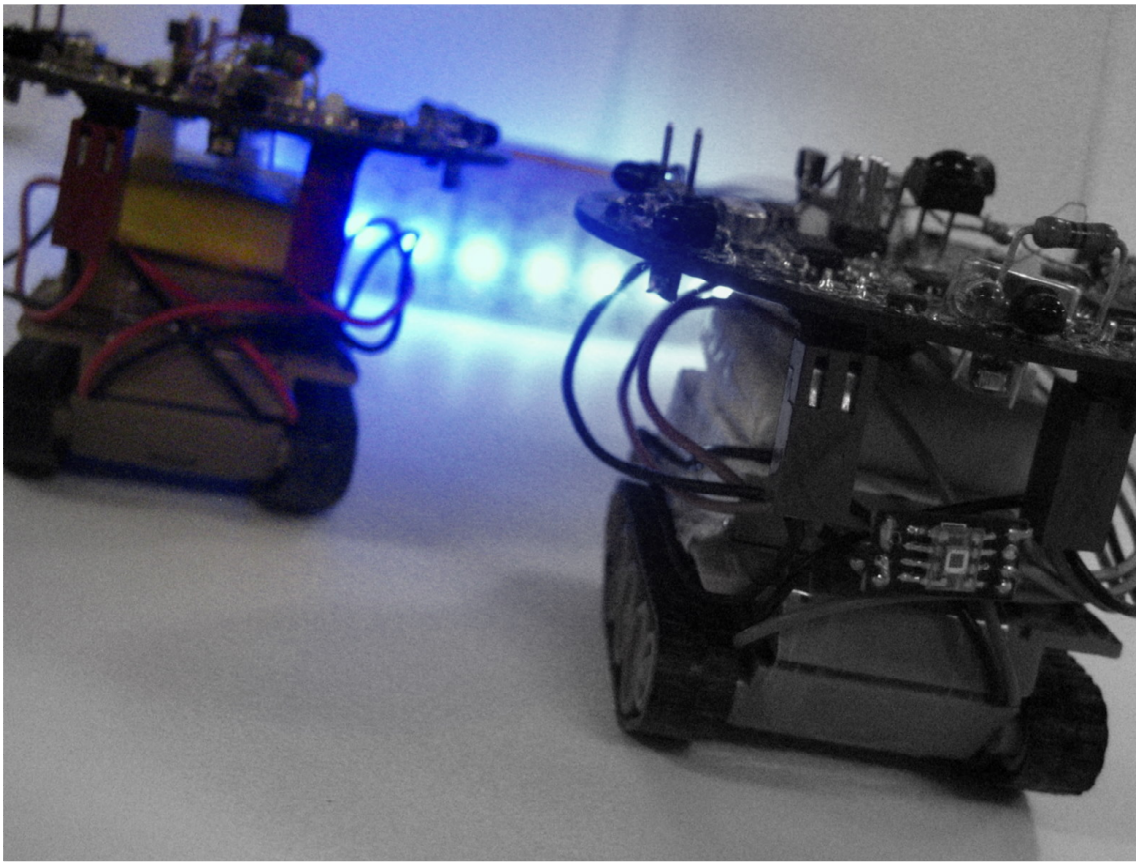
Swarm Robots Implementation – Formation and Collective Behavior



Master Thesis

Yihan Pan

2007

AALBORG UNIVERSITY, DEPARTMENT OF ELECTRONIC SYSTEMS

Automation & Control
Fredrik Bajers Vej 7C
DK-9220 Aalborg Ø
Phone.: +45 9635 8600
Web: http://www.control.aau.dk

TITLE:

Swarm Robots Implementation
- Formation and Collective Behavior

SPECIALIZATION:

Intelligent Autonomous Systems

PROJECT PERIOD:

10th semester
1th February 2007 - 1th October 2007

PROJECT GROUP: 07*gr*1035*a*

GROUP MEMBERS:

Yihan Pan

SUPERVISORS:

Thomas Bak
Trung Dung Ngo

NUMBER PRINTED: 4

NUMBER OF PAGES: 161

ANNEXES: 1 CD

FINISHED: 1th Oct. 2007

ABSTRACT:

This report documents the implementation of swarm robots. The novel designed robots are introduced and built up. The swarm robots are designed to be perceptive and cognitive. Each micro robot is equipped with six couples of IR transmitter and receiver, six ambient light sensors, one color sensor and one global IR receiver. The maneuverability is provided by two motors on each side. An MCU is used to coordinate the whole system.

Based on the hardware functionalities, algorithms are developed to provide a signal robot with several capabilities when doing navigation, including communication, distance measurement, obstacle avoidance, light source approaching and color reaction.

With these capabilities, swarm robots can work together. The algorithms of two robots' formation with obstacle avoidance are developed and validated in both simulation and experiments. Two approaches to keep the formation are discussed and compared.

A collective task is planed and separated into two subtasks. One is to find the food and move the food. The other one is to return home with security guard. For each subtask, separated algorithms are developed respect to the responsibility of each robot and validated by implementations of two robots.

Preface

This thesis is submitted for Master at the specialization of Intelligent Autonomous Systems, Section of Automation and Control, Aalborg University. The work has been carried out in the period from February 1^{st} 2007 to October 1^{st} 2007.

The main topic of this project is the implementation of swarm robots' behaviors. The simulation programs are written in `steve` Language by using free softwares *Breve V2.5.1* and *Jasmine simulator V7*. The programs for the real implementations are written in `Embedded C` Language and complied by *GCC V4.1.2*. The `Embedded C` programs are tested in free software *Visual Micro Lab V3.12*. The *AVR Studio V4.13* is used to download the programs. The results of conducted work are illustrated by *MATLAB R2007a*.

The enclosed CD-ROM contains this report in .pdf format, data sheets, references and source codes. The videos recorded from simulations and real experiments are also included.

The thesis is intended for supervisors, examiner, students and others that might have interest in swarm robots.

Aalborg University, 2007

Yihan Pan

Contents

Chapter 1

Introduction

The significations of swarm robots, formation and collective behavior are introduced firstly. Then the research backgrounds in two specific areas are addressed briefly. The motivation, scope, objectives and contributions of this project are presented continuously. In the end, the outline of the thesis is stated.

1.1 Swarm robots

The technique of robotics has achieved a significant progress in the recent decades. A lot of different kinds of Robots have been developed and applied widely for both manufacturing and scientific research. Instead of humans, robots can perform tough and repeated jobs more accurate and efficient as long as the designers let them known what they should do and how to do it. Another excellent advantage is that the robots can be used in environments unsuited for human, allowing us to explore these areas without the risk of human life.

Nowadays, swarm robots, that is multi-robot systems, are being explored extensively due to their better functionalities for specific objectives in parts of industrial and scientific areas.

Swarm robots is a novel approach to the coordination of large number of relatively simple physical robots. It is inspired form the observation of social insects, such as ants, termites, wasps and bees. These insects shows how a large number of simple individuals can interact to create collectively intelligent systems. Social insects are known to coordinate their action to accomplish tasks that are beyond the capabilities of a single individual. Most of

social insects coordination capabilities are much beyond the reach of current multi-robot systems.[sro]

Parker has presented the current state of the art in multi-robot systems. His work has identified eight primary research topics concerning multi-robot systems, including biological inspirations, communication, architectures, localization, object transport and manipulation, motion coordination, reconfigurable robots, and learning[Par00]. In this project, in order to demonstrate the swarm robots' behaviors, formation and collective behavior are implemented. During the implementation, some topics identified by Parker are covered, including communication, object transport and motion coordination.

1.2 Formation

It is easy to see formation behaviors in the nature. A flock of animals are marching with some regular formations in order to maximize their capabilities to percept the environment and response quickly when encountering emergencies. By keeping distance from each other and using communication to share the information, a flock of animals can achieve the bigger opportunities to to prey or avoid predators.

In analogy to the animals, the same idea has derived is that swarm robots could similarly get the benefits by using formation tactics. Formation allows each individual team member to concentrate its own sensors on a portion of environment. Then the information from all the members can be shared in order to ensure full coverage of an environment. This approach is potentially applicable in many other domains such as search and rescue, agricultural coverage tasks and security patrols[BA98].

For clearly understanding the benefits from formation, it would take the satellite formation as an example to illustrate[NG01].

The satellite formation flying permits scientists to obtain measurements by combining different data from several satellites rather than mounting all the instruments on one costly satellite. Multiple scientific instruments often present competing and conflicting requirements on a satellite design and its operation. In reality, complexity of a system is inherently easy to be faulty. A satellite with less functions decreases the complexity in the design stage since engineers need less effort to consider the interference between components of to design fault-tolerant control to ensure one satellite working permanently.

The above advantages ensure to collect data conveniently and correctly. The following is concerning the characteristic of data themselves. The sensitivities of some scientific

instruments are increased by expanding the effective observation baselines, that is separated distance. By distributing these scientific instruments over separated satellites, the formation flying enables to collect expected data that are unavailable from a single satellite, such as stereo views or simultaneously collecting data from a same ground scene in different angles.

It can be seen that formation is necessary to be performed for sake of collecting data effectively and efficiently.

1.3 Collective behavior

The collective behavior can be described as a group of robots working on a common job by cooperation. Compared with the meaning of formation, this definition is relatively wide and general since different collective behaviors depend on different common jobs. While formation can be treated as one approach or process to realize the final aimed job. For different objectives, the swarm robots would be assigned with different common jobs, for example, mapping, exploration, transportation, and so forth.

In this project, a specific collective behavior, that is object transportation, is mainly considered. This task requests multiple robots to cooperatively carry, push, or pull a common object to a specific area. Single robot has limited strength to move an heavy object, or keeps balance of an big object. These problems can be solved by using swarm robots. They can distribute the mass of an object and also they can change the place to hold an object so as to fit different terrains.

1.4 Background

Formation

The work in formation generation[TaS89] and keeping[BA98][Wan89] by swarm robots has been investigated by many researchers before.

A review of efforts exerted on the area of the multi-robot formation has been presented in [BA98] in detail. In the following, the background of formation is described briefly referring to the previous works.

An early application of artificial formation was to simulate the flocks of birds and schools

of fish for computer graphics. The dynamics and stability of multi-robot formation have drawn recent attention. Wang[Wan89] developed a strategy for robots' formation where individual robot is given a specific position to maintain relative to a leader or neighbor. Wang's analysis centered on feedback control for formation maintenance and stability of the resulting system. It did not include integrative strategies for obstacle avoidance and navigation. Chen and Luh[CL94] implemented the formation generation by distributed control. Large groups of robots cooperate to move in various geometric formations. Chen's research also centered on the analysis of group dynamics and stability and did not provide for obstacle avoidance.

There are other related paper of interest on formation control for robot team. [Par94] concerns the coordination of multiple heterogeneous robots and simulates a military movement, that is, one group moves a short distance while the other group overwatches for dangers. [EYM94] and [Yam97] investigate how robots can use only local communication to generate a global grouping behavior. Similarly, [Gag92] examines how robots can use local sensing to achieve group objectives like coverage and formation maintenance.

Based on the researches mentioned above, Tucker and Ronald did further contribution on formation control[BA98]. In their work, reactive behaviors for four geometric formations and three formation reference types are successfully simulated and implemented in two types of mobile multi-robot systems. The robot in one system is using 16 ultrasonic range sensors for hazard detection and controlled by onboard laptop computer running Linux. They communicate over a wireless network supporting Unix sockets via TCP/IP. The other robot is unmanned ground vehicle equipped with DGPS as position sensor, control computers and actuation devices for steering and speed control.

Collective behavior

Enabling swarm robots to move an object cooperatively has been a long-standing, yet difficult, goal of multi-robot systems. Many researches have dealt with this area. But, fewer of these projects have been demonstrated on physical robot systems.

Numerous variations on this task area have been studied, including constrained and unconstrained motions, compliant versus non-compliant grasping mechanisms, cluttered versus uncluttered environment, global system models versus distributed models. The most demonstrated task involving cooperative transport is the pushing of objects by multi-robot teams [DRJ95][SB93]. This task is inherently easier than the carry task, in which multiple robots must grip the common object and navigate to a destination in a coor-

dinated fashion[ZWN00][OKC96]. A novel form of multi-robot transportation that has been demonstrated is the use of ropes wrapped around objects to move them along desired trajectories.[Par00]

Most of previous works in this area only consider robots moving in a flat surface. A challenging issue is to implement the object transportation over uneven outdoor terrains.

1.5 Motivation

Research on swarm robotics has been on the rise during the last decade. A number of successful swarm robotic systems have already been developed and the study of coordination in swarm robotic system has become a hot topic of research.[sro]

In the project called *Swarm-bots*[sbt], an application of search and rescue in complex environments by swarm robots are successfully implemented. In the project called *Jasmine robots*[jas], a large number of robots have been built to investigate self-organization, emergent phenomena and so forth.

The research in this area is important to understand underlying principle of information and knowledge processing, adaptation and learning for very limited autonomous systems. These systems represent the result of miniaturization processes in such fields as robotics, micro- and embedded controllers, sensor networks, environmental monitoring, ubiquitous systems, medical and nano-technological research. Even today the autonomous microsystems are of interest for entertainment and toy industry.[jas]

By considering these merits in the research area resulting from swarm robots systems as well as only a few of physical swarm robots systems exist in the current world, it is suggested to start building a swarm robots system for Aalborg University (AAU).

1.6 Scope

In this project, micro mobile robots with limited sensors and processing ability are used. The robots had been designed by Trung Dung Ngo before this project was started. The objective to design such robots is to demonstrate distributed swarm robots behaviors on physical robot systems. Therefore, instead of designing robots from the beginning, the frame of this robot is taken into implementation for this project.

The robots are provided with perception by infrared sensors that are used both for distance

measurement and communication. An ambient light sensor and a color sensor are used to detect the light and distinguish the color, respectively. Each robot is built based on a chassis of a toy tank. The robot can be regarded as a two-wheeled driven mobile system. The wheels on each side are controlled by two motors, respectively. This actuation system provides the robot with maneuverabilities, including forward motion, backward motion and rotation. All the signals and data are processed by an onboard Microcontroller Unit (MCU). The MCU has the duty to make a decision on the robot's behaviors.

Based on the designed circuit diagram for robots, with knowing characteristics of controller, sensors and motors, proper strategies of negotiation and cooperation between multiple robots are feasible to be developed.

This project is conducted based on the robots mentioned above. After this project is finished, a new design robot has come out and the new circuit diagram is shown in Appendix A.2.

1.7 Objectives

In order to realize the swarm robots implementation, the objectives at this project are robots fabrication and test, algorithm development for formation and collective behavior, and algorithm validation by simulation and the physical robots system. In all, the tasks that have to be done in this project are presented in the following:

- Solder the PCBs for robots. Assemble the PCBs with actuation systems to construct the robots.
- Build a scenario with an object inside for running the robots with different aims.
- Test sensors and actuation systems. Modify the circuits to improve their functionalities if necessary, especially for the performance of communication and distance measurement.
- Develop algorithms for obstacle avoidance, light source approaching and color reaction during basic navigating.
- Develop an algorithm for formation generation and keeping along with obstacle avoidance.
- Develop an algorithm for a planned collective behavior.

- Validate algorithms by simulation and the physical robots system.

However, two points that have to be noted in this project are:

- There is no critical requirements defined when implementing robots' behaviors since these robots are still in the developing and testing phase. They are not sophisticated products and every functionalities achieved are limited by the current layout of hardware. Actually, one objective of testing the robots has to be conducted by adjusting the components in order to improve their functionalities as much as possible.
- The simulation is only used to test the logic in algorithms. The shape and size of robots defined in the simulation are not necessary to be the same as the real robots. Also, the sensors and the motion of robots are ideally configured in the simulation, which can not be achieved by current designed robots.

1.8 Contributions

This project make a progress on the practical research area of swarm robots rather than advanced theoretical research area. It starts a new swarm robots system, which can be further developed based on the contributions from this project. The main contributions are listed in the following:

- A novel design robot for distributed swarm robots system is introduced and implemented in this project. Each robot is cheap and micro. Even with limited size, each robot has several functionalities, including communication, distance measurement, light and color detection. During the project, suggestions in hardware configuration are proposed in order to improve the functionalities of the robots.
- The algorithm of formation is developed. This algorithm includes integrative strategies for obstacle avoidance and navigation, which are not considered in the previous work on formation. Furthermore, this algorithm is validated by simulation and the physical robots system.
- The algorithm of collective behavior is developed. The algorithm asks robots to play different roles in a collective task. The algorithm is much more valuable than the previous work that all robots perform the same role in a collective task. This algorithm is also validated by physical robots system.

1.9 Outline

Below, the remaining chapters of the thesis are listed and briefly summarized.

Chapter 2 Description

The schematics of the designed robots are described at the beginning of this chapter. Then, the scenario is introduced. The remainder of the chapter covers the platforms set for simulation and implementation, respectively.

Chapter 3 Hardware Functionalities

The realization of functionalities for a robot provided by hardware are described in this chapter. Firstly, two critical functionalities, communication and distance measurement are discussed deeply and a compromised configuration is decided. Then, light and color detection as well as maneuverability are addressed.

Chapter 4 Single Robot Behaviors

The algorithms for communication and distance measurement are generated firstly. Based on the above two algorithms, the algorithm for obstacle avoidance is developed by considering both wall and robots avoidance. In the end, two algorithms for light source approaching and color reaction come out, respectively.

Chapter 5 Formation

The formation of two robots is discussed in this chapter. According to the different approaches to keep formation, two cases are discussed and compared. The algorithms developed for each case are validated.

Chapter 6 Collective Behavior

The collective behavior is described in this chapter. One collective task is planned and realized by two subtasks. The algorithms are developed for each subtask and validated.

Chapter 7 Closure

The conclusion and future work are stated in the chapter.

Chapter 2

Description

In this chapter, the designed robots for this project are described concerning controller, sensors, actuation and power system. A scenario with an object inside is established for running the robots inside. The platforms for simulation and implementation are established.

2.1 Robot

The robot was designed by Trung Dung Ngo. The main usage of this kind of robot is to implement swarm robots' behaviors. Therefore, a group of robots have to be built and it requests each robot to be small and low cost.

The schematics of the robot is depicted in Figure 2.1. The Figure 2.2 shows an assembled robot viewed from front and right sides. The robot is mainly composed by two parts. The bottom part is a chassis, including the actuation system and supporting the top part. The top part is a Printed Circuit Board (PCB) mounted with components, mainly including a MCU and sensors. In order to use the space efficiently and keep the robot to be small, the diameter of the round PCB is only 6.4cm and all the components are compactly soldered on both sides of the PCB.

The robot is provided with five kinds of sensors. They are one color sensor, six ambient light sensors, one global IR receiver and six couples of local IR transmitter and receiver. The global IR receiver has not been implemented yet. The robot has an actuation system driven by two motors on each side. The MCU is in charge of all the components. By analyzing the data from sensors, the MCU functions as a control center to decide behaviors

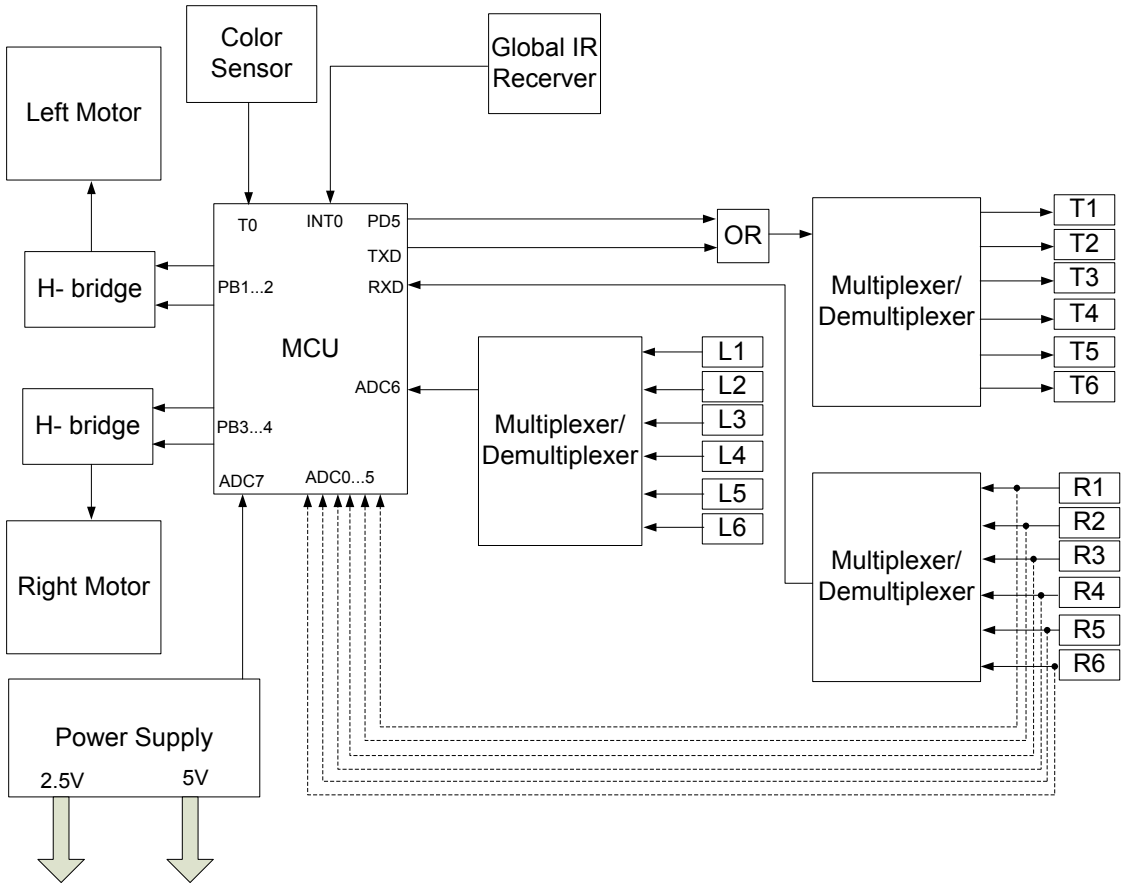
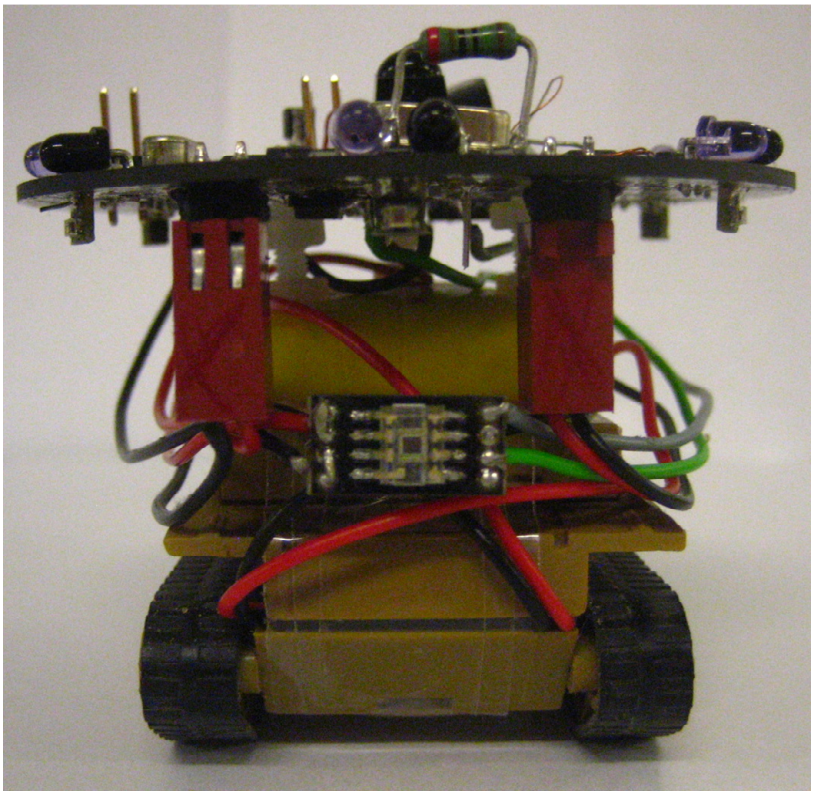
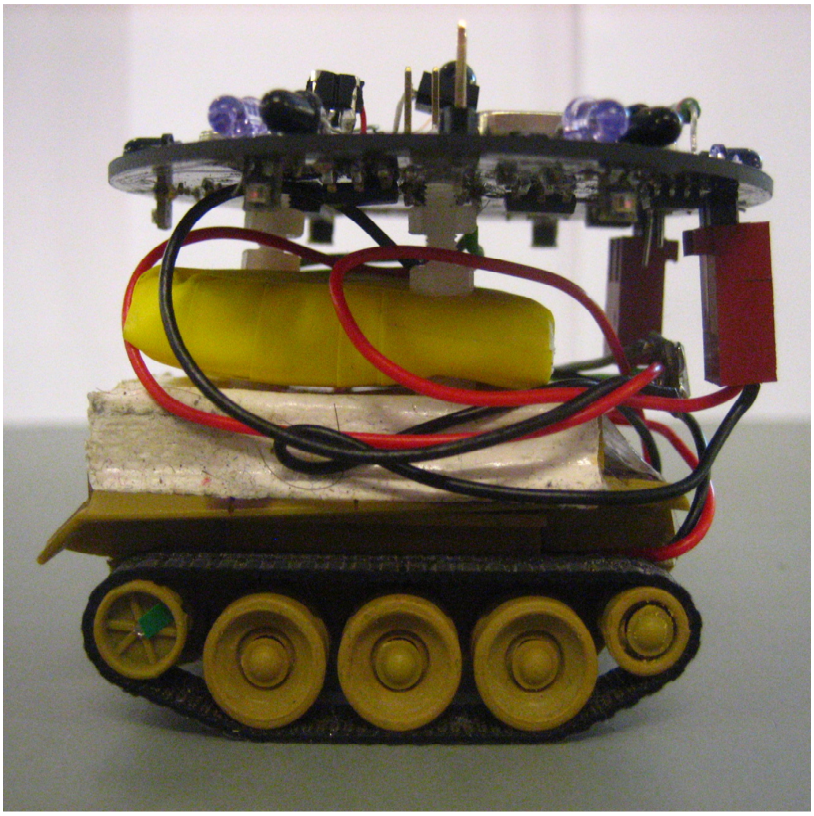


Figure 2.1: Schematics of the robot. **L** represents ambient light sensor. **T** and **R** are local IR transmitter and receiver respectively. The arabic numerals from **1** to **6** are used for identifying the sensors placed on different sectors.



