/e-puck-intro.pdf>

Mondada, F. & Bonani, M. (2006), ‘Tutorial for programming the e-puck robot using the bootloader via bluetooth’. Accessed: 2013-08-01.

URL: http://www.verlab.dcc.ufmg.br/_media/projetos/epuck-player/getting_started_with_epuck.pdf

Mondada, F., Bonani, M. & Raemy, X. (2009), ‘The e-puck, a robot designed for education in engineering’, *Proceeding of the 9th Conference on Autonomous Robot Systems and Competitions* **1**(1), 59–65.

Panait, L. & Luke, S. (2004), A pheromone-based utility model for collaborative foraging, *in* ‘In Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems’, pp. 36–43.

Robotics (2013). Accessed: 2013-07-15.

URL: <http://en.wikibooks.org/wiki/Robotics>

Sperati, V., Trianni, V. & Nolfi, S. (2011), ‘Self-organised path formation in a swarm of robots’, *Swarm Intelligence* **5**(2), 97–119.

Xu, K. (2010), Integrating centralized and decentralized approaches for multi-robot coordination, PhD thesis, Rutgers University.