(a) Front view of the robot.



(b) Right view of the robot.

Figure 2.2: An assembled robot.

of the robot, which could be realized by cooperation of two motors.

A power supply system is necessary to support the robot to be alive. But in this project, the power is taken from a power supply instrument. In expectation, the robot has to be equipped with a rechargeable battery.

In the following subsections, each part will be described in detail.

2.1.1 Controller

A MCU is selected to be the controller for the robot. The merit of highly integrated MCU drastically reduces the number of chips and the amount of wiring and PCB space that would be needed to produce equivalent systems using separate chips[mcu]. For the robot, the MCU is like the brain for the human. It needs to collect all the data detected from sensors and finally make a decision. Especially, the rules between data collection and decision making have to be memorized by the MCU.

Considering both memory and peripherals integrated in a MCU, **ATmega8L** from **Atmel Corporation** is used. The interested features are listed in the following[atm06]:

- Advanced RISC architecture
  - 130 Powerful instructions. Most are single-clock cycle execution.
  - 32 $\times$ 8 general purpose working registers.
- Nonvolatile program and data memories
  - 8K bytes of in-system self-programmable flash.
  - Endurance: 10000 write/erase cycles.
  - 1K byte internal SRAM.
- Peripheral features
  - Two 8-bit Timer/Counter with separate prescaler.
  - one 16-bit Timer/Counter with separate prescaler.
  - Three PWM channels.
  - 8-channel ADC with 10-bit accuracy.
  - Two-wire serial interface.
  - Programmable serial USART.

- I/O
  - 23 programmable I/O lines.
- Operation voltages
  - 2.7V-5.5V.
- Speed grades
  - 0-8 MHz.

The pins configuration of **ATmega8L** can be referred to Appendix A.

2.1.2 Sensors

Sensors give the robot abilities of perception and cognition. With different kinds of sensors, a robot can not only be conscious of environment, but also talk to the other robots.

The schematics of sensors' placement on the PCB is shown in Figure 2.3 and real layout of sensors can be seen from Figure 2.4. The round PCB is separated into six average sectors with angle of 60°. Six sectors are named by sequential numbers. The front sector is given the number 0 and the rest sectors are numbered clockwise. There are three sensors in each sector, including one couple of local IR transmitter and receiver and one ambient light sensor. The sensors on each sector are responsible for that specific direction. One color sensor is mounted on the front sector and one global IR receiver is fixed in the center of the PCB.

Six IR couples of transmitter and receiver

One each sector, there is one couple of IR transmitter and receiver mounted on the top of PCB. The IR transmitter is useing **SFH487** from **SIMENS**. This is an IR emitting diode with half angle of $\pm 20^\circ$ [sf4]. The IR receiver is using **SFH309FA** from **SIMENS**. This is an IR phototransistor with half angle of $\pm 12^\circ$ [sfh99].

The IR couple on one sector has to be in charge of that direction. It can detect the distance to another object in that direction. Moveover, the robot can do communication in that direction by using IR transmitter to emit signal and IR receiver to get signal.

In expectation, each IR couple can take the responsibility in an area with range of 60°. This ideal configuration can let the robot cover the area with 360° and also avoid the

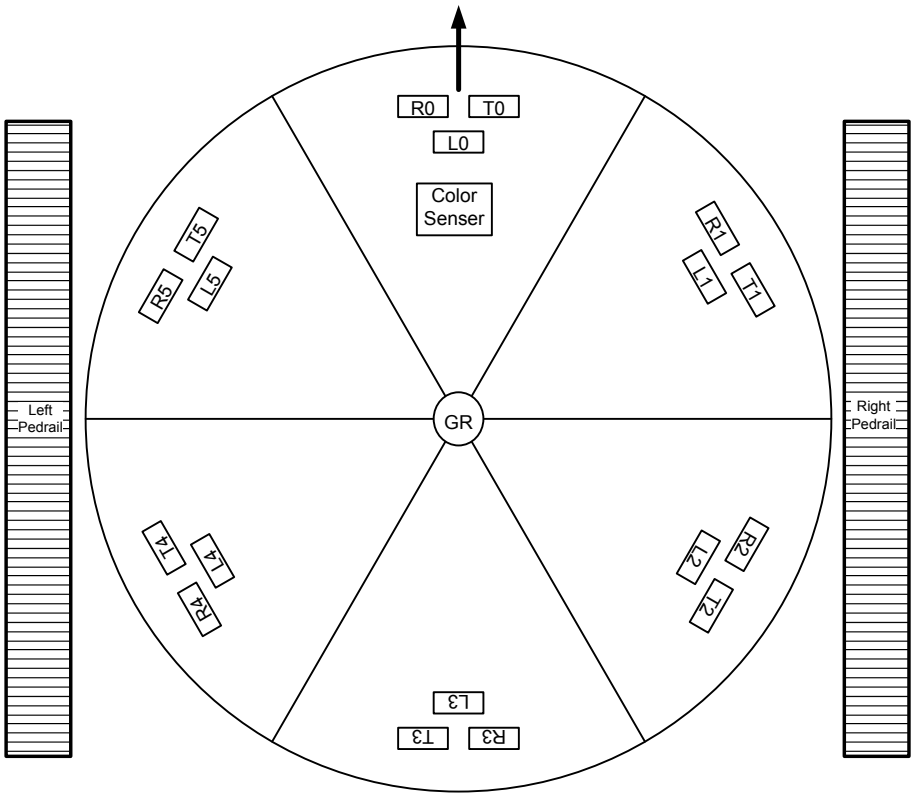
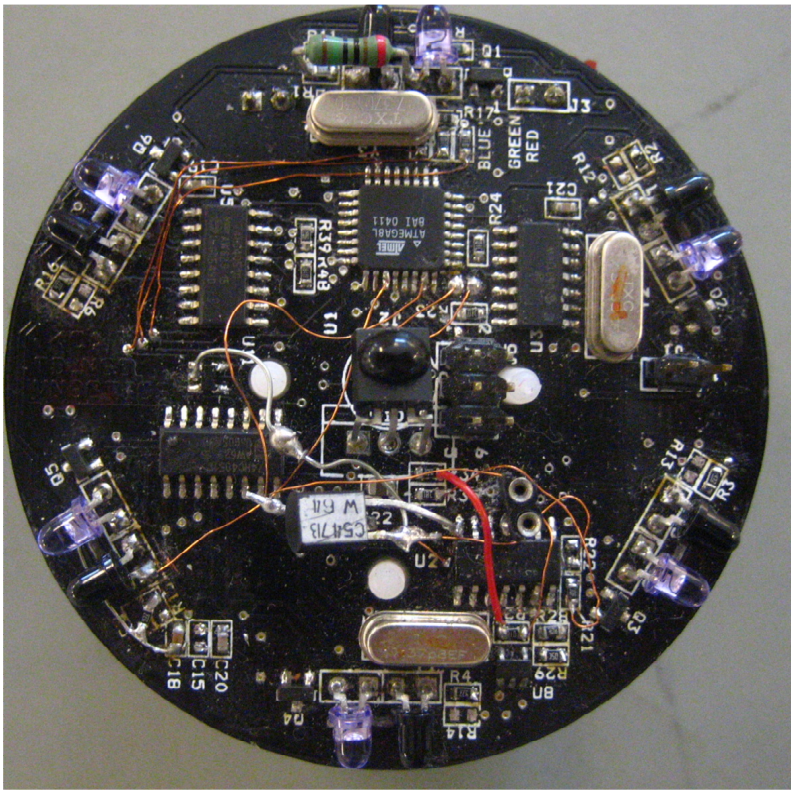
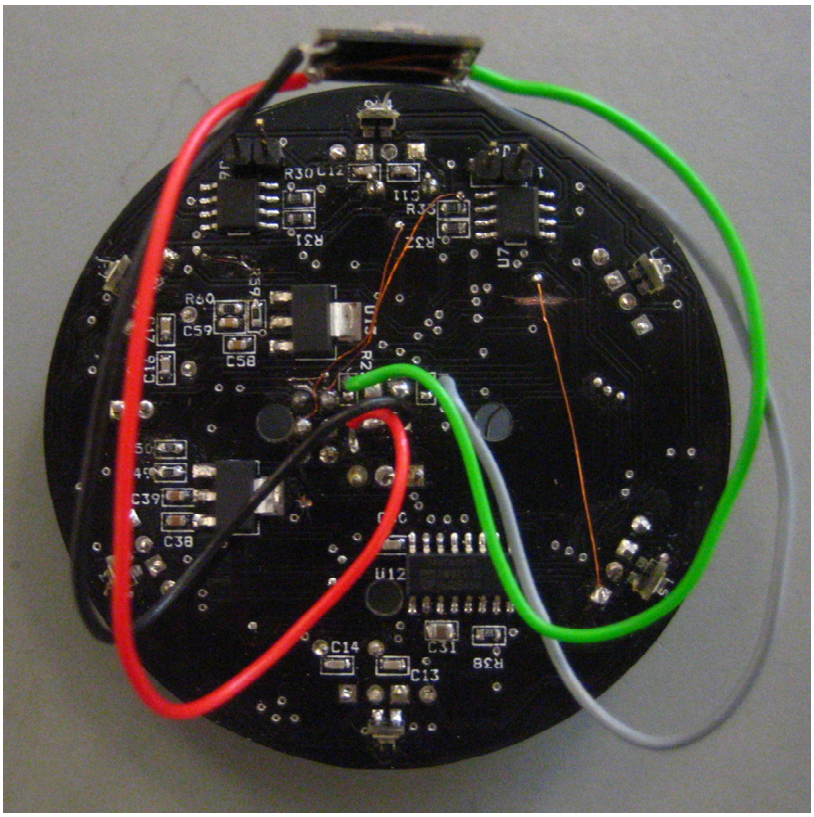


Figure 2.3: Schematics of a robot. **L**, **T**, **R** and arabic numerals from **1** to **6** are the same representations as in Figure 2.1. **GR** is the global IR receiver. The rectangles with horizontal lines represent left and right pedrails respectively. The arrow ↑ shows the forward direction of the robot. The sensors are placed on the both sides of PCB, which can be seen from Figure 2.4.



(a) Top view of the PCB. The global IR receiver and six couples of local IR transmitter and receiver are placed on this side.



(b) Back view of the PCB. The color sensor and six ambient light sensors are placed on this side.

Figure 2.4: PCB views.

interference from the neighboring IR couples. However, the validity of communication has be to firstly ensured. After several pairs of IR transmitter and receiver were tested by Trung Dung Ngo, the above IR couple was selected. Even though the selected IR transmitter and receiver only cover the range with angle of 40° and 24°, respectively, they can perform communication effectively and validly.

Six ambient light sensor

There is also one ambient light sensor on each sector. They are placed on the back side of PCB. The ambient light sensor, **APDS-9002** from **Avago**, is used to detect the light. This sensor consists of a spectrally suited phototransistor, which peaks in human luminosity curve. Hence, it provides an excellent responsibility that is close to the response of human eyes. It outputs the photocurrent whose value mainly depends on the irradiance. The size of this sensor is relatively small that is only $0.8mm \times 2mm \times 1.25mm$ [adp04].

One color sensor

There is a color sensor mounted in the front of the robot. The color sensor is **TCS230** from **TAOS**. This sensor is a color light-to-frequency converter which combines configurable silicon photodiodes and a current-to-frequency converter on a single monolithic CMOS integrated circuit. The output is a square wave with frequency directly proportional to light intensity. The light-to-frequency converter reads a 8×8 array of photodiodes. Sixteen photodiodes have blue filters, 16 photodiodes have red filters, 16 photodiodes have green filters, and 16 photodiodes are clear with no filters. The four types of photodiodes are interdigitated to minimize the effect of non-uniformity of incident irradiance. Which type of photodiode the device uses during operation can be selected by digital input pins. [tcs04]

2.1.3 Actuation system

The actuation system is a significantly important part to execute the behavior decided by the MCU. To build an actuation system needs to synthetically take into account of motor system, gear system and wheel system. These subsystems have to be combined together and fixed into a chassis suitably. Furthermore, the chassis also needs to be designed carefully to fit with all the other parts. Because these procedures are relatively time-consuming, a chassis including an actuation system is directly disassembled from a toy

tank. Thus, two pedrails of the tank are used to drive the robot. The height, length and width of the chassis are 2.8cm, 6cm and 4cm, respectively.

2.1.4 Power system

The expected power is supplied by a rechargeable battery. In the Figure 2.1, it can be seen that one ADC channel of MCU is connected with the power supply, which is used to detect the voltage level of the battery. This value can be used to calibrate the sensors' readings. However, the rechargeable battery has not be implemented yet. During this project, the power is taken from a power supply instrument since it is easy to check the voltage and current for the robot system in the developing phase.

In order to have a stable voltage supply, two voltage regulator chips are used. The chip is **LM317** from **National Semiconductor**. One is to regulate the voltage to be 2.5V for motors and the other one is to provide 5V for other components.

2.2 Scenario

On implementations, all robots are supposed to perform in a scenario and react with objects.

The boundary of the scenario is built by foams, which is used to emulate the wall that robots can not go over. It gives a limited area to run the robots.

A round object has been built, which is decorated by four LED arrays of different colors, including red, blue, yellow and white. All the LED arrays can be remotely controlled by using radio frequency. The wireless radio modules, **iDwaRF-168** from **chip45.com** are used to realize this function [idw]. This module combines a **Cypress 2.4GHz DSSS** radio transceiver with an **Atmel AVR ATmega168** MCU. One module is connected with an USB port of PC by using an USB-to-UART interface which has been integrated in the module of **iDwaRF-HubBoard**[lit]. The other module connects to a PCB which was designed in order to switch the LED arrays. After being programmed, the object can change the light colors by entering commands on the PC side. The detailed commands can be found in Appendix B.

This object with different color lights can be used to play different roles according to the specific robots' behaviors. For example, the red light could simulate the food that robots want, the blue light could be the enemy that robots has to be escaped from, the white light

could be treated as the robots' home, and the yellow light or no light could represent an obstacle that robots need to avoid. Using the remote control, the object's lights can be quickly changed , which is useful to simulate an accidently changed environment and test the robots' responses.

2.3 Simulation platform

The simulation platform is established based on Jasmine simulator 7 version [jas]. There are several attractive features in Jasmine simulator.

- A scenario can be created easily.
- An object of the robot has been already created.
- An object of IR sensor has been already created.
- A robot can be controlled to move or rotate at different speeds.
- A multi-robot system can be created and each robot's behaviors can be decided independently.
- Communication between robots is available.

The robots can be created to be 3D. The realization of the simulation is based on Breve, which is an open-source 3D simulation environment with an OpenGL engine. The programming language is Steve, which is an oriented-object language. All the simulation programs are developed with Breve 2.5.1 version[bre].

In simulation, the motions of robots and communication between robots can be ideally configured. No accident would happen if the algorithms were correct. In real experiments, the behaviors of robots are much influenced both by its own and environmental factors. Therefore, the main objective to do simulation is to test and improve the logic inside developed algorithms. However, the algorithms of behaviors which request the robots to react on light or colors are not simulated.

2.4 Implementation platform

To implement the algorithms on the real robots, there are two recursive procedures, including programming and debugging.

Programming platform

In order to realize different robots' behaviors, the corresponding algorithms have do be developed and proved in the simulation firstly. Then, the validated algorithms will be written in embedded C language based on the type of MCU. The C programs are complied by GCC and downloaded to the MCU by In-System Programming (ISP). These procedures are indicated in Figure 2.5. The **AVR Stdio** takes on the jobs of compiling and downloading. The interface between USB and ISP is validated by a programmer **CrispAVR-USB**[cri].

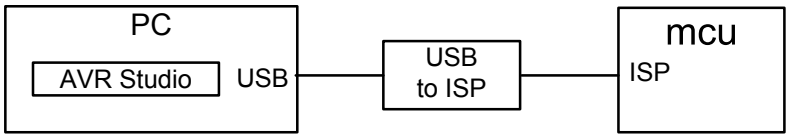


Figure 2.5: Programming environment.

Debugging platform

A debugging system can be used to adjust the programs. It is necessary to read the data out from the robots. By observing and analyzing the actually data in the robots' memories, the programs and the algorithms can be both improved to fit the real situations.

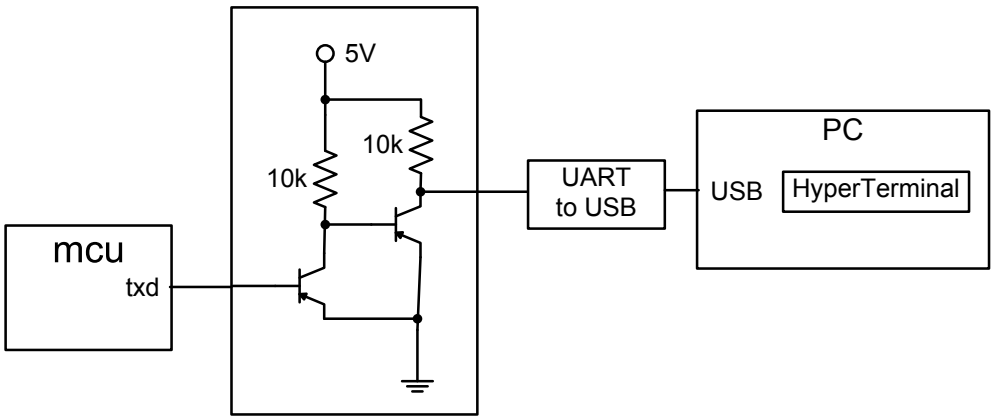


Figure 2.6: Debugging environment.

The debugging platform is established according to Figure 2.6. The interested data will be configured to output from txd pin of MCU. However, during the experiments, the high level voltage is only $1.5V$, which can not be recognized by the UART-to-USB converter. Thus, a circuit shown in the second block of Figure 2.6 is used to pull up the high level

signal to be $5V$. This result is indicated in Figure 2.7. In the end, the data can be read out from the **HyperTerminal** on the PC side.

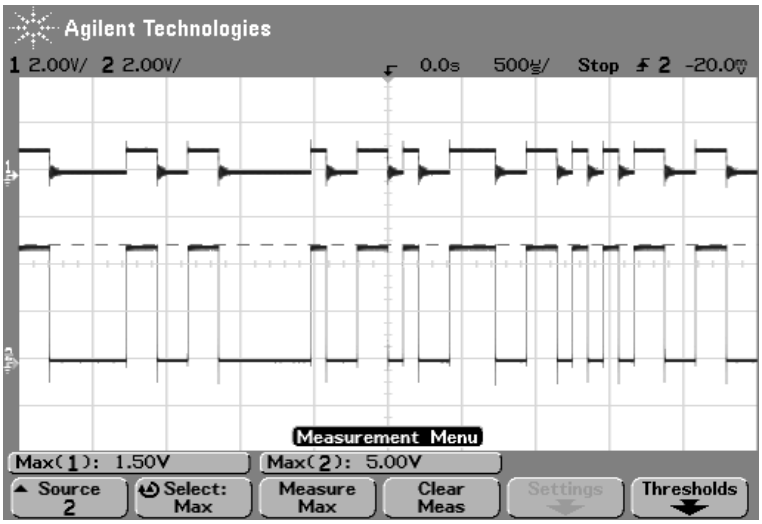


Figure 2.7: The upper one shows the signal directly output from txd pin of MCU. The lower one shows the high level signal is pulled up to be $5V$ from $1.5V$.

2.5 Summary

In this chapter, the robot used in this project was described in Section 2.1, including controller, sensors, actuation system and power system. However, The global IR receiver has not been implemented and a rechargeable battery has not been fixed yet.

In Section 2.2, a scenario was built by foams and an object with four LED arrays was built in order for playing different roles in the scenario. This object was programmed and could be controlled remotely by using radio frequency.

In the Section 2.3, the simulation platform was established based on Jasmine simulator. The main aim of doing simulation is to test the logic in algorithms.

In the Section 2.4, the implementation platform was described, including programming and debugging platforms.

Chapter 3

Hardware Functionalities

Based on the current hardware design, it provides the robot with several functionalities, including communication, distance measurement, ambient light detection, color detection and Maneuverability. Every functionality is investigated by doing experiments in the laboratory. In this chapter, the realization of each functionality related to the specific hardware is introduced and the results from experiments are discussed. Suggestions are given in the end.

3.1 Communication

In order for running swarm robots rather than a single robot, the main functionality is that one robot can communicate with the other robots. From communication signal, they can know what the other robots are doing and decide what they have to do. Therefore, communication between robots is the most significant functionality need to be realized above all.

For this designed robot, communication is realized by six couples of local IR transmitter and receiver with Universal Asynchronous Receiver/Transmitter (UART). The layout of six couples and the characteristic of the used IR transmitter and receiver have been described in Section 2.1. In this section, the realization of communication between robots will be described.

3.1.1 UART

UART is commonly used for embedded systems communications. It is useful to communicate between MCUs and also with PCs. The debugging platform was setup by using UART. The PC asynchronously receives data from the MCU and then displays in the **HyperTerminal**.

Between MCUs on robots, UART was also used as the approach to build up the communication. The six couples of local IR transmitter and receiver are the wireless bridges to load signal and let signal pass through.

The frame format used for communication is that one start bit plus eight data bits plus two stop bits. It is indicated in Figure 3.1. The IDLE line is always high. The start bit is always low. The next eight data bits is either low or high depending on the information included in the data. The final two stop bits are always high, which provide an interval for receiving the next start bit. Two stop bits used here give more time for the interval and also the second stretched stop bit helps resynchronization. The UART recovers character timing information from data stream by using designated start and stop bits to indicate the framing of each character. [usa]

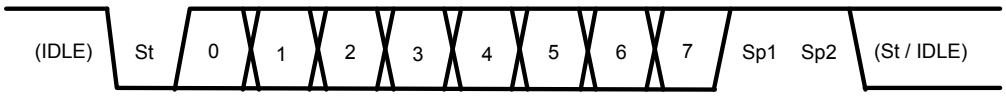


Figure 3.1: The frame format of UART. **St** represents start bit. **0** to **7** represent eight data bits. **Sp1** and **Sp2** represent two stop bits. **IDLE** represents no transfers on the communication line(RXD or TXD).

3.1.2 Communication by IR couples

Before to choose the baud rate of the UART, this subsection describes how IR couples work for communication. The Figure 3.2 indicates the schematics of one couple of the local IR transmitter and receiver. The IR transmitter and receiver can be treated as mounted on one robot or separately on two robots because of identical design. There have to be other five couples connecting with the inputs of multiplexers and ADC channels of MCU. Only one couple is clearly shown in Figure 3.2 due to their same functions.

On the IR transmitter side, the MCU sends out a signal from the TXD pin that is connected to the base of a N-channel transistor. The transistor is pulled up in the collector. Then the signal can make the transistor output high or low signal to an OR gate. The function of

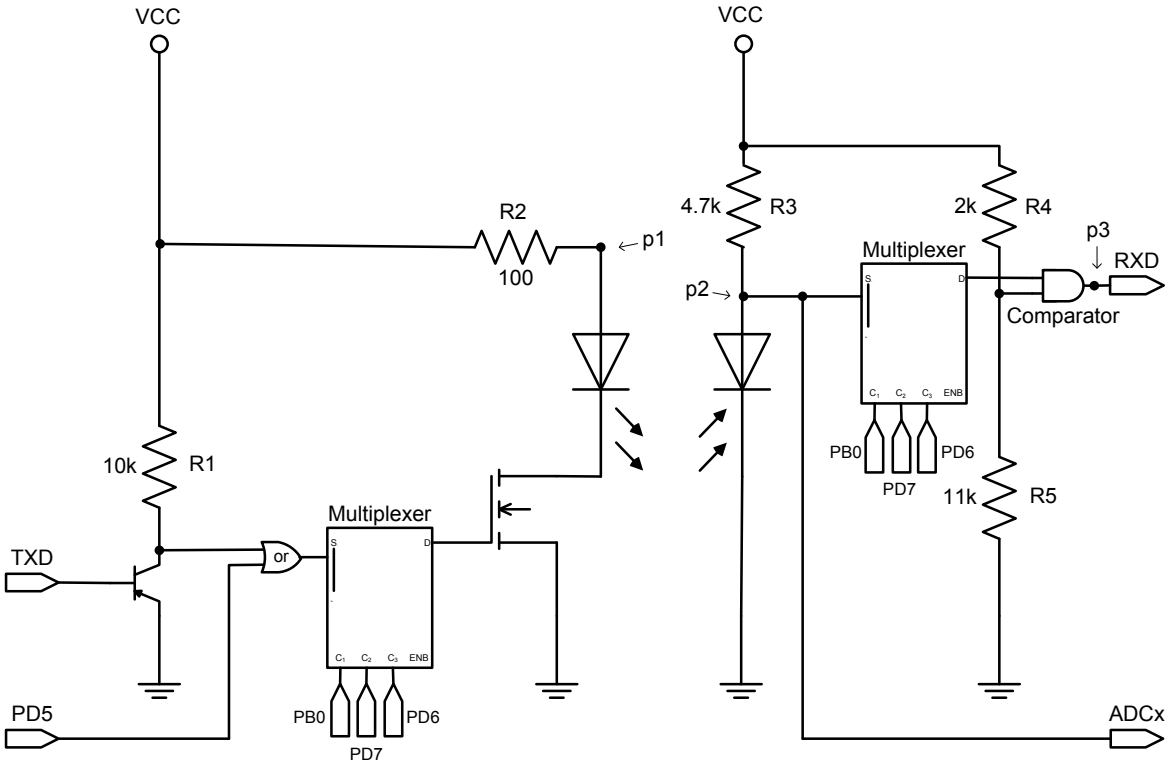


Figure 3.2: Schematics of one couple of the local IR transmitter and receiver for either one or two robots. I/O pins connect to the MCU. There should be six of IR couples in all, but not drawn out repeated. The ADCx represents ADC channels 0 to 5 of MCU.

the OR gate is to switch between communication mode and distance measurement mode depending on the output of PD5 from the MCU. If PD5 outputs low, so the output side of OR gate is decided by the input originally from TXD, that is communication signal. By setting digital outputs PB0, PD7 and PD6 from MCU, a multiplexer determines which one of six channels allows the signal to pass through. The output from the multiplexer switches a N-channel digital FET to be either on or off. If the FET is on, there will be current going through the IR transmitter, which sends out an IR signal. If the FET is off, no IR signal will be sent out.

On the IR receiver side, if no signal were received, the output to the multiplexer would be as high as the VCC value. If the IR receiver receives signals, there would be photocurrent going through. Therefore, the output voltage value to the multiplexer will be lower than the VCC value, which is depending on the intensity of the received IR signal. The multiplexer functions the same as one on the IR transmitter side. Before the output of multiplexer enters to the RXD pin of MCU, it needs to be trimmed firstly by using a comparator. The comparator compares the output from the multiplexer and reference voltage level and then outputs a binary state signal, one or zero. Finally, a stream of binary signal that need to be recovered will enter the UART part of MCU .

The Figure 3.3 shows the signals detected in two points, p1 and p2 labeled in Figure 3.2, which are on the transmitter and receiver side, respectively. The data used for testing is ”01011010” in binary. Packing with one start bit and two stop bits, the whole package is ”00101101011” which is identical to the signals shown in the Figure 3.3. When doing this testing, one robot’s IR transmitter closely points to the other robot’s IR receiver and the used baud rate is $9600bps$.

3.1.3 Communication speed

In this subsection , one critical factor is discussed based on the hardware design, which is communication speed decided by baud rate.

If the baud rate is higher, the robots can spend less time on communication. Therefore, during one specific time interval, there are more chances to talk to each other. According to the information included in the received data and their own states during every time interval, the robots can response immediately.

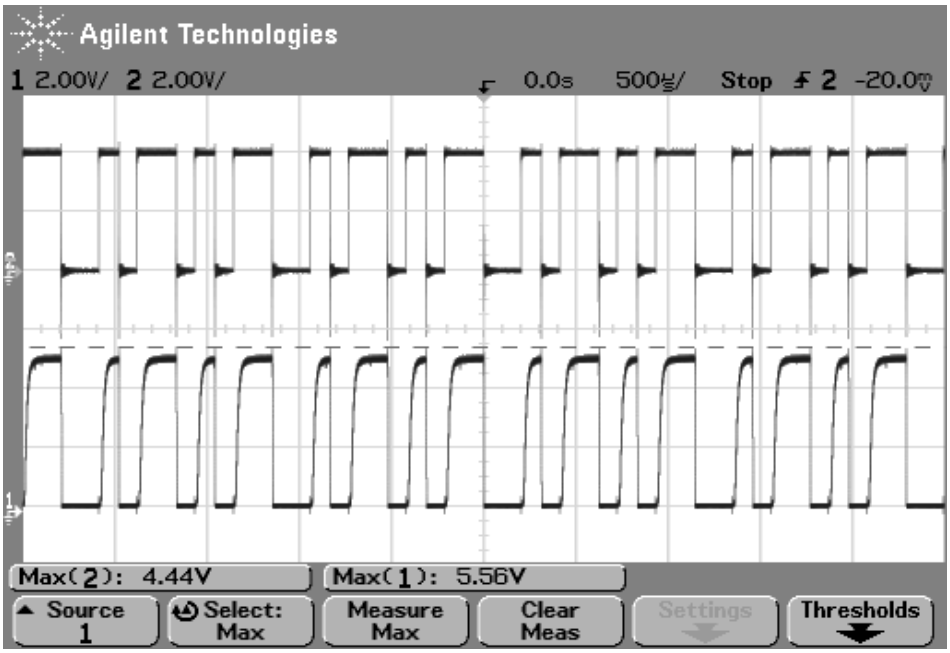


Figure 3.3: Transmitted and received signals. The testing data for communication is "01011010". The upper signals with GND 2 is detected at the p1(Figure 3.2) on the transmitter side. The lower signals with GND 1 is detected at the p2(Figure 3.2) on the receiver side. The baud rate is 9600bps.

Crystal oscillator selection

In order to ensure the validity of data transferred by UART, a suitable crystal oscillator with frequency of 7.3728MHz is selected. Referring to the datasheet of ATmega8L [atm06], with this frequency for system clock source, the error between the actual generated baud rate and the target baud rate keeps 0.0% from baud rate 2400bps to 230.4kbps whenever in normal or double speed mode. The max speed grade of ATmega8L could reach to be 8MHz, but there exist errors in most baud rate selections except 250kbps and 0.5Mbps. Because baud rate is important in synchronization, it is worthwhile to low down 7.84% speed for whole system but get the benefit of no error in baud rate.

Double speed Operation

With the accurate baud rate setting and system clock, the transfer rate can be doubled by setting UART running in double speed mode. Thereafter, the highest baud rate can be 1Mbps[atm06]. However, the receiver in this case use half the number of samples for data sampling and clock recovery. For the transmitter, there are no downsides.

Asynchronous data reception

Before to select a suitable baud rate, it is introduced firstly that how to recover asynchronous data [atm06]. There are two step to recover data. First step is to recover asynchronous clock which is introduced in detail in [atm06]. And the second step is to recover asynchronous data which is more related to the baud rate selection.

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has eight states for each bit in double speed mode. Figure 3.4 shows the sampling of the data bits. Each of the samples is given a number that is equal to the state of the recovery unit.

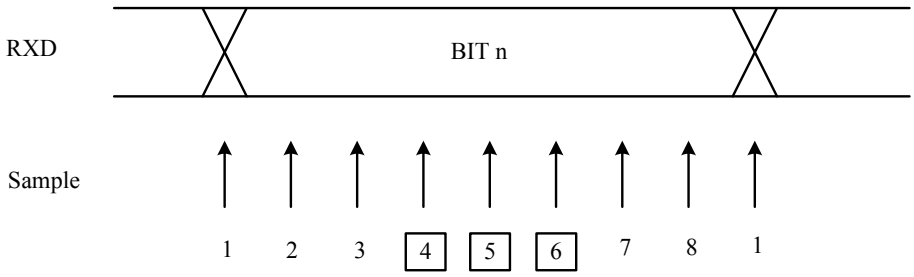


Figure 3.4: Sampling of data.

The decision of the logic level of the received bit is taken by doing a majority voting of the logic value on the three samples in the center of the received bit. The center samples are emphasized on the Figure 3.4 by having the sample number inside boxes. the majority voting process is done as follows: if more than two samples have high levels, the received bit is registered to be a logic 1. On the contrary, it is 0. The recovery process is then repeated until a complete frame is received.

Analysis of signal transfer

From Figure 3.3, it is obviously to see that there is a rising process for the IR receiver when changing from signal received to no signal received. The Figure 3.5 is the zoom in of the Figure 3.3 at -402μs to more clearly show the delay and rising time for only one bit transfer. The delay time is defined as the error from the rising time of the upper signal to the time of the lower one beginning to rise. The delay time is around 12μs. And the rising time is defined as the time spent on rising from 10% to 90% of top value, which is around 20μs.

There is also a falling process for IR receiver when changing from no signal received to

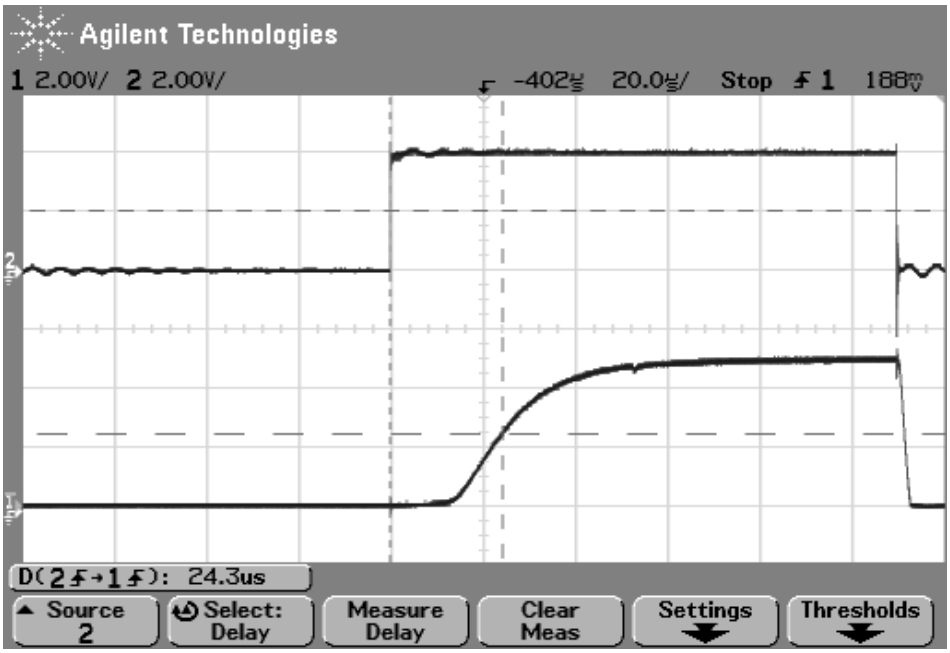


Figure 3.5: Zoom in of Figure 3.3 at -402μs. The upper signal with GND 2 is detected at the p1(Figure 3.2) on the transmitter side. The lower signal with GND 1 is detected at the p2(Figure 3.2) on the receiver side. The delay time is around 12μs and the rising time is around 20μs.

signal received. Seen from the Figure 3.5, it is estimated to be 2μs. This time is much shorter than the time spent on the rising process. Thereafter, only the rising process is considered to select the baud rate.

Figure 3.6 shows the received and trimmed signals. The lower signal has been trimmed to be a stream of binary states which inputs to the RXD of MCU. The MCU recovers the original data completely based on this binary stream. Therefore, it is critical to make sure the time sequence of trimmed binary states to be close to the transmitted signal.

Figure 3.7 is the zoom in of Figure 3.6. It is more clearly to show that the comparator outputs high when the input is higher than the reference value which is defined to be 0.846VCC. Otherwise, the output is low. The error time, which defines the time from the upper signal beginning rising to the lower signal changing to high, is around 25μs.

Baud rate selection

From the above analysis, the total delay time, which is defined from the rising of the transmitted signal to the rising of the trimmed signal, is around 37μs.

The Table 3.1 lists the one bit transfer time and the error according to different baud rates.

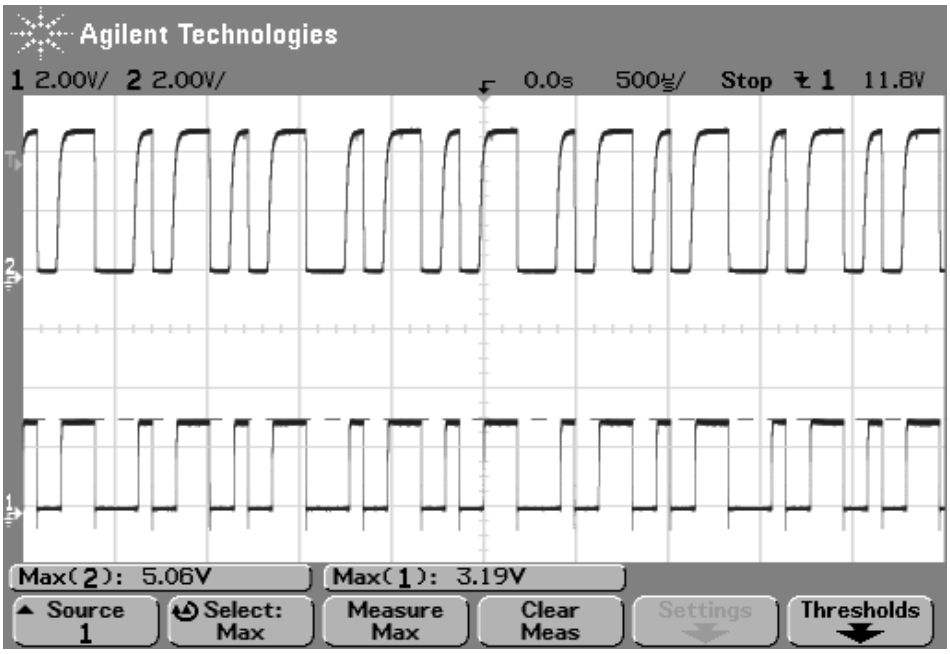


Figure 3.6: Received and trimmed signals. The useful data received is ”01011010”. The upper one with GND 2 is the received signal detected at the p2(Figure 3.2). The lower one with GND 1 is the signal trimmed by the comparator and detected at the p3(Figure 3.2). The baud rate is 9600bps.

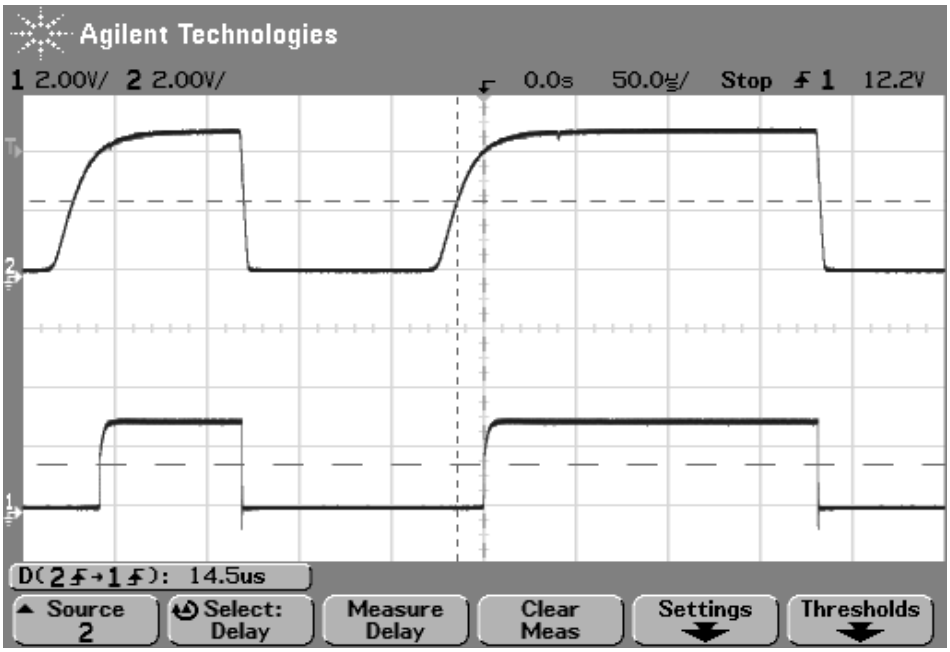


Figure 3.7: Zoom in of Figure 3.6 at 0μs. The upper one with GND 2 is the received signal detected at the p2(Figure 3.2). The lower one with GND 1 is the signal trimmed by the comparator and detected at the p3(Figure 3.2). The error time is around 25μs.

Table 3.1: Comparison of characteristics in different baud rates.

Baud Rate (bps)	One bit transfer time (μs)	Error (%)
2400	417	0.0
4800	208	0.0
9600	104	0.0
14.4k	69	0.0
19.2k	52	0.0
28.8k	35	0.0

Considering that the logic value is decided by majority voting on three centering samples, it has to ensure at least two centering samples are correct after the delay time. This rule written in Equation 3.1 can be used to judge a baud rate is suitable or not. The T_{obt} represents the time of one bit transfer.

$$\frac{4}{8} \times T_{obt} > 37\mu s \tag{3.1}$$

Substitute the T_{obt} by the one bit transfer times listed in the Table 3.1. Only three is satisfied, which are $2400bps$, $4800bps$ and $9600bps$. It is evidently to choose $9600bps$ since it is the highest one in the three choices.

There are two ways can be used to increase the baud rate according to the current hardware layout. One is to decrease the reference voltage of the comparator to make shorter of time error shown in Figure 3.7. The other is to replace the resistor R3 shown in Figure 3.2 by a lower one. It can let the rising process be finished in shorter time. But as a tradeoff, increasing the baud rate would sacrifice the capability of communication distance.

3.1.4 Communication distance

The communication distance is another important factor need to be considered. In most cases, it prefers that the robots can communicate with each other with longer distance. However, the robots' ability of long distance communication can bring one drawback when a large number of robots are implemented in a scenario. One robot will be easy to receive many interference signals from several robots as long as they stay in the range of communication distance. This subsection discusses the communication distance related to the hardware design.

An experiment was setup to let one robot's IR transmitter directly point to the other robot's IR receiver. This experiment is repeated three times with setting different distance between two robots. The Figure 3.8 shows the results. It can be seen that, the farther distance, the higher of low level voltage of the received signal. The high level voltage of the received signal always keeps the same as VCC.

This received signal needs to pass through the comparator firstly. The comparator would output zero state only if the low level of the received signal is lower than its reference voltage. Thus, if the low voltage level of received signal that could be recognized by the comparator is higher, the longer communication distance would be realized. However, referring to the datasheet of comparator [lmv99], the input voltage range is up to $4.2V$ with $5V$ DC supply since there has to be voltage drop for transistors inside the comparator. Therefore, the reference voltage value of the comparator is set to be $4.23V$ calculated by Equation 3.2. Even though there is $0.03V$ higher than the suggested value, it performs with no error on the real implementation.

$$V_{ref} = VCC \times \frac{R5}{R4+R5} = 5 \times \frac{11k}{2k+11k} = 4.23V \tag{3.2}$$

There is also another component influencing the communication distance, that is R3 shown in Figure 3.2. The voltage of received signal, V_{p2}, can be computed by Equation 3.3.

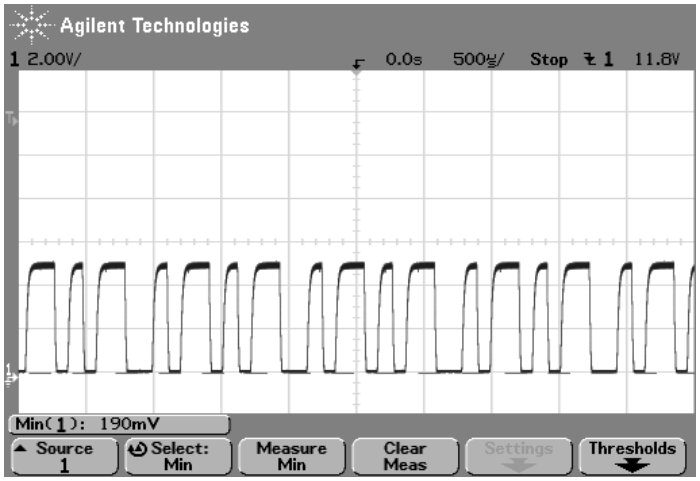
$$V_{p2} = VCC - I_{CE} \times R3 \tag{3.3}$$

where I_{CE} is the photocurrent linearized with the irradiance of received IR [sfh99].

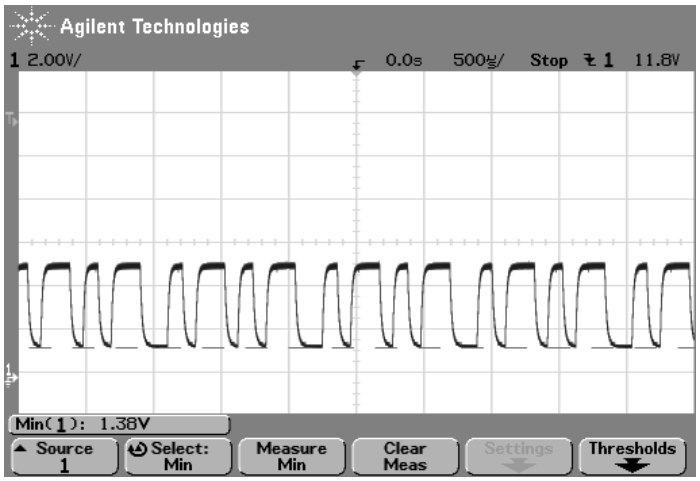
When no IR were received, the V_{p2} would be equal to VCC. If there were IR received, the V_{p2} would be lower than VCC depending on the part of $I_{CE} \times R3$. Assuming the IR receiver is exposed to the constant irradiance of IR signal, the higher resistance of R3, the lower V_{p2}. Similarly, for the same distance, increasing the resistance of R3 leads to the lower of V_{p2}. It can also infer that the longer communication distance can be achieved by a higher resistance of R3.

3.1.5 Summary

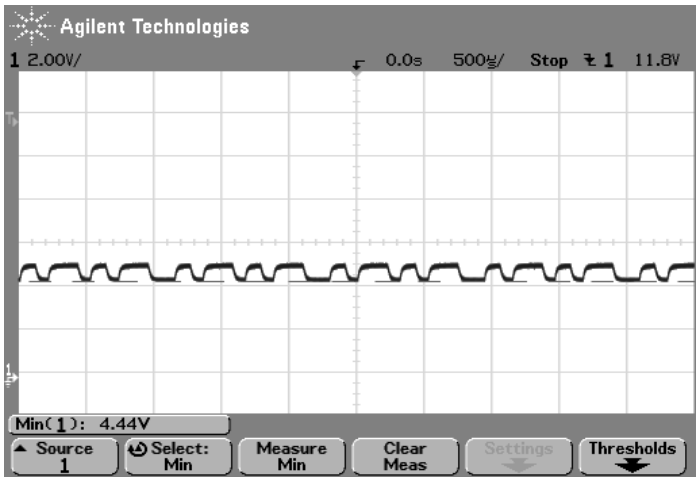
In a short summary, longer communication distance needs higher values of reference voltage and resistance of R3. This is opposite to the communication speed which requires lower value of reference voltage and resistance of R3. The specific values selected for R3, R4 and R5 shown in Figure 3.2 are compromised configurations. The communication



(a) When two robots are close.



(b) When two robots are near.



(c) When two robots are far.

Figure 3.8: These three experimental results show that the low voltage level of the received signal detected at p2(Figure 3.2) is related with the distance from the other robot who transmits signal.

speed is set to be 9600bps and the communication distance is around 20cm.

3.2 Distance measurement by reflected IR

Another important ability for a robot is to detect the distance. With knowing the distance to either obstacles or other robots beside itself, the robot can perform to avoid them or keep specific distance from them. It is a basic functionality.

The robot is equipped with IR couples to do distance measurement. The IR couples are shared by doing communication, but used during different time.

With IR couple, one approach to measure the distance is to counter the time spent between emitting IR signal and receiving the reflected IR signal. It do not need to consider the energy dissipation. Even though the energy of received signal is quite low, what only needs to note is that the received signal has to be the original emitted one.

Considering the MCU of this robot, the system clock is 7.3728MHz. When the clock source of Timer/Counter is set as the system clock with no prescaling, the maximal resolution of the Timer/Counter is 0.13563μs. During this period of time, an IR signal can transfer 40.66m calculated by Equation 3.4.

$$0.13563\mu s \times c = 0.13563\mu s \times 299,792,458m/s = 40.66m \tag{3.4}$$

where c is the speed of light.

So the maximal resolution of the distance can be detected by IR on the robot is half of 40.66m, that is 20.33m. This number is not suitable for the robot's implementation.

There is a second way to measure the distance with IR. The energy of received IR signal is always lower than the transmitted one, which has been dissipated due to its transferred distance as well as the surface, color and material of the reflector. For example, a rough surface is good at dispersing the light and a dark color is easier to absorb light than a light color. Therefore, there should be different rules for different reflectors. If the reflector is a specific one, the distance is possible to be estimated by detecting the energy of reflected IR signal.

3.2.1 Distance measurement by IR couples

The distance measurement is realized by using the same hardware part as to realize communication. Referring to Figure 3.2, the approach to do distance measurement is introduced from the sides of IR transmitter and receiver, resectively.

On the transmitter side, if PD5 is set to be high, both the OR gate and the multiplexer output high. Thus, the N-channel FET is always on and the IR transmitter keeps emitting IR signal.

On the receiver side, according to the Equation 3.3, when the receiver gets continuous IR signal, the IR phototransistor would generate photocurrent whose level is depending on the irradiance of received IR signal. With assumption that the IR couple is pointing to a specific reflector, it can say the irradiance is related to the distance from the reflector. The level of the V_{p2} is read by the ADC part of MCU with 10-bit accuracy.

As the similar to the discussion on communication distance, when detecting the same distance, a higher resistance of R3 can lower down the level of V_{p2}. It also implicates that the robot can measure longer distance if increasing resistance of R3. But the side effect is that the communication speed has to be lower down. The actually visible distance the robot can see is depending on the different reflectors.

3.2.2 Distance of one robot's vision

In this subsection, the measurable distance for a robot is discussed on two parts, including distance to the wall and distance to another robot.

Distance to the wall

The foam was used to build the boundary of the scenario. The robot treats the foam as the wall and it is important for the robot to know how far away from the wall. An experiment was set by directly pointing one IR couple of a robot to the wall and the incident angle is $0°$. The Figure 3.9 shows the relation between the distance to the wall and the ADC value converted from the voltage level of received reflected IR.

It can be seen from Figure 3.9 that the detectable distance is less than $10cm$. If the distance is larger than $10cm$, the curve is still rising slowly but undistinguishable. The Equation 3.5 shows the piecewise defined function disjointed by distance $10cm$. If the distance, d, is less than $10cm$, the ADC value can be computed by a six-order function. If the distance

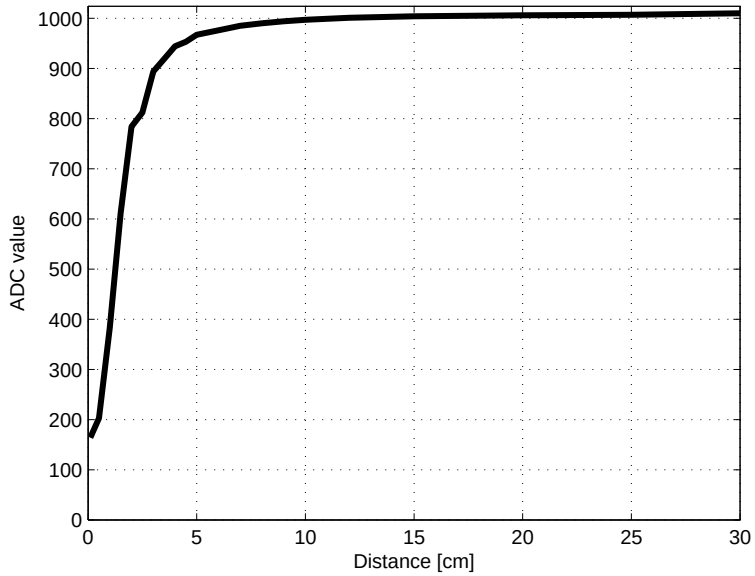


Figure 3.9: The ADC value from reflected signal versus the distance to wall.

is larger than $10cm$, the ADC value is above 1000 but less than 1023 that is the maximum the 10-bit ADC can reach. In the latter case, the ambient light would influence on the IR receiver, so the ADC value could change between 1000 and 1023.

$$ADC = \begin{cases} 0.036d^6 - 1.2d^5 + 16d^4 - 88d^3 + 171d^2 + 208d + 110 & \text{if } d \leq 10 \\ [1000, 1023] & \text{if } d > 10 \end{cases} \tag{3.5}$$

Distance to another robot

The situation is different when one robot meets another robot from the wall. Referring to Figure 2.2, it can be seen that the IR couples are mounted on the top of the robots. There is not enough reflector can be used to reflect signal since no big surface exists in the height of IR couples. There exists infinite relative positions of two robots when they meet each other. So the reflected signal would have different voltage levels even they keep the same distance. In the following, two main relative positions are analyzed.

The first one is that one robot's IR couple points to the IR couple of the other robot. This also means the transmitted IR signal would reflect from the IR couple of the other robot. The result is shown in Figure 3.10(a). It can be seen that the detectable distance is less than $4cm$.

But in reality, the chance that one IR couple uses the other robot's IR couple as reflector is relatively small. In most cases, the IR couple of one robot only points to a point in the range between two IR couples of the other robot. So the reflector would be a combination of small components dispersedly mounted on the PCB and the surface of PCB which is unfortunately parallel to the IR couple. The Figure 3.10(b) shows the experimental result in this situation. It can be seen from Figure 3.10(b) that the robot has an opportunity to see the other robot only if their distance is less than $1cm$. But even when they touch each other, the ADC value still reaches to the height of 974. The noise from the ambient light is easy to reach to this value.

Combining the analysis of above two situations, it can conclude that it is not available to measure distance to the other robot by using reflected IR signal.

3.3 Distance measurement by communication signal

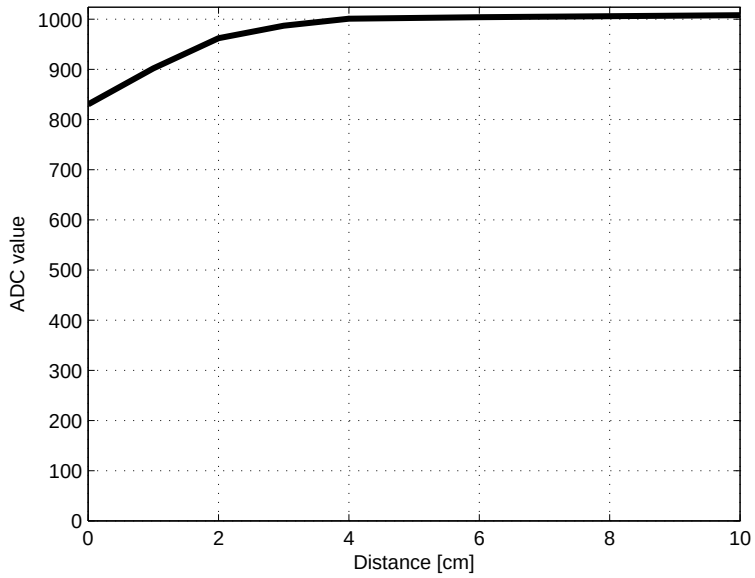
From the analysis in the last section, it has got a conclusion that it is impossible to measure the distance to the other robot by detecting the voltage level of reflected signal. But from the analysis of communication distance in Subsection 3.1.4, it can been seen that the low level voltage of received communication signal is related to the distance between robots. Compared to the reflected signal, the received communication signal has to be much stronger due to no dissipation on reflectors. So it can ensure the measurable distance to the other robot to be as long as the communication distance, that is $20cm$.

An experiment was set up as similar as doing the experiment for communication distance. One robot transmits signal and the other robot receives signal. The IR transmitter and receiver keep pointing to each other straightly. The relation between the ADC value converted from low level voltage of received communication signal and the distance between them is shown in Figure 3.11.

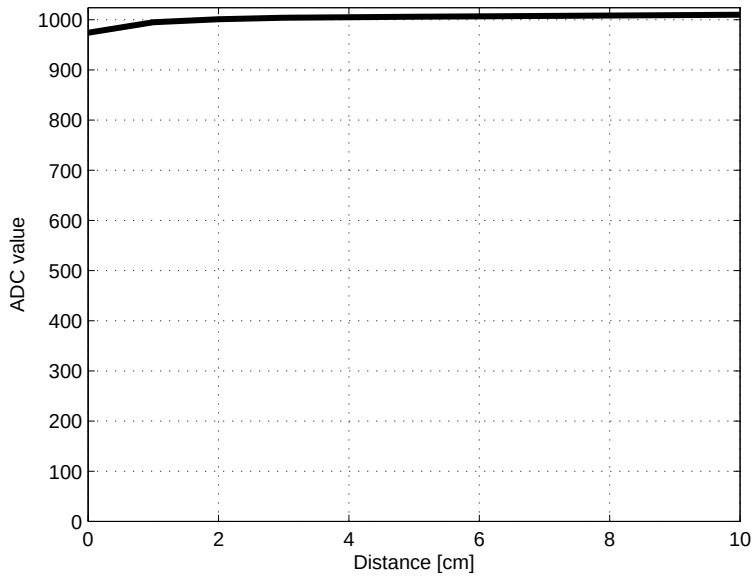
It can be seen from Figure 3.11 that detectable distance to the other robot reaches to $20cm$. The piecewise defined function to show the relation is written in Equation 3.6.

$$ADC = \begin{cases} -0.0019d^6 - 0.11d^5 - 2.6d^4 + 25d^3 - 97d^2 + 123d + 8 & \text{if } d \leq 20 \\ [1000, 1023] & \text{if } d > 20 \end{cases} \tag{3.6}$$

Where ADC is the ADC value and d represents the distance.



(a) When the robot's IR couple points to the other robot's IR couple.



(b) When the robot's IR couple points to the the middle place between two IR couples of the other robot.

Figure 3.10: The ADC value from reflected signal versus the distance to another robot.

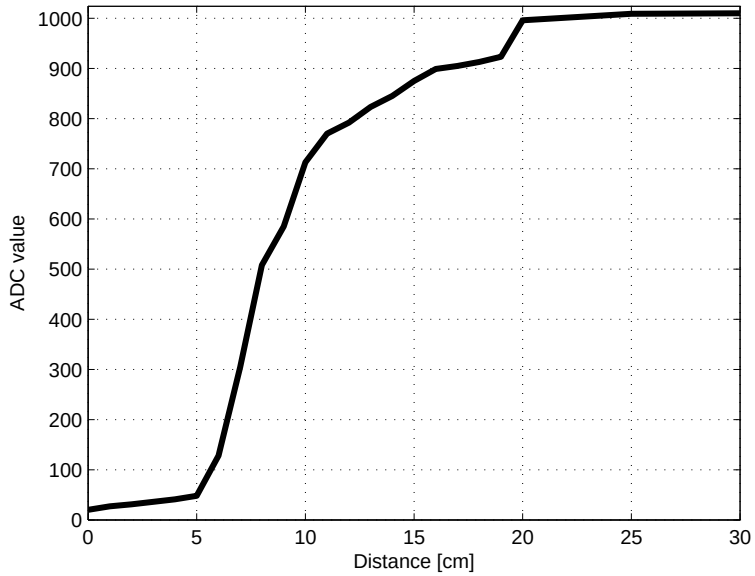


Figure 3.11: The ADC value from the low voltage level of received signal versus the distance between two robots.

However, there still exists a weakness even using communication signal to detect distance. Because when the IR transmitter does not point to the receiver in a straight line. The energy of received signal will be lower. So another experiment was done based on the last experiment. The distance between two robots are kept to be $5cm$, and then rotated the transmitting robot from -60° to 60°. Positive angle is defined to be clockwise rotated. The result is indicated in Figure 3.12. The detectable range of angle is from -40° to 50°. The asymmetry of the curve on the left and right side of 0° is due to the hardware layout of the IR couple. In physical, an IR couple is parallel mounted on one sector side by side but not ideally at one point. It is obviously shown in Figure 3.12 that even a bit deflected angle between the IR transmitter and the IR receiver can induce a notable change of the ADC value. If the distance were longer, the change would be much more evident.

In a short summary, using communication signal for detecting distance to the other robot is the best choice based on the hardware design. And the Equation 3.6 can be used to calculate this distance.

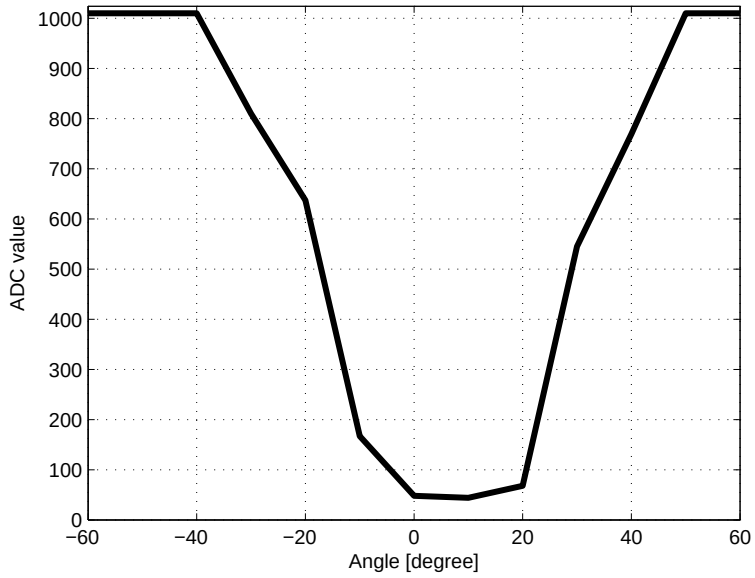


Figure 3.12: The ADC value from the low voltage level of received signal versus the angle deflected from the IR transmitter to the IR receiver when distance is 5*cm*.

3.4 Ambient light detection

Besides the basic functionalities of communication and distance measurement, the robot is also equipped with six ambient light sensors. The placement of these six sensors can be either seen from Figure 2.1 or Figure 2.2. In this section, the main topic is to describe how the ambient light sensor works.

The Figure 3.13 shows the schematics of one ambient light sensor. In the real robot, there are six of them mounted. They are responsible for each sector and can be switched by the multiplexer. This sensor consists of a phototransistor. When it is exposed to a light source, it could generate photocurrent whose value is mainly related to the illuminance, angle and temperature. The detailed information can refer to the datasheet of APDS-9002 [adp04].

The output photocurrent passes to the ground through a resistor of R6. Thereafter, a level of voltage drop $V_{lightsensor}$ is generated at the point of p4 calculated by Equation 3.7

$$V_{lightsensor} = I_{CE} \times R6 \tag{3.7}$$

Where I_{CE} is the photocurrent. Then, this value can be converted to digital value by ADC part of MCU. The function of the capacitor, C1, is to filter noise.

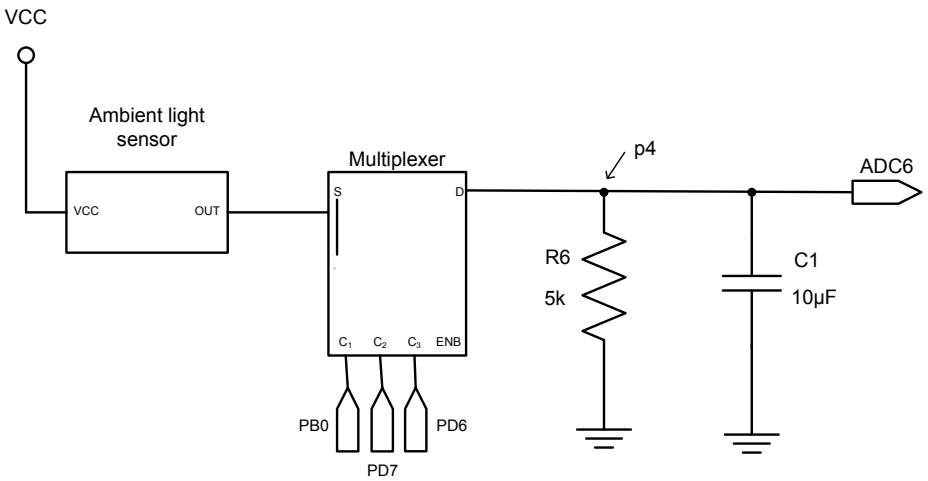


Figure 3.13: The schematics of one ambient light sensor. There should be six of them connected to the multiplexer, but not drawn out repeatedly.

Because all the experiments were done in the laboratory. Compared to the illuminance and angle, the influence from temperature is less important. So an experiment was done to detect the relations between the ADC value and two factors that are illuminance and angle.

The experiment was conducted in a dark environment and a bulb was used as a light source. The front ambient light sensor was tested by rotating the robot clockwise from pointing to the bulb defined as 0° to being back to the bulb that is 180°. Two settings of distance are tested, which are $20cm$ and $40cm$ far from robot to the bulb.

The results are shown in Figure 3.14. The dashdot curve with the legend of dark is the ADC value when the bulb is off. The value keeps 12 for all angles. The solid curve with the legend of $40cm$ is the result from keeping the distance to be $40cm$ and rotating the robot. The tendency of the curve is obviously to show that the ADC value decreases when deflecting the ambient light sensor to the light source. Because the bulb is on, the ADC value still keeps 52 even 180° rotated. The dashed curve is got by setting the distance to be $20cm$. This curve is always higher than the solid curve since the robot is $20cm$ closer to the light source and the illuminance is much stronger on this place than the place on $40cm$ away. Also, it can be seen that the dashed curve decreases slowly until 70° deflected. It dues to the fact that the bulb can not be treated as an ideal point light source if the distance is not far enough from the robot to it. Thus, with the distance of $20cm$, even though the robot is rotated 70°, the ambient light sensor can still directly receive the light emitted from the bulb, but not the light from any reflector.

With six ambient light sensors mounted on six sectors respectively, the robot has more

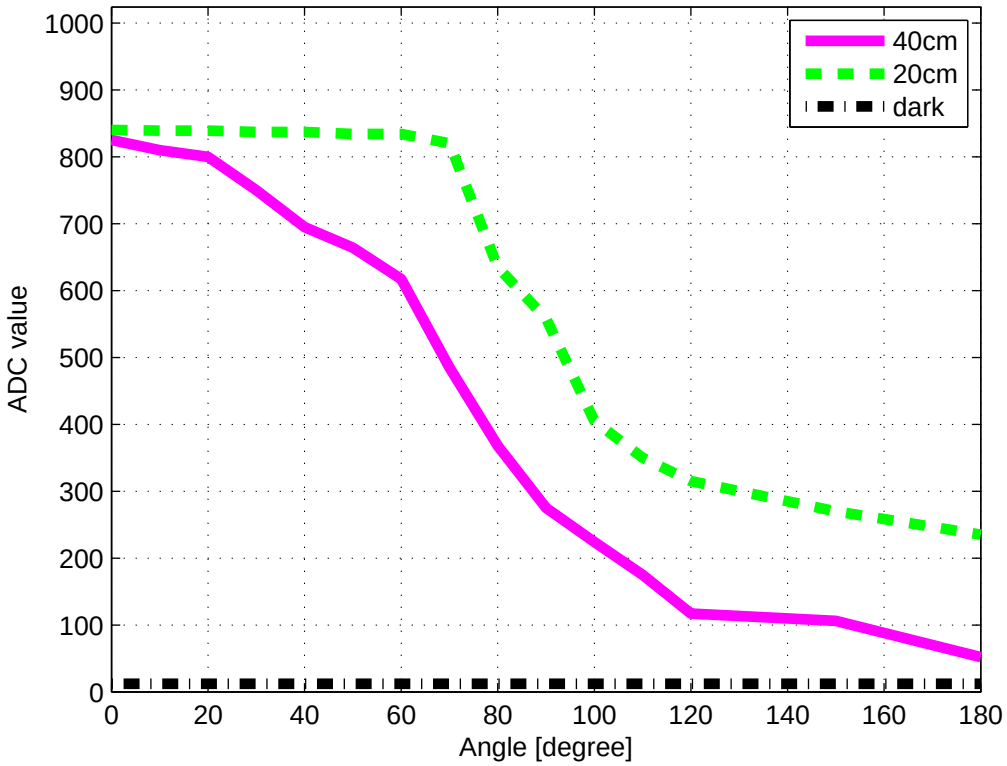


Figure 3.14: The ADC value from the ambient light sensor versus the rotated angle.

chance to feel the light and furthermore to judge the direction of the light source.

3.5 Color detection

The robot has one color sensor placed in the front of itself. The original function of the color sensor, TCS230, is able to detect red, blue and green [tcs04]. But due to the limitation of MCU used for this robot, there are only two pins left that are PD3 and T0. Because this sensor is a converter from color light to frequency, the pin of T0 is important for extracting the information expressed by the frequency. Thereafter, only one pin is left for switching to detect different colors.

The connection of the color sensor is set up accroding to the datasheet of TCS230[tcs04]. And the schematics is illustrated by Figure 3.15. S0 and S1 are the setting for the scaling of output frequency. They are connected to the VCC which implicates that the highest output frequency would larger than 500kHz. The setting of S2 and S3 is used to select what color filter is used. In Figure 3.15, S2 is connected to the ground and S3 is connected to the PD3 of MCU. It leads to the fact that the robot can only detect red or blue corresponding to low or high output from PD3. The output of the color sensor connects to the T0 of the MCU. The clock of Timer/Counter0 is configured to be external clock on the falling edge of the T0 pin.

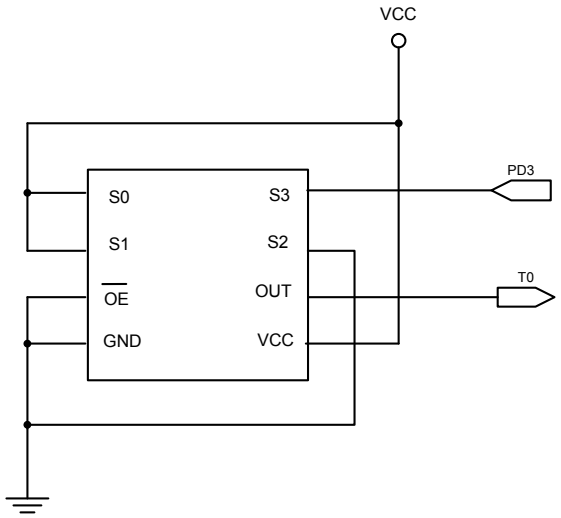


Figure 3.15: Schematics of the color sensor.

But the Timer/Counter0 is 8-bit and the maximal counted number is 255, which is much lower than the highest frequency. There are two ways that can be used to solve this problem.

One way is to counter the falling edge of the output square wave in shorter time. In the program, the interval time for countering is set to be $1ms$. It leads to that only one thousandth of real frequency need to be countered.

The second way is to use the feature of Timer/Counter0 overflow interrupt. Inside the Interrupt Service Routine (ISR), a 8-bit variable is used to increase one every time when this interrupt is triggered. So using this way, a 16-bit counter is created that can count the number up to 65535.

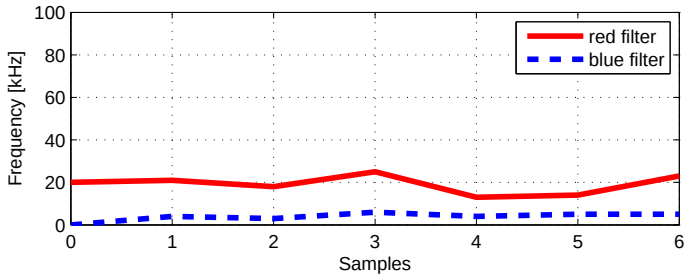
Both ways are used to count the number of falling edges during every $1ms$. The real frequency can be calculated by one thousand times of this number.

On the implementation of color detection, an object decorated with different LED arrays is treated to be the colored object. The object can emit red and blue light that robots need to detect. In a dark environment, several experiments were conducted to test the relation between the output frequencies and the distance to the colored object when using red or blue filter. The color sensor keeps pointing to the object straightly. The results are shown in Figure 3.16(b) and Figure 3.16(c). The Figure 3.16(a) shows the output frequency when no LED array is on, that is, the result from using red or blue filter for the ambient light. It can be seen that in that specific environment without any color in front, the output frequency is around 20kHz if using red filter and the frequency is below 10kHz if using blue filter. With stronger ambient light, both values would increase.

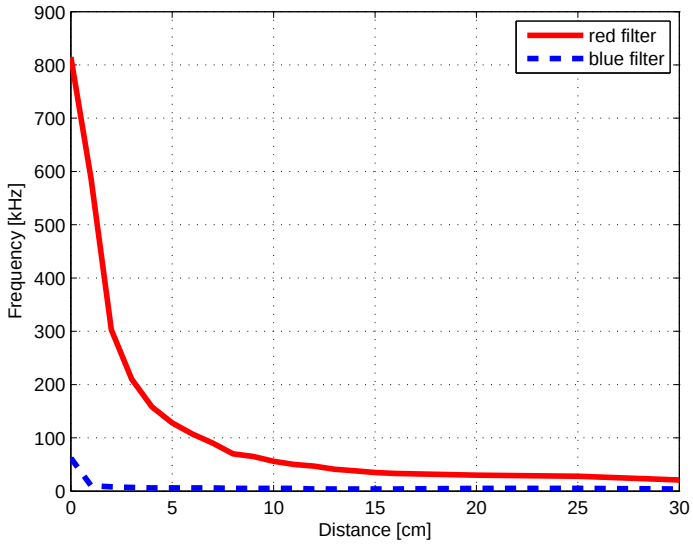
When the object emits red light, the result is indicated in Figure 3.16(b). The output frequency by using blue filter keeps below 10kHz except 62kHz that got from the case when the red light is very close to the sensor. The output frequency when using the red filter is much more sensitive to the distance. The value is around 800kHz when the red light is close to. Beyond the distance more than $15cm$, the value is lower than 25kHz and the decreasing slope of the curve becomes gentle.

The Figure 3.16(c) shows the result when the object emits blue light. In this case, the frequency when using red filter keeps around 20kHz except 38kHz when distance is $0cm$. However, the output frequency when using blue filter is much more sensitive if the distance is less than $5cm$.

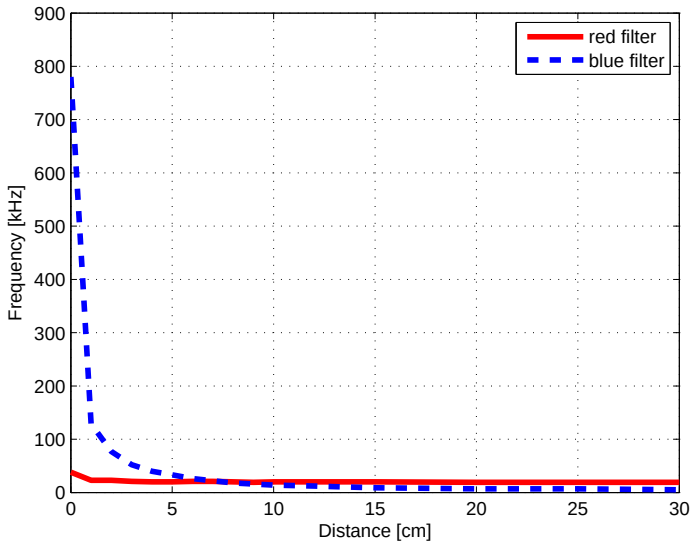
From above analyzing the results from the experiment, it could get a conclusion. In the dark environment, the red light can be detected if the distance less than $15cm$ and the detectable distance for blue light is less than $5cm$.



(a) When LED arrays are off. The X-coordinate is a series of samples.



(b) When the red LED array is on.



(c) When the blue LED array is on.

Figure 3.16: The output frequencies when using red or blue filter in three situations.

3.6 Maneuverability

The robot needs to response immediately when suffering obstacles or getting communication signal from other robots. Thus, the maneuverability for the robot is considered to be an important functionality. In this section, the actuation system is discussed.

3.6.1 Construction of actuation system

This robot is driven by two stripes of plastic pedrails on left and right side, respectively. The two pedrails can be seen from Figure 2.1 and Figure 2.2. Each pedrail is indirectly driven by one micro DC motor that places inside the chassis. The schematics of mechanical construction of actuation system is illustrated in Figure 3.17 briefly. Two motors are fixed side by side. The power for the motivity is derived from the motors and passes though the gear systems. In the end of the gear systems, two wheels are connected to the shafts. The wheels can drive the pedrails to be rolling forwards or backwards.

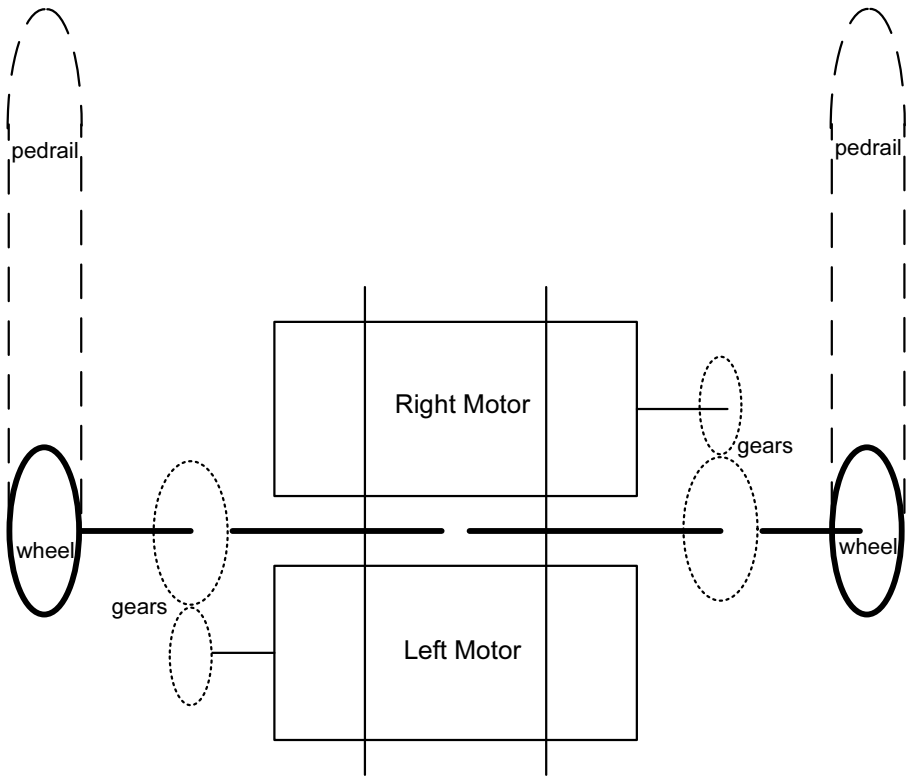
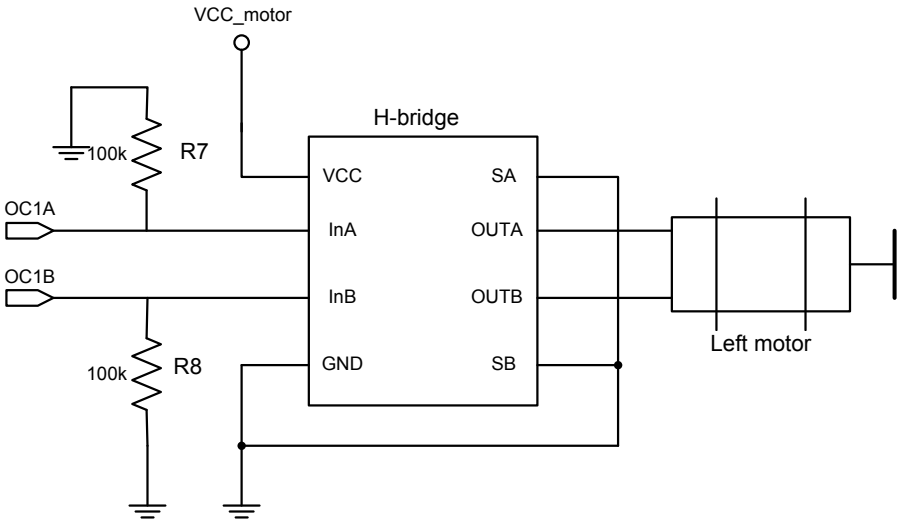


Figure 3.17: The schematics of mechanical part of the actuation system

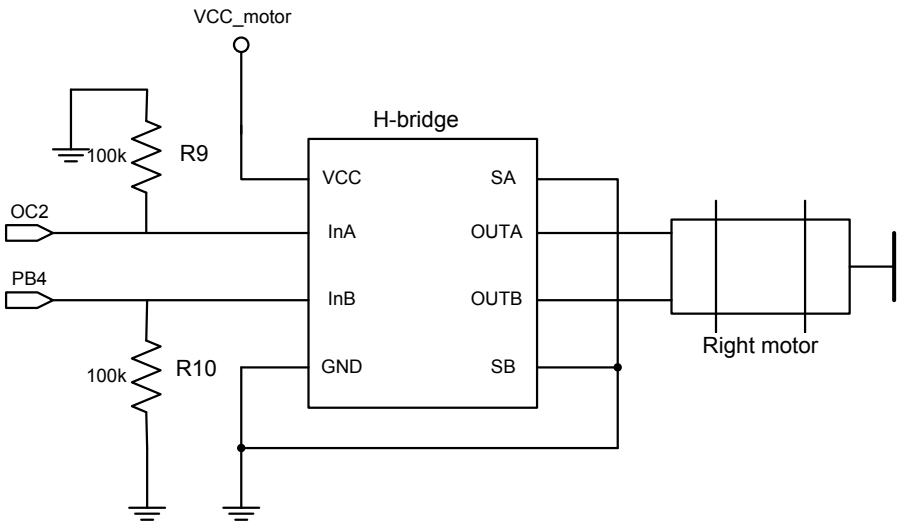
3.6.2 Motor controller

Each micro DC motor is controlled by using a H-bridge chip, that is **Si9986** from **Vishay Siliconix**.

With the H-bridge, the DC motor can be controlled to run forwards or backwards. The schematics of controllers for two sides of motors is depicted in Figure 3.20. The function of this H-bridge chip is shown in Table 3.2. *HiZ* represents the state of high impedance.



(a) Left motor controller.



(b) Right motor controller.

Figure 3.18: The schematics of electronic part of the actuation system.

Taking the Figure 3.18(a) for example, if the MCU outputs high on OC1A pin and low on OC1B pin, the OUTA pin of Si9986 would be high and OUTB would be low. So the

Table 3.2: The truth table of Si9986.

InA	InB	OUTA	OUTB
1	0	1	0
0	1	0	1
0	0	0	0
1	1	HiZ	HiZ

motor can run in the forward way. On the contrary, the motor would run in the backward way. If both outputs of OC1A and OC1B are low, both inputs to the H-bridge are pulled down to the ground and the motor would be stopped. All the situations are the same for the right motor shown in Figure 3.18(b).

The Pulse Width Modulation (PWM) waves generated from MCU are used to control the speed and direction of motors. OC1A and OC1B are for the left motor and OC2 and PB4 are for the right motor.

3.6.3 Generation of PWM signal

PWM signal involving the modulation of its duty cycle can control the amount of power sent to a motor [pwm]. It use a square wave whose duty cycle is modulated resulting in the variation of the average value of the waveform. Considering the square wave form, $V(t)$, input to the motor with a low level voltage $V_{min} = 0V$, a high level voltage $V_{max} = VCC_{motor}$ and a duty cycle D of cycle time T, the average value, $\bar{V}$, of the waveform is given by Equation 3.8.

$$\begin{aligned} \bar{V} &= \frac{1}{T}\int_0^T V(t)dt && (3.8)\\ &= \frac{1}{T}\left(\int_0^{DT} V_{max}dt + \int_{DT}^{T} V_{min}dt\right) && (3.9)\\ &= D\cdot V_{max} + (1-D)V_{min} && (3.10)\\ &= D\cdot VCC_{motor} && (3.11) \end{aligned}$$

According to two motor controllers shown in Figure 3.20, it needs to be four PWM waves to control two motors. But referring to the datasheet of ATmega8L [atm06], there are only three PWMs can be provided. Two PWMs are derived from the Timer/Counter1 and one is from the Timer/Counter2. So one PWM with the square wave form should be generated by programming to frequently change the digital output on one I/O pin. In this way, there is an drawback compared to using the Timer/Counter for generating PWM

wave. The Timer/Counter is a particular hardware part integrated in the MCU and it can run independently as long as it has been configured one time in the program. So during outputing the PWM waves, the program can run other functions. However, for simulating normal I/O pin to output PWM, the program has to stay in this function until output is stopped.

Selection of PWM frequency

The PWM frequency is an important factor that has to be considered. The frequency is the reciprocal value of the cycle time.

If the frequency were higher, the cycle time of PWM wave would be shorter. In this case, the motor will switch between on and off very quickly. Consider the characteristics of the motor that it needs to take a short time whenever switch a motor on due to the static friction inside the motor and also from the load. Therefore, the PWM frequency can not be too high.

On the opposite case, if the frequency were lower, The cycle time of PWM wave would be longer. It will make the motor not run continuously and it is easy to see the motor on and motor off.

An experiment was conducted to see the performance of the motors when set by different PWM frequencies with duty cycle 50%. Finally, the best suitable PWM frequency for the motors is $112.5hz$.

For Timer/Counter0 and Timer/Counter1, this value can be got by setting the clock source to be $\frac{1}{256}$ of the system clock and waveform generation modes to be 8-bit Fast PWM mode and non-inverting Compare Output mode. In Fast PWM mode, the counter counts from BOTTOM of 0x00, to TOP of 0xFF, then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compares, including OC1A, OC1B and OC2, are cleared on the compare match between the registers of *TCNT* and *OCR*. The resolution is 8-bit and the duty cycle can be computed by $\frac{OCR+1}{256} \times 100\%$.

For the PD4, the wave form should be set manually in the program. The cycle time T is $8.889ms$. If define D as duty cycle, the wave form can be generated by repeating the procedure of outputing high for period $D \cdot T$ followed by low for period $(1-D)T$.

3.6.4 Testing of motors

Two experiments were conducted to test the motors driven by PWM waves output from the H-bridge. The first experiment is to let the robot to move forwards. Two PWM waves with duty cycle 40% are generated in the pins of OC1A and OC2 for left and right motor, respectively. While the pins of OC1B and PB4 keep zero. It can be seen from Figure 3.19(a) that the duty cycles are precise on both PWMs. But in real movement of the robot, the robot is always tend to turn left. So it can be deduced that the two motors is not absolutely identical. The experiment was repeated by keeping the OC1A and lowering down the duty cycle for OC2. The Figure 3.19(b) shows that when the duty cycle of OC2 is 2.2% less then 40% of OC1A, the robot can move forwards straightly.

The second experiment is similar as the first one but to let the robot move backwards, which is used to test the other two channels of PWMs output. The duty cycle is set to be 50%. From the Figure 3.20(a), it can be seen that the duty cycle of the wave from PB4 can reach to the same accuracy as the output from Timer/Counter. But due to the non-identical motors, if the robot is expected to move backwards straightly, the duty cycle of OC2 has to be 42% when OC1B is 50%, which is shown in Figure 3.20(b).

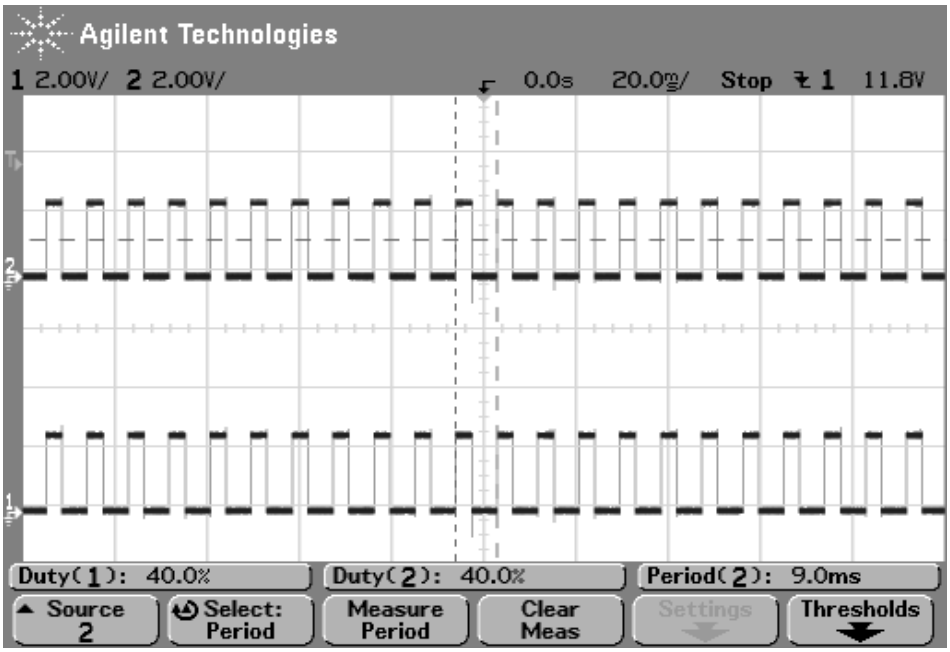
3.6.5 Summary

From the results of experiments, it can be seen that the MCU can output PWM waves with precise duty cycle from either Timer/Counter or normal I/O pin. But a problem is exposed that two motors for the robot is not identical. Also, during the test, it is easy to see the phenomenon of the slippage between the wheels and pedrails, especially in the cases of turning left or right. This is because only cheap plastic pedrails are used and they are easy to become tensionless to the wheel. It leads to the pedrails having less friction between the wheel than the floor. Therefore, every motor in robots need to be treated separately.

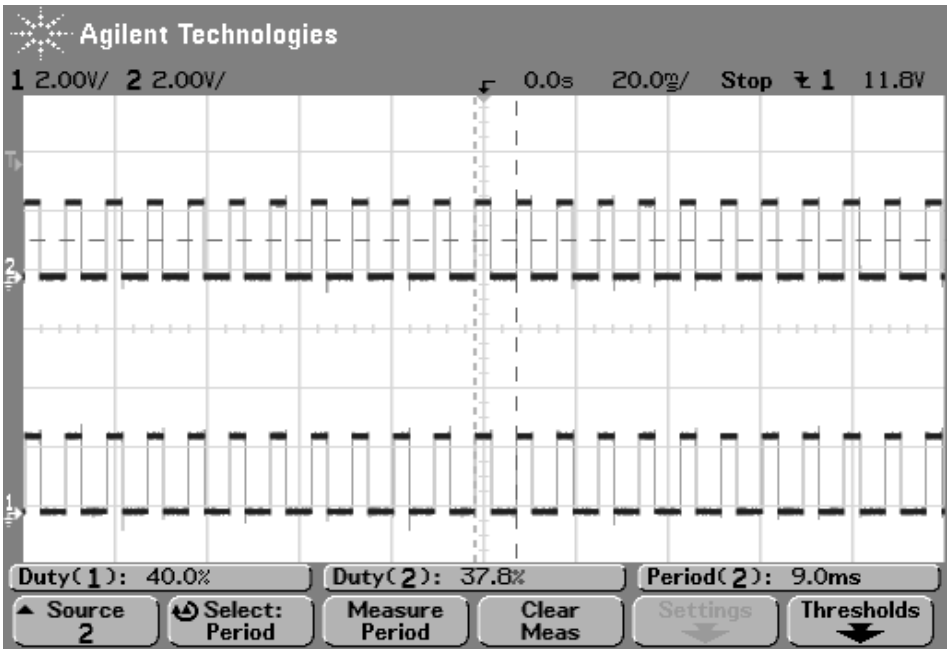
3.7 Summary and suggestion

In this chapter,five main functionalities provided by hardware have been discussed except the Global IR receiver, which has not been implemented.

In Section 3.1, the robots communication is described. The frame format in one data package is defined as 1 start bit, 8 data bits, 2 stop bits. The baud rate is set to be $9600bps$ and the maximum communication distance is $20cm$. The main influence on the com-

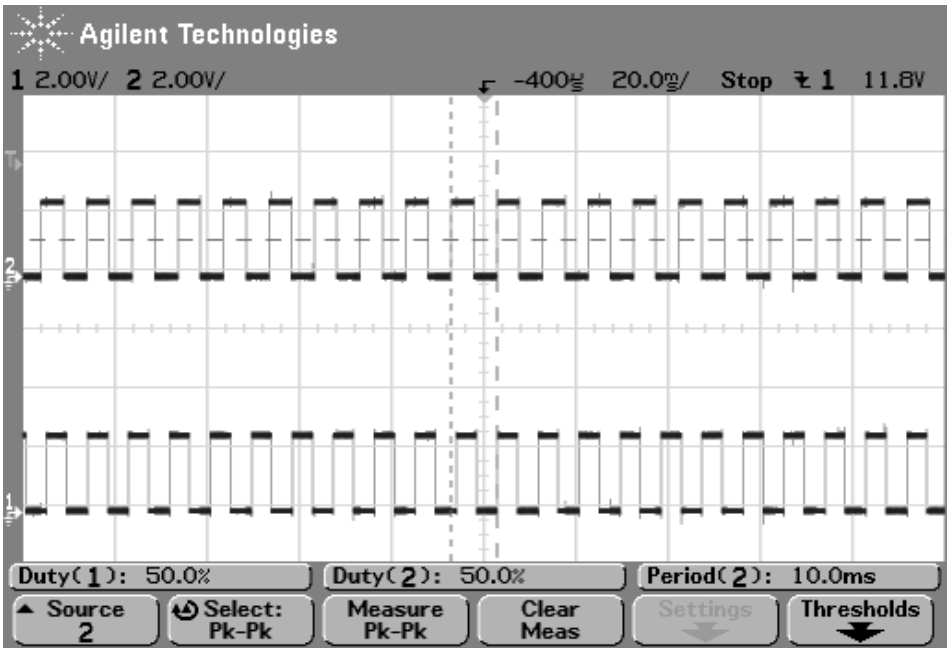


(a) When two pins have same duty cycles, the robot tends to turn left.

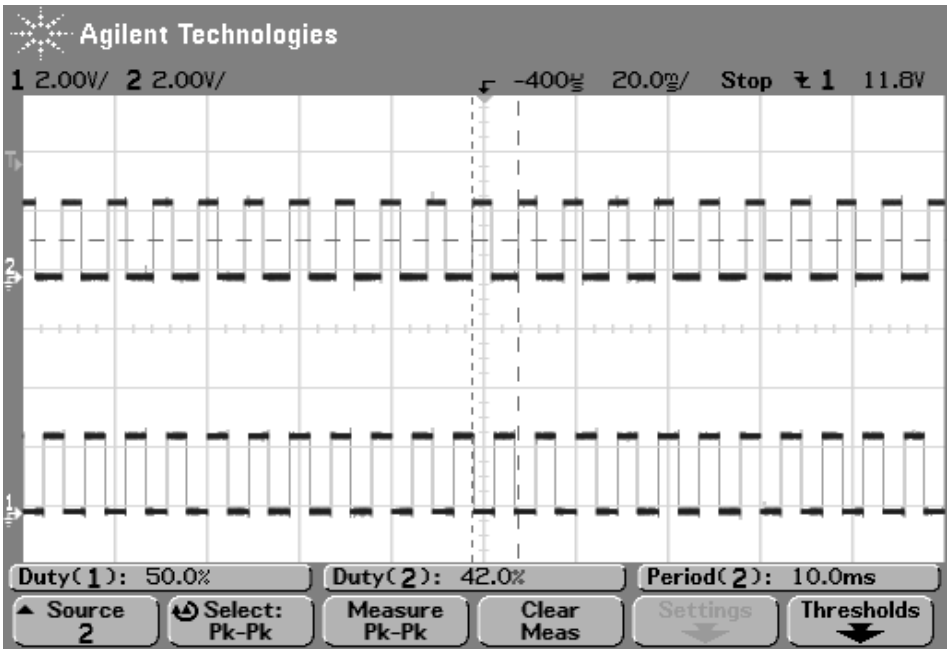


(b) After calibration, the robot can move forwards straightly.

Figure 3.19: Testing results when let the robot move forwards. The upper PWM waves with ground 2 is detected at the pin of OC1A. The lower PWM waves with ground 1 is detected at the pin of OC2.



(a) When two pins have same duty cycles, the tail of the robot tends to the left side.



(b) After calibration, the robot can move backwards straightly.

Figure 3.20: Testing results when let the robot move backwards. The upper PWM waves with ground 2 is detected at the pin of OC1B. The lower PWM waves with ground 1 is detected at the pin of PB4.

munication speed and distance is from the IR receiver. This receiver is actually an IR phototransistor and the rising time is relatively longer than an IR photodiode. Therefore, it suggests to use an IR photodiode rather than the IR phototransistor. Also, the resistance R3 has to be adjusted to ensure the best abilities of communication speed and distance.

On distance measurement, two different approaches are discussed. By the means of detecting the voltage level of the reflected IR signal, the robot can estimate the distance to the wall. This approach is discussed in Section 3.2 and the Equation 3.5 can be used to estimate the distance. The measurable distance is less than $10cm$ for the foam and the value would be much changeable for different reflectors. This ability can be enhanced by increasing the resistance of R3. But this change should be considered synchronously with the communication ability.

In the other case of estimating the distance to another robot described in Section 3.3, a possible approach can be used is to detect the low voltage level of received communication signal. The Equation 3.6 is suited for this case. However, the estimated distance would be false if the IR receiver does not point to the transmitter straightly. This problem could be improved by mounting more IR couples averagely on the circle of the PCB. These new IR couples have to be selected with less aperture angle.

In Section 3.4, the ability of ambient light detection is realized by using six ambient light sensors. The analogue output of this sensor is mainly depending on the illuminance and the deflected angle to the light source.

In Section 3.5, it states that the robot can detect red light inside $15cm$ and blue light inside $5cm$. If the MCU could provide one more I/O pin, the green color would be also detectable.

The Maneuverability is critical for the robot and discussed in Section 3.6. The robot is driven by two pedrails whose power is originally derived from two micro DC motors. By using two H-bridge chips, The MCU can output four PWM waves to drive left and right motors. The PWM waves can be generated with precisely expected duty cycle from either Timer/Counter or normal I/O pin.

However, there exists some problem for this cheap actuation system. The two motors for each side are not identical and they need to be adjusted manually to ensure the robot can move straightly. The other problem is the slippage between the wheels and pedrails. Therefore, a robust motor system is recommended to replace the current one. If speed detection sensors are mounted, each motor can be accurately adjusted to be a reference speed. Also, a metal-made pedrail and fastened joints between wheels and pedrails are

suggested to replace the current one.

Chapter 4

Single Robot Behaviors

The robot is provided with several hardware functionalities, which have been discussed in Chapter 3. In this chapter, the implementations of those functionalities for a single robot are described, including communication, distance measurement, obstacle avoidance, light source approaching and color reaction. The algorithms according to each behavior are developed and validated.

4.1 Communication

The communication between robots needs to be reliable. The swarm robots' behaviors rely on communication significantly. With knowing the statuses of the other robots, one robot can make a decision for itself. The robot can observe another robot by using its own sensors, but it is less believable than the information received directly from the speaking of the other robot. In order to realize the communication, a protocol has to be agreed with firstly by all the robots. Based on the protocol, the algorithms for sending and receiving signals are developed.

4.1.1 Protocol

A communications protocol is a set of standard rules for data representation, signalling, authentication and error detection required to send information over a communication channel. It has the features intended to ensure reliable interchange of data [cmp].

For the communication between robots, the protocol is built mainly on the physical layer,

that is RS-232. RS-232 is a standard for serial binary data signals connecting between a Data Terminal Equipment (DTE) and a Data Circuit-terminating Equipment (DCE). But for functional communication through a serial port interface, conventions of bit rate, character framing, character encoding, data compression and error detection, not defined in RS-232, must be agreed to by both sending and receiving equipment. The implementation for these robots is to use an integrated circuit part, that is UART [rs2]. The UART can convert data from parallel to serial form. It is using asynchronous start-stop character formatting with 8 data bits per frame with ASCII character coding. The data rate defined by the baud rate has set to be $9600bps$. The more information on UART has been discussed in Subsection 3.1.1.

For every robot, there are six communication channels created by local IR couples. But due to the hardware design, they are all unidirectional and they can not send or receive signal with six channels simultaneously. It needs to switch the sending or receiving channel by setting in the program. Also, the half-duplex is used, which means either sending or receiving can operate during one period of time. Therefore, Two robots only have the chance to do communication when the sending channel of one robot is pointing to the receiving channel of the other robot at that moment. A bit of deflected angle is allowed. The allowed value of this angle is depending on the distance between two robots. The farther distance, the less allowed deflected angle. Consider one simple situation that, one robot only switches six channels to transmit signal and the other robot only switches six channels to receive signal. The probability of the robot to receive the signal is only $\frac{1}{36}$. If more functions are added besides signal sending and receiving, there would be much less chance to get the communication signal. So one critical requirement is that, the communication rule has to be defined as simple as possible. On the implementation, assistant programs can be added on to increase the chance for getting the communication signals. If a robot gets no communication signal at one moment but there was signal before, the robot can try to get signal again by waiting for a while or adjusting its direction. If there is still no signal received, the robot would believe that there is no neighbor robot.

In the protocol, no acknowledgment message would be sent back when a robot receives signals and the communication speed is defined to be $9600bps$. The ability of confirmation of a signal completely relies on the UART.

All information needs to tell to another robot is packaged in one character, that is 8-bit data. The schematics of 8-bit data is shown in Figure 4.1. The bit 3 and bit 4 contain the information of the ID of a robot who sends the signal. Because only two robots are implemented, two bits are enough to identify different robots. The bit 0 to bit 2 are used to

carry the information of the channel ID of an IR couple which is used to send the signal. Three bits are enough to enumerate six channels. The channel ID contains an important information that is direction. The other three bits 5, 6 and 7 are free. They can be used for specific behaviors with different aims if necessary.

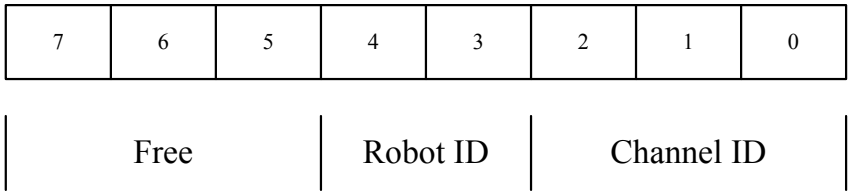


Figure 4.1: The schematics of 8-bit data which is used to contain the useful information.

4.1.2 Signal sending

The algorithm for signal sending is written in Algorithm 1.

The algorithms are represented by statements followed by parameters inside parenthesis, ( ). Variables are written in italics. Arrays are represented by their names with square brackets, [ ]. An algorithm can also be used inside other algorithms. The texts inside the big brackets, { }, are the comments. $A \Leftarrow B$ represents the value A is given by the value B. The other symbols are similar to the C Programming Language. These rules are used in the whole report.

Algorithm 1 Signal sending(*Status*)

1: $PD5 \Leftarrow 0$ {Set the communication mode.}
2:
3: **for** $Channel\_Number = 0$ to 5 **do**
4: Select channel (*Channel_Number*)
5: $TXD\_Char \Leftarrow (Status << 5 \mid Robot\_ID << 3 \mid Channel\_Number)$
6: **for** $Transmit\_Time = 0$ to 3 **do**
7: Output char (TXD_Char)
8: **end for**
9: **end for**

The sending signal is represented by variable, TXD_Char, which is related with the robot's status, ID, and sending channel. The signal will be transmitted four times in each channel, which benefits for confirmation in the receiving end.

4.1.3 Signal receiving

The Algorithm 2 shows how the robot receives signal. The receiving channel is switched sequentially from channel 0 to channel 5.

When one channel is selected to receive signal, it is planed to be waiting for maximal 15000 loops in the algorithm. If no signal were got during this period of time, it would turn to the next channel. Only if the same data are received two times, the robot ensures the data is available and then it switches to the next channel. As well, during the interval between two times to get the same data, the AD converter of MCU is planned to run 80 times to detect the voltage level of received signal. The temporary ADC value is calculated by averaging every two times of ADC results. Finally, only the minimum value is kept, which results from the low level voltage of the received signal. This minimum value has the relationship with the distance to the other robot who sends that signal. Two arrays both with six elements are used to save the received signals and distance information from each channel, respectively. Finally, the received signal has to be handled and the process is written in Algorithm 3.

The Algorithm 3 describes the process of extracting the useful information from the received signals of six channels. For different robots' objectives, the useful information would be different. This algorithm is established by an assumption that one robot is only interested in the other robot with lowest ID. Firstly, the algorithm is to filter the noise or wrong signals by judging whether the numbers of the neighboring robot's ID and sending channel are both valid. If they were valid, the useful information would be memorized, including its own received channel number, distance, the neighboring robot's ID, sending channel and status.

4.2 Distance measurement

From the analysis of Section 3.2, it have been seen that it is only possible to estimate the distance to the wall by detecting the voltage level from reflected IR. For the distance between robots, the approach of detecting the low voltage level of received communication signal is preferred. In last section, the Algorithm 2 developed for signal receiving has already considered the issue of detecting the distance to another robot. In this section, the main attention is paid on detecting the distance to the wall.

The main purpose to do distance measurement is to avoid obstacles, which will be discussed in the next section. Only in the case of emergency, the robot is planned to move

Algorithm 2 Signal receiving()

```
1: PD5 ⇐ 0 {Set the communication mode.}
2:
3: for Channel_Number = 0 to 5 do
4:    Select channel (Channel_Number)
5:    Received_Signal[Channel_Number] ⇐ 0
6:    Distance_to_Robot[Channel_Number] ⇐ 0
7:    RXD_Char ⇐ 0
8:    Last_RXD_Char ⇐ 0
9:    Distance_to_Robot_temp ⇐ 1000 {Initialized by a large number.}
10:   for Waiting_Times = 0 to 15000 do
11:      if USART_Char_Received then
12:         RXD_Char ⇐ USART_Char_Received
13:         if RXD_Char == Last_RXD_Char then
14:            Received_Signal[Channel_Number] ⇐ RXD_Char
15:            break
16:         end if
17:         Last_RXD_Char ⇐ RXD_Char
18:         for ADC_Times = 0 to 40 do
19:            ADC_temp ⇐ ADConvert(Channel_Number) {Take the average value of
               two times of AD conversion.}
20:            if ADC_temp < Distance_to_Robot_temp then
21:               Distance_to_Robot_temp ⇐ ADC_temp {Keep the minimum of ADC
                  value}
22:            end if
23:         end for
24:      end if
25:   end for
26:   Distance_to_Robot[Channel_Number] ⇐ Distance_to_Robot_temp
27: end for
28: Signal handling()
```

Algorithm 3 Signal handling()

```
1: Neighbor_Robot_ID ⇐ 100 {Initialized by a large number.}
2: for Channel_Number = 0 to 5 do
3:   if Received_Signal[Channel_Number] ≠ 0 then
4:     Neighbor_Robot_ID_temp ⇐ (Received_Signal[Channel_Number] & 0x18) >>
       3
5:     Neighbor_Robot_Channel_temp ⇐ Received_Signal[Channel_Number] & 0x07
6:     if Neighbor_Robot_ID_temp is valid && Neighbor_Robot_Channel_temp < 6
       then
7:       if Neighbor_Robot_ID_temp < Neighbor_Robot_ID then
8:         Neighbor_Robot_ID ⇐ Neighbor_Robot_ID_temp
9:         Neighbor_Robot_Channel ⇐ Neighbor_Robot_Channel_temp
10:        Distance_to_Neighbor_Robot ⇐ Distance_to_Robot[Channel_Number]
11:        Status_of_Neighbor_Robot ⇐ Received_Signal[Channel_Number] >> 5
12:        My_Received_Channel ⇐ Channel_Number
13:        There_is_a_Neighbor_Robot ⇐ 1
14:      end if
15:    end if
16:  end if
17: end for
```

backwards on purpose. In most cases, the robot is excepted to move in forward directions. Thus, it is only necessary to use three front IR couples, which are mounted in channel 0, 1 and 5.

This process has been written in Algorithm 4. Using this algorithm, the robot will take turns on channel 1, 5 and 0 to detect the distances to the wall from three directions. These three distances are compared. The minimum distance as well as the error between maximal and minimal distance is memorized.

4.3 Obstacle avoidance

One of basic behaviors for a robot is to avoid obstacles. It is a relatively important functionality need to be achieved. Before a robot can do something else, it has to ensure itself not to be in a potential crash. On the implementation of this robot, two kinds of obstacles are needed to be taken into account, including the wall and the other robots.

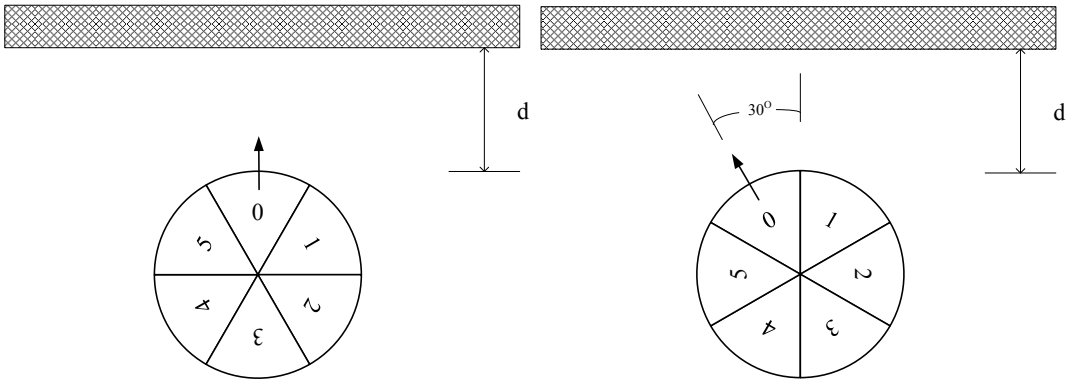
Algorithm 4 Distance measuring()

```
PD5 ⇐ 1 {Set the distance measurement mode.}

Select channel (1)
Distance_to_Wall[1] ⇐ ADConvert(Channel_Number)
Min_Distance ⇐ Distance_to_Wall[1]
Max_Distance ⇐ Distance_to_Wall[1]
Channel_of_Min_distance ⇐ 1

Select channel (5)
Distance_to_Wall[5] ⇐ ADConvert(Channel_Number)
if Distance_to_Wall[5] < Min_Distance then
    Min_Distance ⇐ Distance_to_Wall[5]
    Channel_of_Min_distance ⇐ 5
else
    Max_Distance ⇐ Distance_to_Wall[5]
end if

Select channel (0)
Distance_to_Wall[0] ⇐ ADConvert(Channel_Number)
if Distance_to_Wall[0] < Min_Distance then
    Min_Distance ⇐ Distance_to_Wall[0]
    Channel_of_Min_distance ⇐ 0
else if Distance_to_Wall[0] > Max_Distance then
    Max_Distance ⇐ Distance_to_Wall[0]
end if

Error_of_Max_Min_Distance ⇐ Max_Distance − Min_Distance
```

4.3.1 Wall avoidance

From the Figure 3.9, it can be seen that the detectable distance to the wall is less than 10*cm*. Taking the channel 0 for example, this value is got by setting the wall perpendicular to the radial of IR, which is shown in Figure 4.2(a). In this situation with the assumption of glazed surface of the wall, the angle of incidence as well as the angle of reflection is 0° . Thus, the energy detected from reflected IR has to be largest in this situation. If keeping the same distance to wall and rotating the robot 30° clockwise, the distance that IR signal needs to pass would be longer and both angles of incidence and reflection become 30°. This situation is shown in Figure 4.2(b). In reality, the wall is ridged, so the IR receiver can still receive signal. But the energy would be much less if the IR couple points to the wall with an angle. In the situation of Figure4.2(b), apparently from the sensor's reading, the robot will judge the distance from itself to the wall to be much longer than d. The result is that the robot will continue to move forwards even though the actual distance d has been in the dangerous range.

Also, the sensors' readings are all absolute values and influenced by the ambient light. The more IR inside the ambient light, the lower ADC values from all channels.



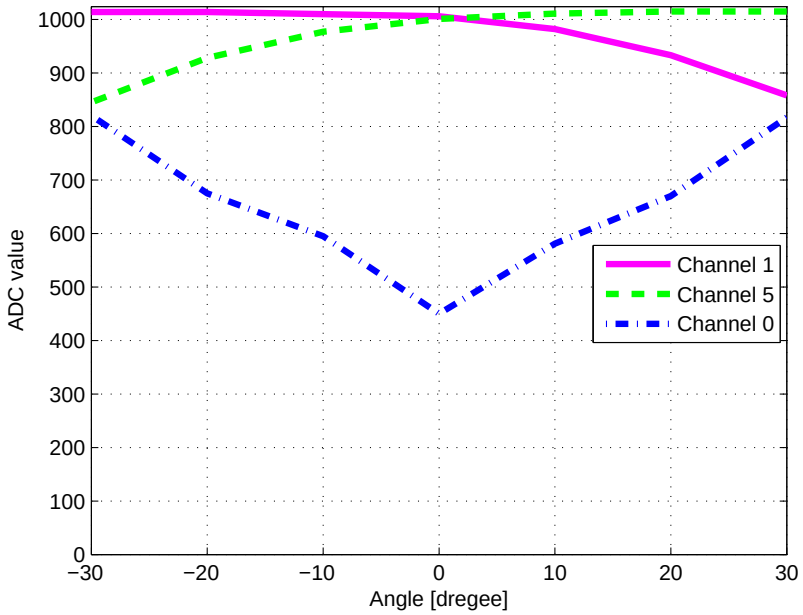
(a) The IR couple of channel 0 is perpendicularly pointing to the wall.

(b) The robot turns left 30° and keeps the same distance.

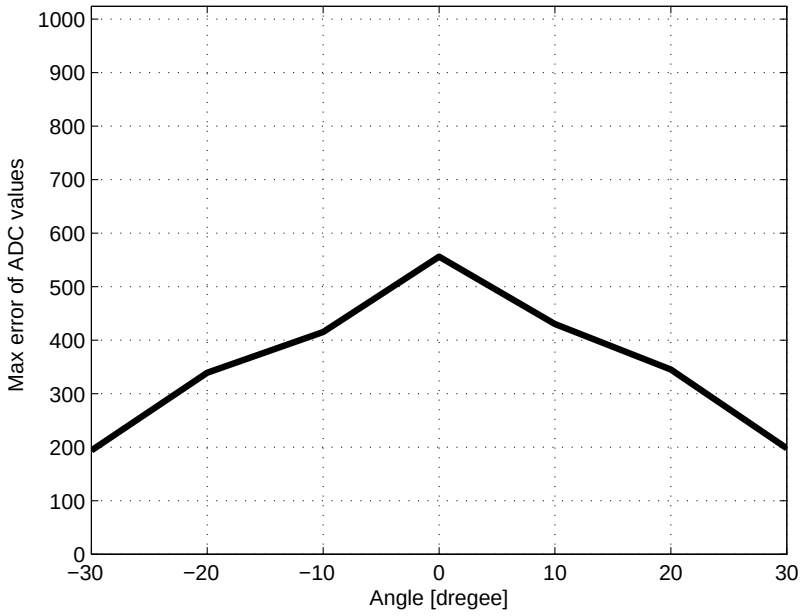
Figure 4.2: The relative position and direction between the robot and the wall. d represents the distance between them.

From the analysis above, it can be seen that it is risky to avoid the wall by directly considering the absolute ADC values. However, by analyzing the ADC values from three channels simultaneously, the problem can be solved. In the following, the maximal error between three ADC values is used to realize the wall avoidance.

The Figure 4.3(a) is the result from an experiment. Referring to Figure 4.2(a), the exper-



(a) The ADC values from the front three channels.



(b) The max error between the three channels' ADC values.

Figure 4.3: The result of an experiment, which was conducted by keeping the distance $2cm$ to the wall and rotating the robot from -30° to 30°. 0° represents the channel 0 perpendicularly points to the wall.

iment was conducted by keeping the distance to be 2*cm* and rotating the robot from -30° to 30°. The positive angles represent clockwise rotating and the negative angles represent anticlockwise rotation. The three curves show the ADC values from three channels, respectively. During this range, the value from channel 0 is never larger than the values from the other two channels, which is lowest at 0° and increases with the rotated angle. The value from channel 1 decreases slightly from -30° to 30°. The value from channel 5 is opposite to the tendency in channel 1.

Figure 4.3(b) shows the tendency of the maximal error of ADC values from three channels. In the case when channel 1 or channel 5 is nearest to the wall with distance of 2*cm*, the maximal error would also be larger than 200. In the case of no obstacle around the robot, this error value is less than 10 induced from non-uniform environment and non-identical IR couples.

Because this max error is an relative value calculated from three channels, it would not be much influenced by the ambient light. Therefore, if 2*cm* is defined as the alarm distance, as long as the maximal error is larger than 200, the robot has to make a decision to avoid the wall.

4.3.2 Robot avoidance

Not only the wall, but also the other robots have to be avoided by a robot. Even though robots would have a plan to cooperate for some specific tasks, they still need to stay outside the safe line from each other. Otherwise, a crash is easy to happen.

Consider a robot called robot1. When this robot meets the other robot called robot2, there might be three situations of the robot2 at that moment. These situations are decided by the IR couple that points to the robot1, including sending nothing, sending communication signal, sending distance measurement signal, which are discussed in the following.

Sending nothing

In this case, the robot2 keeps quiet on that IR couple and sends nothing out. It results that the robot1 can not get any signal from the robot2. The only chance of the robot1 to detect the robot2 is to use the functionality of distance measurement by reflected IR. However, the robot1 can not recognize the robot2 and only knows there is something in front of itself.

This topic has been talked in Section 3.2.2 and the conclusion shows it is difficult to detect

the distance to the other robot by using reflected IR. But for the aim to avoid the other robot, the requirement is not so critical as to realize distance measurement. Seen from the Figure 3.10(b), when the distance between two robots is less than 1*cm*, the ADC value from the distance detecting channel is a bit lower than the value in no obstacle situation.

In this case , the other two channels except the channel that points to the other robot, can not detect the other robot and will keep the ADC values above 1000. Therefore, as similar to the case of wall avoidance, the robot can use the relative value of max error to judge whether a robot is in front of itself. The robot needs to avoid obstacle if the max error is larger than 25, which is corresponding to the distance near 0*cm*.

Sending communication signal

If the robot2 sends communication signal, the robot1 can estimate the distance by detecting the low voltage level of received signal. This functionality has been talked in Section 3.3 and the algorithm has been included in Algorithm 2 of *signal receiving()*.

But an unexpected situation would happen. It can be read from the Figure 3.12 that the ADC value is 637 when keeping the distance 5*cm* but deflecting angle -20°. Referring to the Figure 3.11, the value of 637 is corresponding to the distance around 9.5*cm*. The robot would think that 9.5*cm* is still far from being crash and continue to move forwards. But actually, the distance is only 5*cm*.

On the implementation, the dangerous distance is defined as 2*cm* corresponding to the ADC value 31. If the ADC value is less than 31, the robot needs to do avoidance behavior.

Sending distance measurement signal

If the robot2 sends continuous IR signal for measuring distance, the IR receiver of the robot1 will get quite strong IR by using reflected IR to detect distance. The robot1 can not receive any signal in signal receiving mode. In this case, the ADC value from that IR receiver can go down below 100. The robot will treat this small value with the implication of an object being significantly close to itself.

Since if one IR receiver of the robot1 could get the distance measurement signal from one IR transmitter of the robot2, it is also possible to get the communication signal from the robot2. On the implementation, only three channels are set to do distance measurement and the process of doing communication is much more complicated than doing distance measurement. Thus, doing distance measurement spends much less time than doing com-

munication. When running programs for the robots, the chance to get the communication signal is much higher than the chance to get distance measurement signal. Therefore, even though the robot1 receives a strong IR signal, referring to the previous communication results, it can judge whether this signal is from the robot2 or its own reflected IR. This situation will not be considered separately in the algorithm for avoiding obstacles.

4.3.3 Algorithm

From the analysis above, it can be classified into two main situations that a robot needs to perform obstacle avoidance. One is the robot meets the wall or another quiet robot. The other one is the robot meets another robot, from whom it gets the communication signal. The process has been written in Algorithm 5.

Algorithm 5 Obstacle avoiding()

Require: Distance measuring()
Require: Signal receiving()
1: $Obstacle \Leftarrow 1$
2: **if** $There\_is\_a\_Neighbor\_Robot == 0 \&\& Error\_of\_Max\_Min\_Distance > 25$ **then**
3: Wall and quiet robots avoiding()
4: **else if** $There\_is\_a\_Neighbor\_Robot == 1 \&\& Distance\_to\_Neighbor\_Robot < 31$ **then**
5: Robots Avoiding()
6: **else**
7: $Obstacle \Leftarrow 0$
8: **end if**

The robot judges whether there is an obstacle based on the information got from doing distance measurement and signal receiving. So this algorithm requires these two processes before it.

In the first situation, the thresholds represented by max error for wall avoidance and quiet robots avoidance are different. For wall avoidance, the threshold is 200. While for quiet robots avoidance, the threshold is 25. The requirement of obstacle avoidance is to avoid everything provided with potential crash or danger for the robot. So the threshold is set to be 25 in order to avoid both obstacles. With this threshold, the wall can be avoided with the alarm distance of $5cm$.

The Algorithm 6 describes the robot's behavior to avoid the wall and quiet robots. The behavior is decided by judging which one of the front three sectors is nearest to the obstacle. If the sector 0 is the closest one, which is represented by the lowest value from the

channel 0, the robot will further compare the values from the channel 1 and the channel 5. If the value from the channel 5 is lower, the robot decides to turn right 120°. Otherwise, the robot will turn left 120°. In the cases that the sector 5 or sector 1 is closest to an obstacle, the robot would decide to turn right or left with angle 60°, respectively.

Algorithm 6 Wall and quiet robots avoiding()

```
if Channel_of_Min_distance == 0 then
    if Distance_to_Wall[1] > Distance_to_Wall[5] then
        Turn right(120°)
    else
        Turn left(120°)
    end if
else if Channel_of_Min_distance == 1 then
    Turn left(60°)
else if Channel_of_Min_distance == 5 then
    Turn right(60°)
end if
```

In the second situation, if a robot gets communication signal from a neighboring robot and judges that the ADC value is less than 31, it would perform to avoid the neighboring robot.

The robot will judge the position of the neighboring robot relative to itself firstly. If the neighboring robot is in front of itself, the robot would move backwards. Otherwise the neighboring robot is in the back, the robot would move forwards. This behavior has been written in Algorithm 7.

Algorithm 7 Robots avoiding()

```
if My_Received_Channel == (0||1||5) then
    Move backwards()
else
    Move forwards()
end if
```

4.3.4 Testing results

The Algorithm 5 is implemented in both simulation and real experiments.

In the real experiments, there exist several weak points. The ability to detect distance is significantly different from object to object. Also, the robot is easy to lose the communication signal even a neighboring robot is close. Moveover, the actuation system can not be controlled accurately.

The main objective of doing simulation is to test the logic of algorithms. Thus, the problems mentioned above are not supposed to occur in simulation. The ability of distance measurement is ideally provided to each robot. One robot knows the accurate distance either to the wall or to another neighboring robot as long as the object is in the range of sensors' maximal abilities. The distance between robots defined in simulation is ideally from one robot's center to other robot's center, which is not the same as the distance defined for real robots. Also, the communication is supposed to be ideal, which means one robot can get signal as long as another robot stands in the communication range and sends signal. Moveover, the motions of a robot, including move forwards or backwards and turn left or right, are also ideally configured.

The parameters defined in simulation are also much more flexible than in real experiments. For example, the real robot is only possible to avoid another quiet robot when they are relatively colse, while this threshold can be defined to be $8cm$ in simulation.

Results of simulation

Two robots labeled by robot0 and robot1, respectively, are tested in a scenario of $120cm \times 180cm$.

The Figure 4.4 shows the moving traces of two robots, it can be seen that both robots do never touch the wall. Every time when they are close to the wall, they will rotate to change their moving direction.

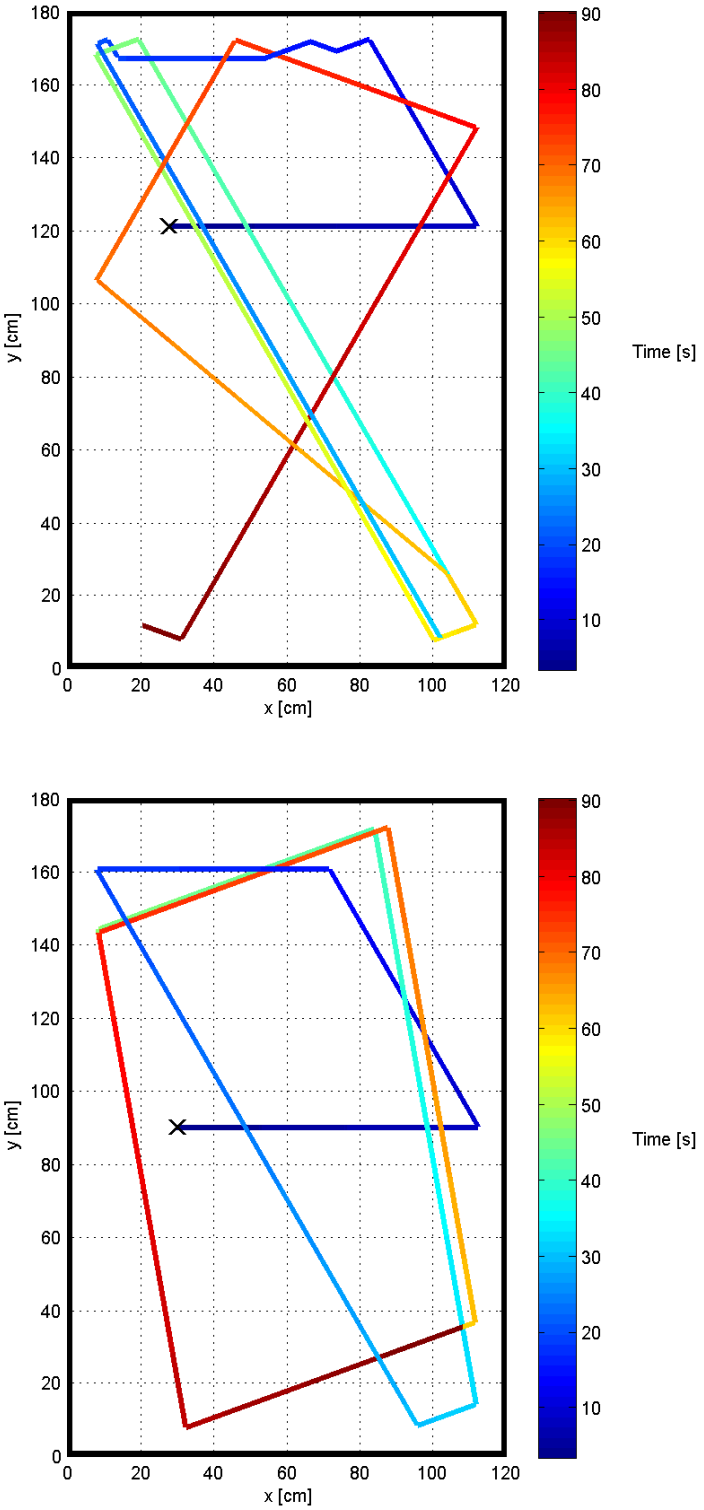
For the robots avoidance, it can be seen from Figure 4.5 that the distance between two robots is always larger than $0cm$. When they are close to each other with distance $8cm$, they will adjust their direction to avoid the crash.

Results of experiments

In the real experiments, the robots are supposed to move slowly. It gives the benefit to the robots that they could have more time to do sensing in a slowly changed environment. But during the experiments, it has been seen that the robots are hard to meet each other when let them run in a big scenario.

In order to test the behaviors of robots more efficiently, the scenario is not suggested to be used on this experiment as well as future experiments. The wall and robots will be set randomly by manual way. All the figures showing the results of experiments are clipped from videos.

The Figure 4.6 shows the wall avoidance. The robot meets the wall and then turn right to



(b) Robot1.

Figure 4.4: The moving traces of two robots in simulation. The sign of $\times$ shows the initial position. The color represents the time.

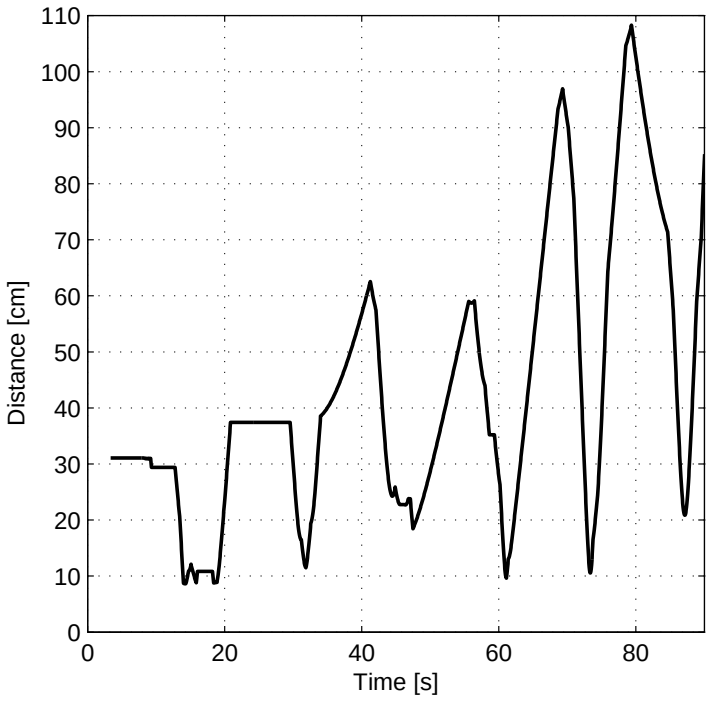


Figure 4.5: The distance between two robots.

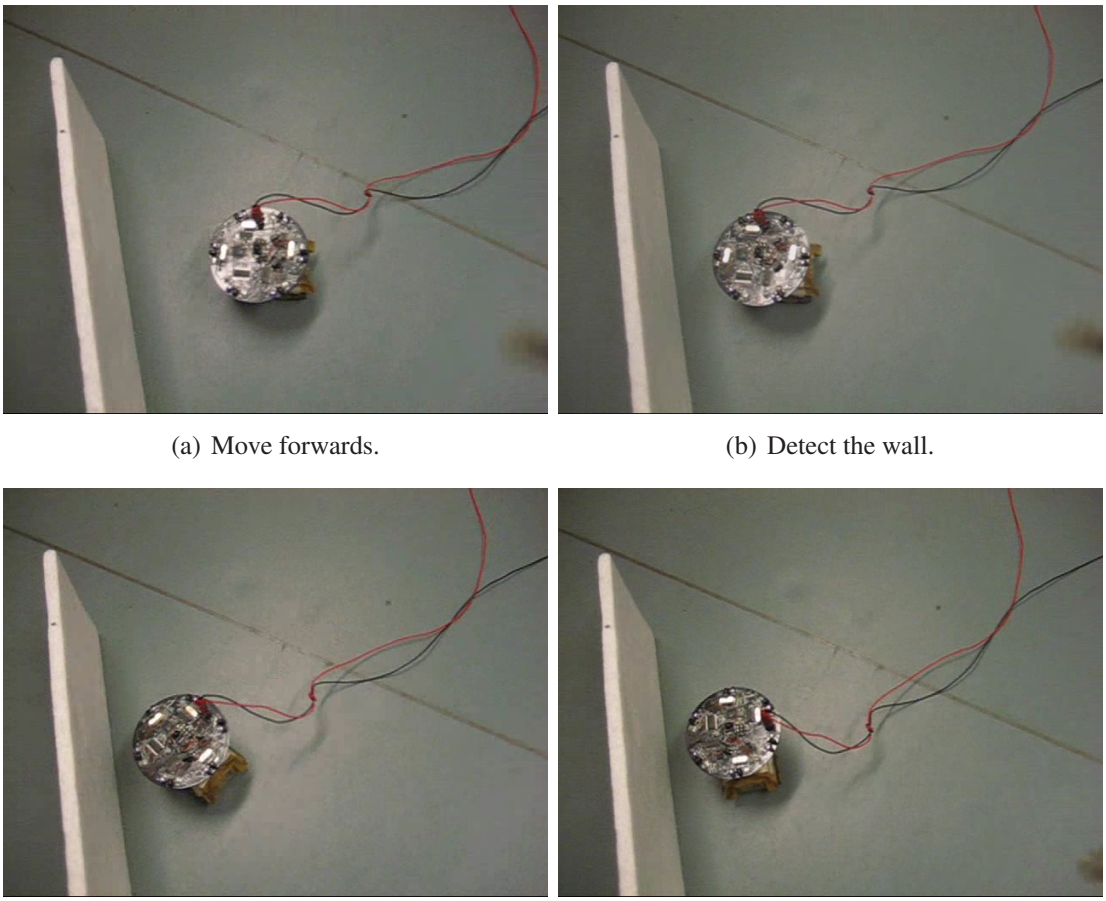
avoid it.

The Figure 4.7 shows the case when two robots meet each other without receiving communication signal. Both robots turn to the direction that there is no obstacle in front.

The Figure 4.8 shows the case when a robot meets the other robot and gets the communication signal. The robot detects the low voltage level whose ADC value is less than the threshold. Then the robot knows there is another robot in its front and that robot is quite colse to itself. Finally, the decision made by the robot is to move backwards.

4.4 Light source approaching

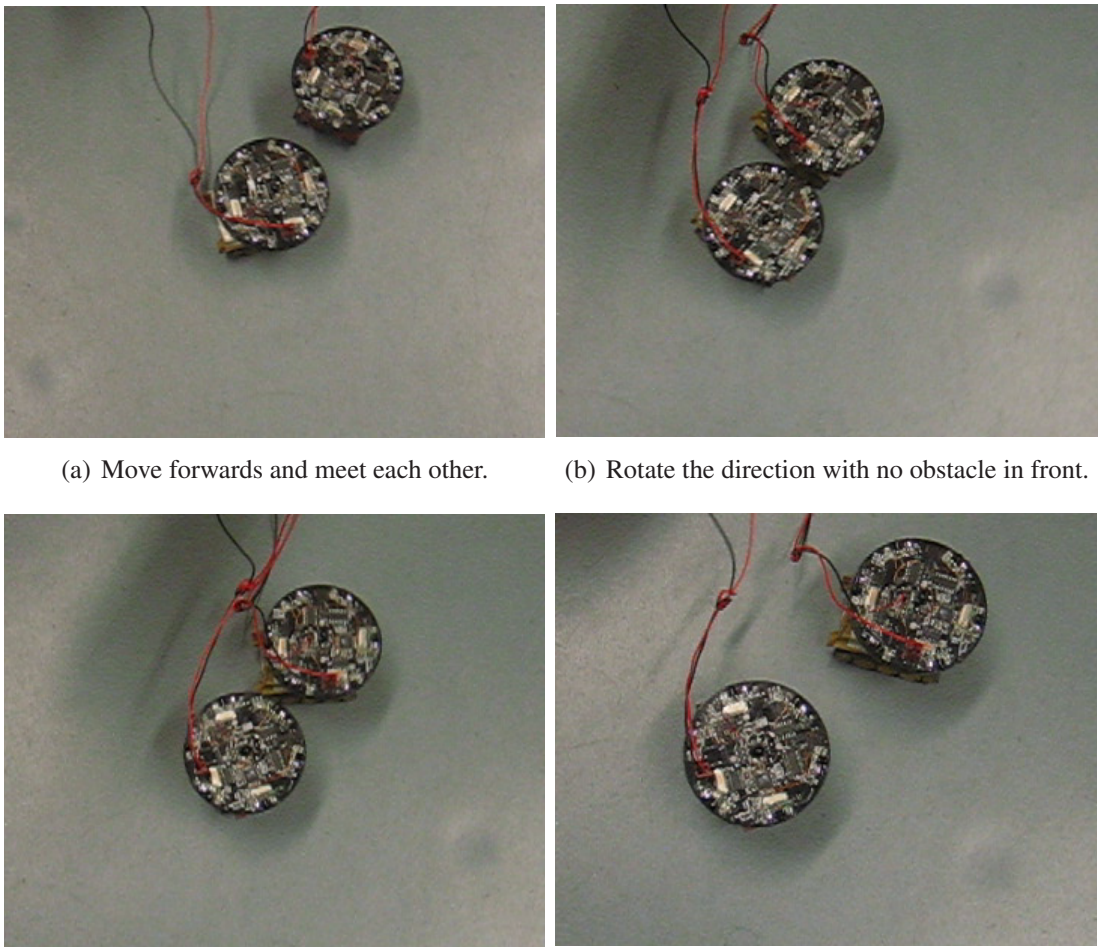
The robot is mounted with six ambient light sensors and the hardware functionality for detecting the light has been discussed in Section 3.4. In this section, the main task is to develop an algorithm for the robot approaching a light source.



(a) Move forwards. (b) Detect the wall.

(c) Turn right. (d) Avoid the wall.

Figure 4.6: Wall avoidance.



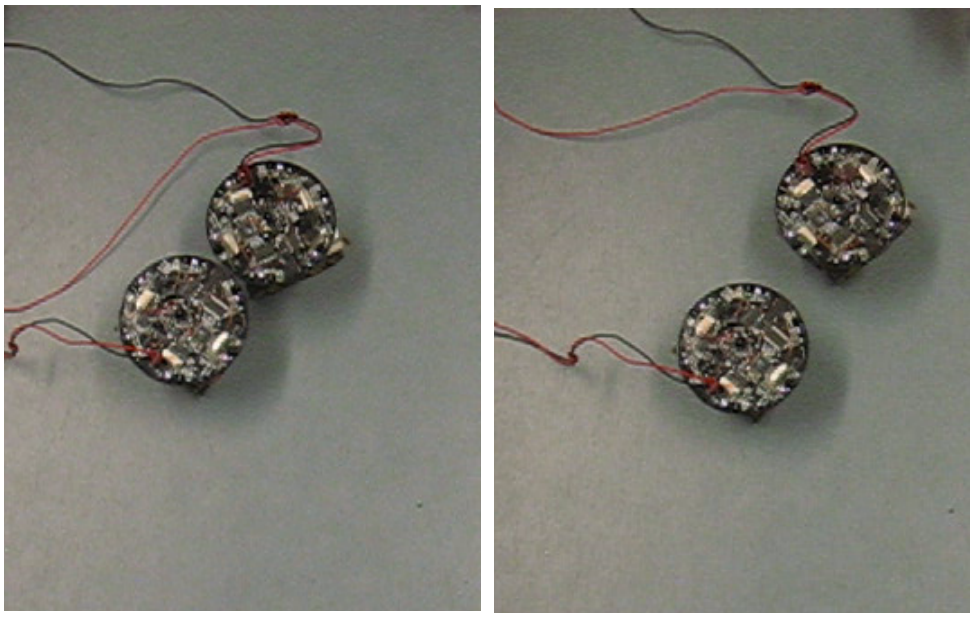
(a) Move forwards and meet each other.

(b) Rotate the direction with no obstacle in front.

(c) Continue to rotate.

(d) Ensure no obstacle in front and move forwards.

Figure 4.7: Robots avoidance without communication signal.



(a) Meet the other robot and estimate the distance from the communication signal.

(b) Move backwards.

Figure 4.8: Robots avoidance when receiving communication signal.

4.4.1 Analysis

In order to approach a light source, it is necessary to let the robot know where is the light source. By using the ambient light sensors, it can not detect the distance accurately since different object has different illuminant intensity. But with six ambient light sensors averagely mounted on 360° of the PCB, the direction of a light source can be estimated. This direction is defined as the error angle between the forward direction of the robot and the direction pointing to the light source. This defined error can be seen from Figure 5.6 to 4.12. Before to calculate this error, the relationship between six ambient light sensors' outputs and the direction of the light source is analyzed in the following.

The Figure 3.14 shows the ADC value from the front ambient light sensor when the robot is rotated clockwise from 0° to 180°. Because the ambient light sensors are symmetrically mounted around the PCB, when anti-clockwise rotating the robot, the ADC value from the front sensor is symmetrical to the value in the case of clockwise rotating. Keeping the distance to be 40*cm*, the samples from an experiment are drawn by the dashdot curve in Figure 4.9.

It can be seen that the in the range between -100° and 100°, the dashdot curve is similar to the shape of a parabola. Thereafter, the dashed line in Figure 4.9 shows an two-order approximated parabola calculated from the samples in that range. The formula of this

curve is written in Equation 4.1

$$y = -0.0634x^2 + 0.0138x + 812.9823 \qquad x \in [-100, 100] \tag{4.1}$$

On the experiment, a bulb was used as the light source, which has a bigger illuminant surface than an ideal point light source. Thus, if the robot is close to the bulb, the ADC value changed by the rotated angle is not as sensitive as in the case with longer distance. This comparison has already shown in Figure 3.14. When the distance is $20cm$, the curve can not be approximated by a parabola. Seen from the curve with distance of $40cm$, the bulb performs much close to the characteristic of an point light source.

In the following, the algorithm is developed with the assumption of a point light source.

The experimental result from the Figure 4.9 also implicates the distribution of detected ADC value on 360° of the PCB when a bulb stays away with distance of $40cm$. The 0° represents the reference point in the PCB which has the minimal distance to the bulb.

Therefore, it can get an inference that when a point light source points to a point of PCB labeled by 0°, the distribution of detectable ADC value from -180° to 180° with reference of the 0° point has the similar tendency as the dashdot curve in Figure 4.9.

4.4.2 Direction estimation

From the analysis about distribution of detectable ADC value, it could provide the robot with the ability to estimate the direction of a light source referred to it own forward direction. This error between two directions is calculated in the following.

One ambient light sensor can have the maximal value in six sensors when the light source points to the sector where it is mounted. Take the front ambient light sensor fixed on the sector 0 for example, only if the error were between -30° and 30°, the output from the front ambient light sensor could be larger than values from the other five sensors. And the values from two sensors beside the front one would be larger than three back sensors.

According to the largest and two second largest ADC values from two neighboring sensors, the error can be estimated. The error is defined to be in the range between -180° and 180°. In the following , three specific cases are discussed firstly and then the formula for error estimation is generated.

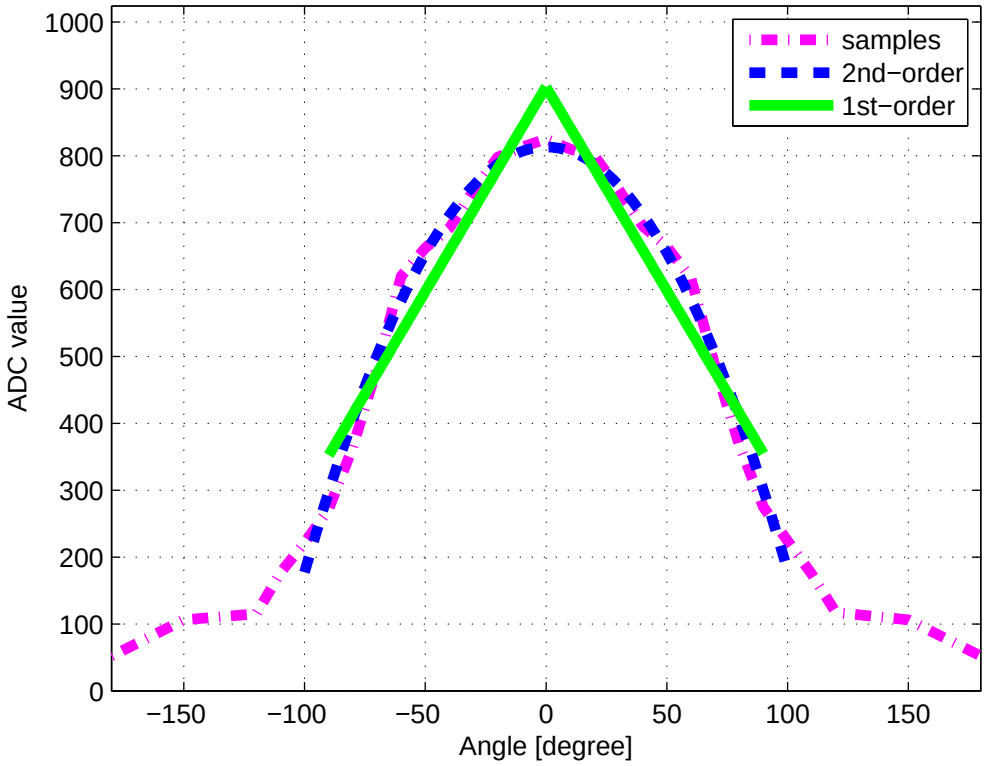
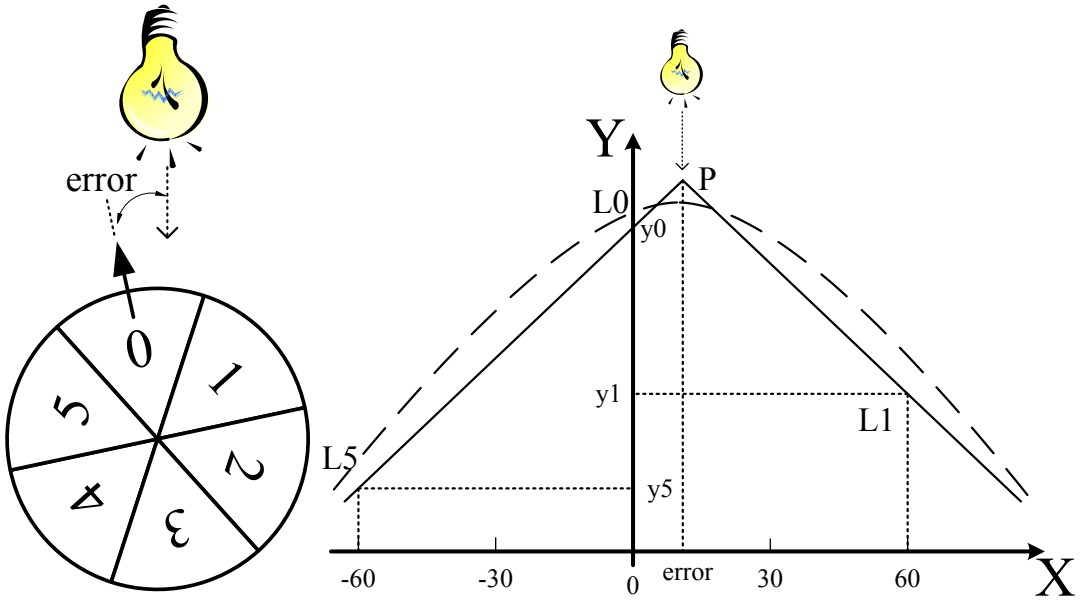


Figure 4.9: The distribution of detected ADC value from -180° to 180° when the distance is 40*cm*. 0° is the case when the bulb points to the front ambient light sensor. The dashdot curve is drawn from samples of the experiment. The dashed curve is the second-order approximation between -100° and 100°. The solid curve is the first-order approximation in the range from -90° to 90°.

Case one

The Figure 4.10(a) illustrates the situation when the froward direction deflects clockwise from the direction of the light source. The light source is indicated by a bulb. The error is defined to be positive in this relative direction. And also this error is less then 30° in order to make sure that the ADC value from sector0 is maximum.



(a) Relative direction when error is in the range between 0° and 30°.

(b) Approximated curves for calculating the error angle.

Figure 4.10: Direction estimation in case one.

Referring to the Figure 4.9, the approximated parabola is drawn by the dashed curve in Figure 4.10(b) in order for showing the tendency. The X-coordinate is the degree of angle where 0° represents the place mounted with the front ambient light sensor. The Y-coordinate is the ADC value. The vertex of the parabola is an point which the bulb points to. This point is also related with the error of direction.

The parabola can be described by a second-order formula written in Equation 4.2.

$$y = ax^2 + bx + c \qquad x \in [-60, 60] \tag{4.2}$$

Because three readings from sensors on sector0, sector1, and sector5 are satisfied to calculate the error, the independent variable, x, is only need to be considered in the range between -60° and 60°. Those three readings are represented by three points in Figure 4.10(b), that are $L0(0, y0)4$, $L1(60, y1)$ and $L5(-60, y5)$. Substituting three points into

Equation 4.2, three parameters a, b and c can be achieved. With the full expressed formula, the error can be calculated by Equation 4.3.

$$error = -\frac{b}{2a} \tag{4.3}$$

Even though these calculations are not expensive for MCU, but the limited memory size in the MCU is relatively expensive. Furthermore if using data of double type in order for a more accurate result, the flash memory in the MCU would be occupied more. Therefore, a first-order approximated function is suggested to be used.

The first-order approximation is shown by the solid curve in Figure 4.9. This approximation is calculated from the samples between -90° and 90°. The approximated function is shown in Equation 4.4.

$$y = \begin{cases} 6.1224x + 903.3091 & \text{if } x \leq 0 \\ -6.0903x + 902.9636 & \text{if } x > 0 \end{cases} \tag{4.4}$$

It can be seen that this first-order curve fits worse than the parabola, especially in the range beside the 0°, where the first-order curve is relatively acuate compared with the parabola which is gentle. But it contributes much on unloading the burden of MCU.

In Figure 4.10(b), the first-order approximated curve is drawn by the solid line with a peak point P. This curve can be described by Equation 4.5.

$$y = -k|x - error| + b \qquad x \in [-90, 90] \tag{4.5}$$

Where k is the slope of the increasing line on the left side. Substituting the three sample points, the results are written in Equation 4.6.

$$\begin{aligned} y0 &= -k|0 - error| + b \\ y1 &= -k|60 - error| + b \\ y5 &= -k|-60 - error| + b \end{aligned} \tag{4.6}$$

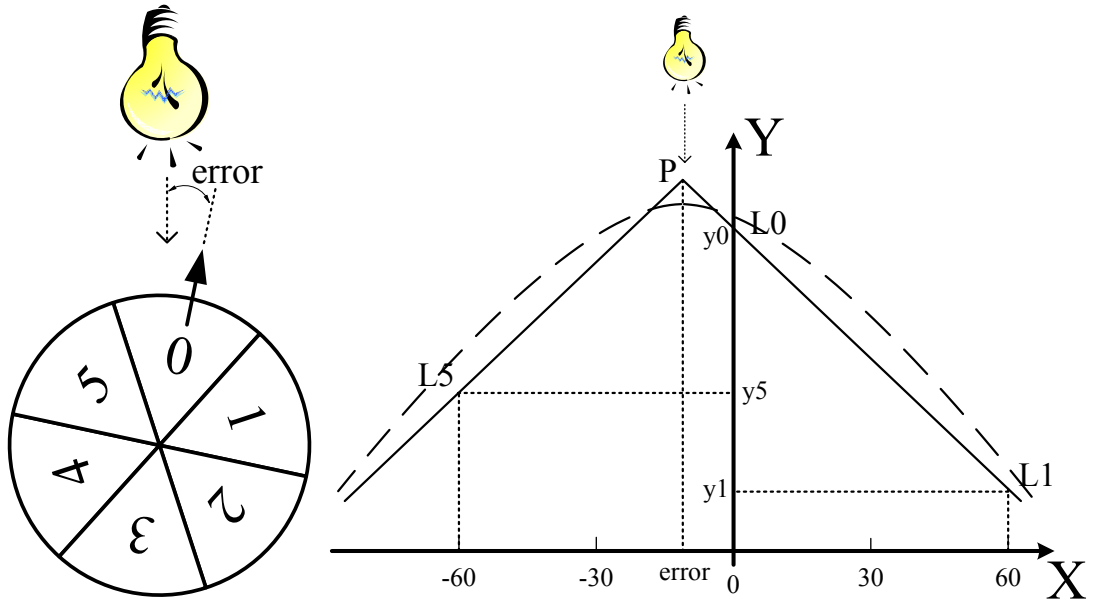
There are three functions with three unknown parameters. In this case, the error is between the range from 0° to 30°. Finally, the value of error can be solved by Equation

4.7

$$error = 30 \cdot \frac{y1 - y5}{y0 - y5} \tag{4.7}$$

Case two

In this case, shown in Figure 4.11(a), the forward direction of the robot is on the right side of the bulb, which is opposite to the relative direction in the case one. The error is defined to be a negative angle when the robot rotates anticlockwise. In order to ensure the maximal sensor value is from the sector0, this error is limited in the range between -30° and 0°.



(a) Relative direction when error is in the range between -30° and 0°.

(b) Approximated curves for calculating the error angle.

Figure 4.11: Direction estimation in case two.

The Figure 4.11(b) shows the the first-order and second-order approximated curves in this case. As similar to the computation in the case one, the error can be calculated by Equation 4.8.

$$error = -30 \cdot \frac{y5 - y1}{y0 - y1} \tag{4.8}$$

Case three

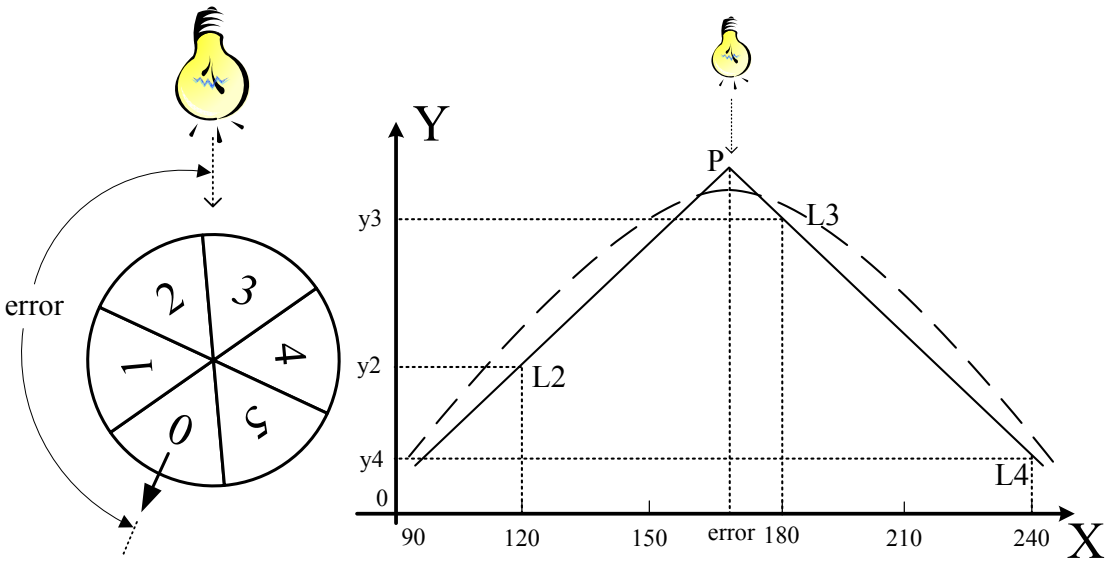
The third case is indicated in Figure 4.12(a). The bulb points to the back of the robot and the error is in the range from 150° to 180° In this case, the ambient light sensor on sector3 has the maximal ADC value. The sensors on the sector2 and sector4 have the second and third largest ADC value.

Based on these three values, the different angle between the error and 180° labeled in Figure 4.12(b) can be calculated by using the similar approach as in the case two. The formula is shown in Equation 4.9

$$error - 180 = -30 \cdot \frac{y2 - y4}{y3 - y4} \tag{4.9}$$

Therefore, the error can be got by Equation 4.10

$$error = 180 - 30 \cdot \frac{y2 - y4}{y3 - y4} \tag{4.10}$$



(a) Relative direction when error is in the range between 150° and 180°.

(b) Approximated curves for calculating the error angle.

Figure 4.12: Direction estimation in case three.

Formula of error calculation

From the analysis of the above three cases, it can be seen that the error can be estimated wherever the light source is in front or in back. The complete formula to calculate the error angle has been developed and written in Equation 4.11 and 4.12.

$$error_{360} = \begin{cases} 60 \cdot X + 30 \cdot \frac{V_{Right} - V_{Left}}{V - V_{Left}} & \text{if } V_{Right} \geq V_{Left} \\ 60 \cdot X - 30 \cdot \frac{V_{Left} - V_{Right}}{V - V_{Right}} & \text{if } V_{Left} > V_{Right} \end{cases} \tag{4.11}$$

Where V is the maximal ADC value detected from sector X. V_{Left} and V_{Right} are values read from sensors mounted on the left side and right side of sector X, respectively. $error_{360}$ is the error of angle defined in the range from 0° to 360°.

However, the supposed error angle is in the range between -180° and 180°. Therefore, in order to get the wanted *error*, the $error_{360}$ needs to be treated further according to the Equation 4.12.

$$error = \begin{cases} error_{360} & \text{if } error_{360} \leq 180 \\ error_{360} - 360 & \text{if } error_{360} > 180 \end{cases} \tag{4.12}$$

4.4.3 Algorithm

The algorithm used for robot approaching the light source has been developed and written in Algorithm 8.

Algorithm 8 Light source approaching()

Require: Ambient light detecting()

```
1: if Channel_with_Max_Lux == 2 then
2:     Turn right(120°)
3: else if Channel_with_Max_Lux == 4 then
4:     Turn left(120°)
5: else if Channel_with_Max_Lux == 3 then
6:     Turn left(180°)
7: else
8:     Direction estimating()
9:     P controlling()
10: end if
```

The robot needs to detect the light from six channels firstly. This process is written in Algorithm 9. Because all channels are connected to the pin ADC6 of MCU, the ADC

channel does not to be changed. The voltage levels from ambient light sensors are continuously sampled from ADC6 and the average of every two ADC values from the same channel will be saved in an array. In the meanwhile, the number of the channel which gets the maximal ADC value is recorded in an variable.

Algorithm 9 Ambient light detecting()

```
1: Max_Lux ⇐ 0
2: for Channel_Number = 0 to 5 do
3:    Select channel (Channel_Number)
4:    Ambient_Light[Channel_Number] ⇐ ADConvert(6)
5:    if Ambient_Light[Channel_Number] > Max_Lux then
6:       Max_Lux ⇐ Ambient_Light[Channel_Number]
7:       Channel_with_Max_Lux ⇐ Channel_Number
8:    end if
9: end for
```

On the implementation, if the detected light source is in the back of robot when the channel 2, 3 or 4 gets the maximal ADC value, the robot is supposed to rotate until it faces to the light source. Therefore, the error angle only needs to be estimated when channel 0, 1 or 5 gets the maximal ADC value. In these cases, the error angle will be in the range between -90° and 90°. Based on the analysis in the last section, the Algorithm 10 is developed to estimate the direction to the light source. The error angle returned from this algorithm can be used as feedback information for further controller implementation.

By taking information from sensors, the robot is supposed to approach the light source in a smooth way. This behavior can be realized by using the closed-loop control. A P-controller has been developed for this implementation and written in Algorithm 11.

In this closed-loop control system, The reference angle for the robot is 0°, that is the direction pointing to the light source. The feedback is the error angle calculated from sensors. So the value input to the P-controller is $(0 - Error)^\circ$.

This input value only has opposite sign to the error angle. Therefore, in the algorithm, the value of error angle is directly used as input signal to the P-controller. Firstly, the absolute value of error angle is used to judge the robot's motion. If the error is in the range of dead zone, the robot will move forwards at default speed. Otherwise, the P-controller will be utilized. A speedup value for the motor is decided by the P-gain and the absolute error value, which describes how much speed should be added on the default forward-moving speed of the left or right motor. This speedup value is also limited by a threshold defined as the potentially maximal speedup value. If the error angle is larger than 0°, the speedup value will be added on the left motor, which leads the robot to turn right. If the error angle

Algorithm 10 Direction estimating()

Require: Ambient light detecting()

if $Channel\_with\_Max\_Lux == 0$ **then**
  if $Ambient\_Light[1] > Ambient\_Light[5]$ **then**
    $Error \Leftarrow 30 \times \frac{Ambient\_Light[1]-Ambient\_Light[5]}{Ambient\_Light[0]-Ambient\_Light[5]}$
  else
    $Error \Leftarrow -30 \times \frac{Ambient\_Light[5]-Ambient\_Light[1]}{Ambient\_Light[0]-Ambient\_Light[1]}$
  end if
else if $Channel\_with\_Max\_Lux == 1$ **then**
  if $Ambient\_Light[2] > Ambient\_Light[0]$ **then**
    $Error \Leftarrow 60 + 30 \times \frac{Ambient\_Light[2]-Ambient\_Light[0]}{Ambient\_Light[1]-Ambient\_Light[0]}$
  else
    $Error \Leftarrow 60 - 30 \times \frac{Ambient\_Light[0]-Ambient\_Light[2]}{Ambient\_Light[1]-Ambient\_Light[2]}$
  end if
else if $Channel\_with\_Max\_Lux == 5$ **then**
  if $Ambient\_Light[0] > Ambient\_Light[4]$ **then**
    $Error \Leftarrow -60 + 30 \times \frac{Ambient\_Light[0]-Ambient\_Light[4]}{Ambient\_Light[5]-Ambient\_Light[4]}$
  else
    $Error \Leftarrow -60 - 30 \times \frac{Ambient\_Light[4]-Ambient\_Light[0]}{Ambient\_Light[5]-Ambient\_Light[0]}$
  end if
end if

Algorithm 11 P controlling()

Require: Direction estimating()

$Absolute\_Error \Leftarrow |Error|$
if $Absolute\_Error < Threshold\_of\_Deadzone$ **then**
  Move forwards()
else
  $Speedup\_Value\_of\_Motor \Leftarrow K_p \cdot Absolute\_Error$
  if $Speedup\_Value\_of\_Motor > Threshold\_of\_Max\_Speedup\_Value$ **then**
    $Speedup\_Value\_of\_Motor \Leftarrow Threshold\_of\_Max\_Speedup\_Value$
  end if
  if $Error > 0$ **then**
    Left motor speed up($Speedup\_Value\_of\_Motor$) {Turn right.}
  else
    Right motor speed up($Speedup\_Value\_of\_Motor$) {Turn left.}
  end if
end if

is less than 0°, the right motor will speed up and the robot will turn left.

4.4.4 Testing results

The Algorithm 8 has been tested in reality. A bulb is used as a light source in a dark environment. Because the bulb emits the light with strong IR, the IR sensors of the robot can not function. The P-gain is set to be 5 and the dead zone is in the range between -3° and 3°. The max speed is limited by the duty cycle of PWM with the number of 75%.

The Figure 4.13 shows six video clips from the experiment. It can be seen that the robot has the ability to discover the light source and approach it smoothly.

4.5 Color reaction

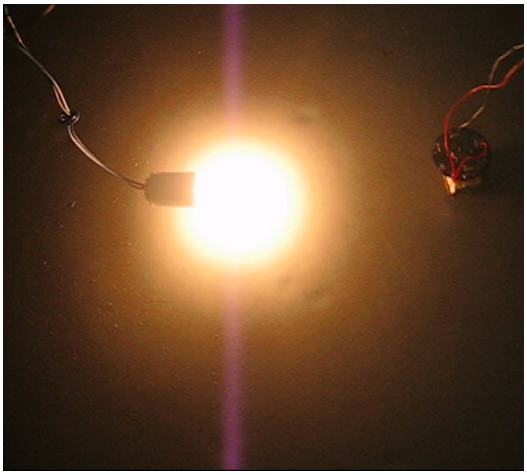
A color sensor is equipped in the front of the robot. The hardware functionality of color detection has been discussed in Section 3.4. Due to the limitation of hardware, only colors of red and blue can be detected by the robot. On the implementation of color reaction, an object with red color is assumed to be the thing that the robot is interested in. While, an object with blue color is what the robot is afraid of. When the robot recognizes different colors, it is expected to have reactions of approaching the red light and escaping from the blue light.

4.5.1 Analysis

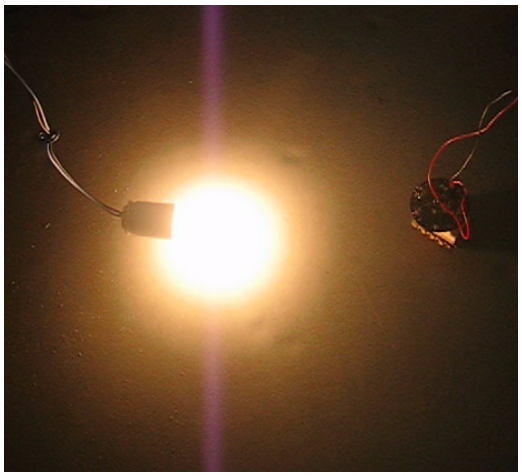
Before to have reaction, the robot needs to decide whether there is an object with colors in front of itself. If with colors, the robot has to further judge what the color is. On the implementation, there are three situations for the robot to decide, which are classified in the following:

1. No object with colors.
2. An object with the red light.
3. An object with the blue light.

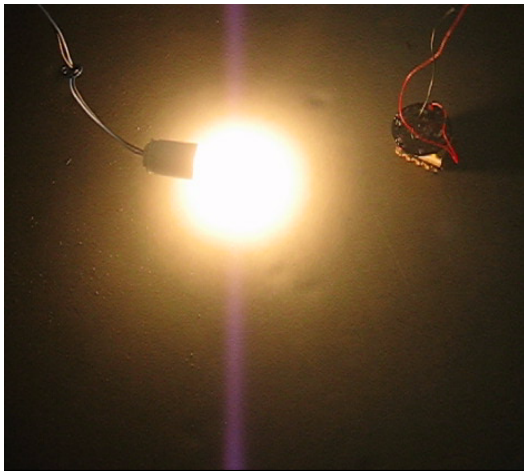
The robot has to distinguish the different situations by comparing two output frequencies when using red filter and blue filter. In a specific environment, the result in three situations



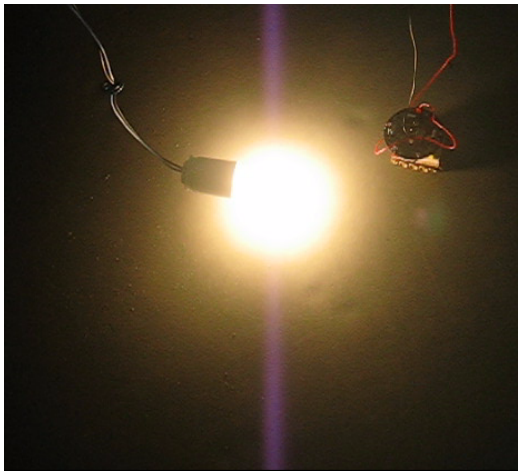
(a) Discover the light source in the back.



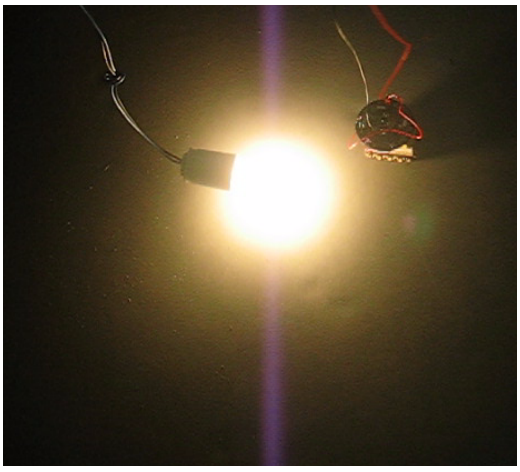
(b) Turn left.



(c) Right motor speeds up.



(d) Move with right motor speeding up.



(e) Move with right motor speeding up.

(f) Approach the light source.

Figure 4.13: The video clips of light source approaching.

has been plotted in Figure 3.16.

In situation one, it can be seen from Figure 3.16(a) that the frequency output from red filter is always higher than using the blue filter. This result fits to the characteristic of the photodiode spectral responsivity of the color sensor [tcs04]. In that environment, the values from the red filter and blue filter are below 30kHz and 10kHz, respectively.

In situation two, referring to Figure 3.16(b), if the distance is less than 15cm, the value from the red filter is much larger than the value from the blue filter. At the distance of 15cm, the value from the red filter is three times of the value from the blue filter. The less distance, the more times. If the distance is more than 15cm, both values tend to be the values as similar as in the situation one.

In situation three, seen from Figure 3.16(c), the blue light can be detected if the distance is less than 5cm, during which the value from the blue filter is larger than the value from the red filter. If the distance is farther than 5cm, both values tend to be the case when there is no color light in front.

4.5.2 Algorithm

From the analysis above, the color reaction for the robot is written in Algorithm 12.

Algorithm 12 Color reacting()

```
1:  Color deciding()
2:  if Color == clear then
3:     Turn left(60°)
4:     Times_of_Turn_Left++
5:     if Times_of_Turn_Left > 5 then
6:        Move forwards()
7:        Times_of_Turn_Left ⇐ 0
8:     end if
9:  else if Color ⇐ red then
10:    Move forwards()
11: else if Color ⇐ blue then
12:    Move backwards()
13:    Turn left(180°)
14:    Move forwards()
15: end if
```

Firstly, the robot needs to decide the color by analyzing the data from color sensor. The process of color decision has been written in Algorithm 13.

Algorithm 13 Color deciding()

Require: Timer/Counter0 overflow ISR()

1: $PD3 \Leftarrow 0$ {Select the red filter}
2: $TCNT0 \Leftarrow 0$ {$TCNT0$ is the register of Timer/Counter0 counter value.}
3: $Count\_of\_Timer0\_Overflow \Leftarrow 0$
4: Delay ms(1) {Delay $1ms$}
5: $Red\_Filter\_Frequency \Leftarrow 256 \cdot Count\_of\_Timer0\_Overflow + TCNT0$
6:
7: $PD3 \Leftarrow 1$ {Select the blue filter}
8: $TCNT0 \Leftarrow 0$
9: $Count\_of\_Timer0\_Overflow \Leftarrow 0$
10: Delay ms(1)
11: $Blue\_Filter\_Frequency \Leftarrow 256 \cdot Count\_of\_Timer0\_Overflow + TCNT0$
12:
13: **if** $Red\_Filter\_Frequency > 30$ && $Red\_Filter\_Frequency > 3 \times Blue\_Filter\_Frequency$ **then**
14: $Color \Leftarrow red$
15: **else if** $Blue\_Filter\_Frequency > Red\_Filter\_Frequency$ **then**
16: $Color \Leftarrow blue$
17: **else**
18: $Color \Leftarrow clear$ {$clear$ represents no color.}
19: **end if**

Algorithm 14 Timer/Counter0 overflow ISR()

1: $Count\_of\_Timer0\_Overflow$++ {To record the overflow times.}

The output frequencies when using red and blue filters are calculated in the beginning. The Timer/Counter0 counts the times of falling edges of the square wave output from the color sensor during $1ms$. An variable is set to count the times of Timer/Counter0 being overflow, which is shown in Algorithm 14. The real frequency has to be one thousand times of the value calculated by that variable and the number in the register $TCNT0$. Then, based on the analysis of three situations, the color can be distinguished. And the color will be recorded in an variable.

Secondly, with the information of the kind of color, the robot will response on different colors. If no color detected, the robot will turn left 60° and try to detect any color again. But if the robot has rotated 360° with no color detected, it will move forwards to another position. This process of color searching is repeated until any color is detected. If detected color is red that the robot is interested in, the robot will move forwards to approach it. If detected color is blue that the robot is afraid of, the robot will move backwards, then turn left 180° and move forwards to escape from the object with blue color.

4.5.3 Testing results

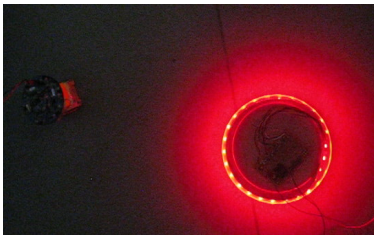
The Algorithm has been validated in the experiments. Figure 4.14 shows nine consecutive video clips when the robot reacts to the object with red light. The robot will move forwards if it detects the red light in front. But the moving direction can not be ensured to be directly pointing to the red light. So the robot will lose the detected color and need to detect the red light again. Finally, the robot is successful to approach the object with red light.

Figure 4.15 shows the reaction of the robot when there is a object with blue light in front. The robot succeeds to escape from it.

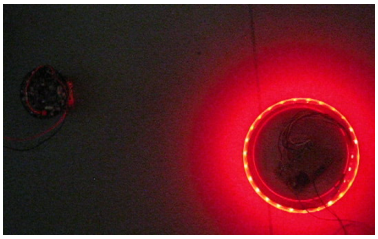
4.6 Summary

In this section, five abilities of a single robot have been discussed. All the abilities are realized based on the hardware functionalities.

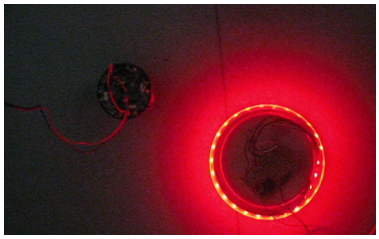
For robots' communication, the protocol is established. The realization of communication mainly relies on the UART. The baud rate is $9600bps$. The data for communication is packaged in one character. Besides the information of robot ID and channel ID. There are also three bits left that can be used for different aims. The algorithms for sending and



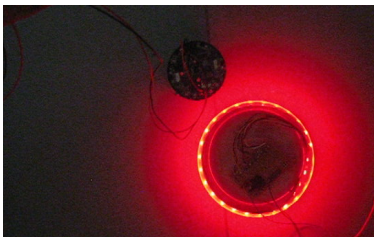
(a) Turn left and try to detect any color.



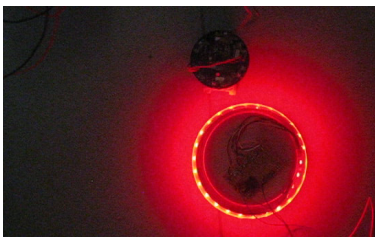
(b) Detect the red light.



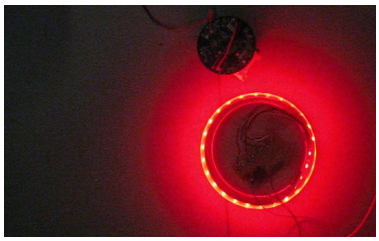
(c) Move forwards to approach it.



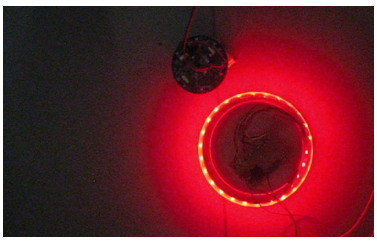
(d) Detect no color.



(e) Turn left.

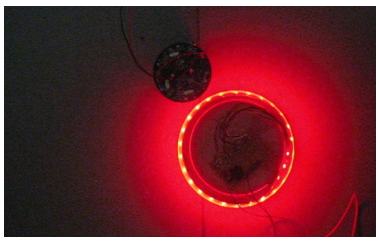


(f) Turn left.



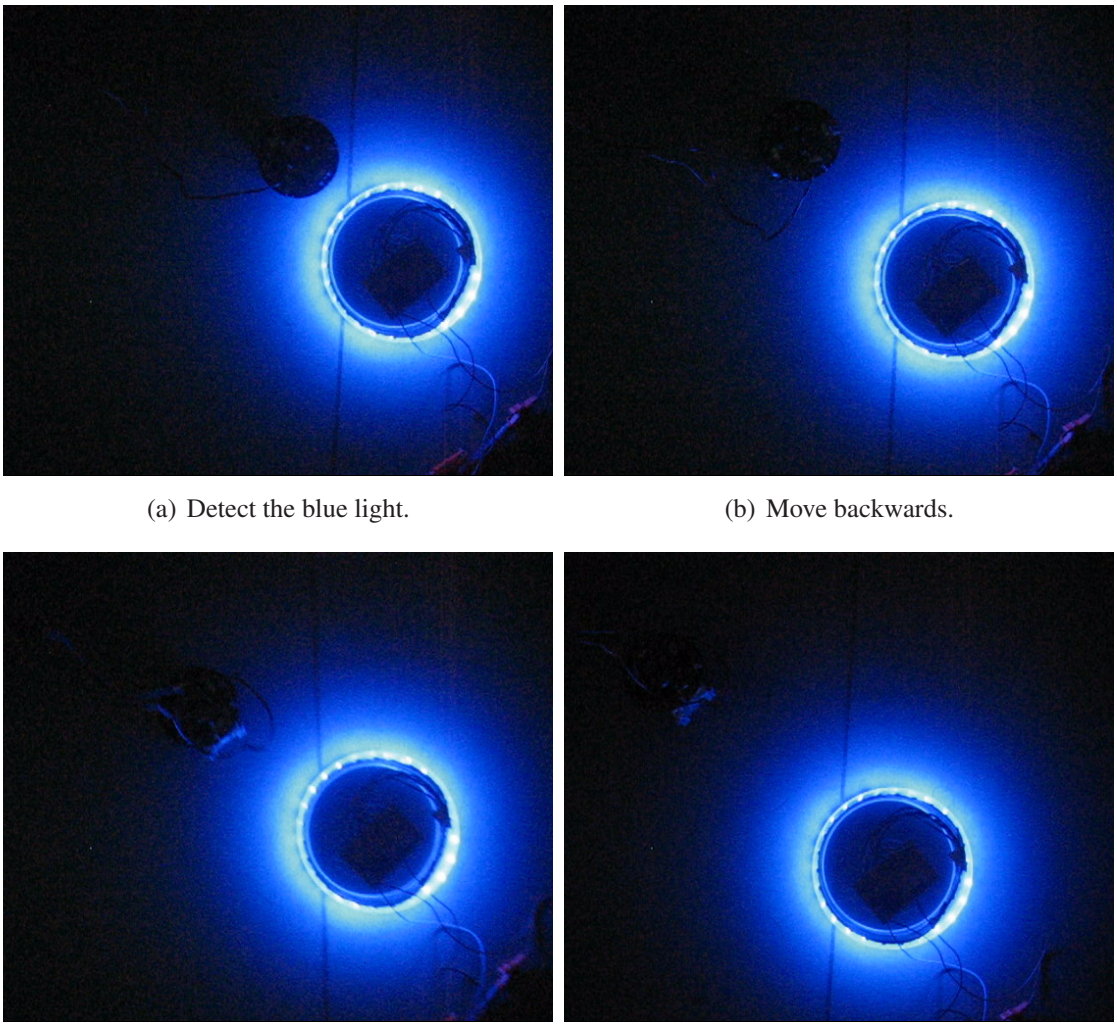
(g) Detect the red light again and move forwards.

(h) Detect no color.



(i) Turn left and detect the red light again.

Figure 4.14: The video clips of red light reaction.



(a) Detect the blue light.

(b) Move backwards.

(c) Turn left 180°.

(d) Move forwards.

Figure 4.15: The video clips of blue light reaction.

receiving signals and are developed in Section 4.1. Since the channel for signal sending and receiving needs to be switched, it has a chance to lose the communication signal even robots are close to each other. The useful information is extracted from the received signals.

The algorithm for estimating the distance by using the reflected IR is developed in Section 4.2. Because in most cases, the robot moves in front direction and only needs to know the distance to objects on its moving way, only three front IR couples are used for this objective.

The robot is necessary to avoid any potential crash. A robot needs to avoid the wall as well as other robots. Because the reflector surface from the other robot is relatively small, it prefers to estimate the distance between robots by communication signal. But in reality, a robot will keep quiet and send no communication signal out. In this case, to avoid another robot is not as easy as to avoid the wall. Considering both obstacles, the algorithm of obstacle avoidance is developed in Section 4.3 and has been validated both in simulation and real experiments.

With the help of six ambient sensors, the robot can estimate the direction of the light source. By knowing the error angle to the light source, a P-controller is implemented to control the robot to approach it in a smooth way. The algorithms for direction estimation and light source approaching are developed in Section 4.4 and have been validated in the experiments.

The robot can use the color sensor to distinguish the red and blue color. The algorithm is developed in Section 4.5 as well as the reaction to the different colors. The algorithm has been proved in the experiment that the robot can succeed to approach the red light and avoid the blue light.

Chapter 5

Formation

In this chapter, one of the swarm robots behaviors, that is formation, is implemented by two robots. The formation is discussed in two cases. One is only relative direction is kept after the formation is generated. The other is both relative direction and distance are always kept. For each case, separated algorithms are developed for each robot with different roles. The algorithms for the first case are validated in both simulation and experiments. While the algorithms for the second case are only validated in simulation.

5.1 Formation analysis

Two micro robots are implemented in the formation. In order to clearly distinguish robots from each other, they are given the names of robot0 and robot1, respectively. Two robots are supposed to keep a specific formation.

A specific formation can be defined by the relative direction and distance of two robots. On this implementation, the relative direction is defined to be that the robot0's sector5 points to the robot1's sector2, and the relative distance is defined to be $12cm$. This formation is depicted in Figure 5.1.

The process of formation can be realized by two steps. Firstly, two robots generate the formation. Then, the formation has to be kept when they are moving. These two steps are discussed in the following.

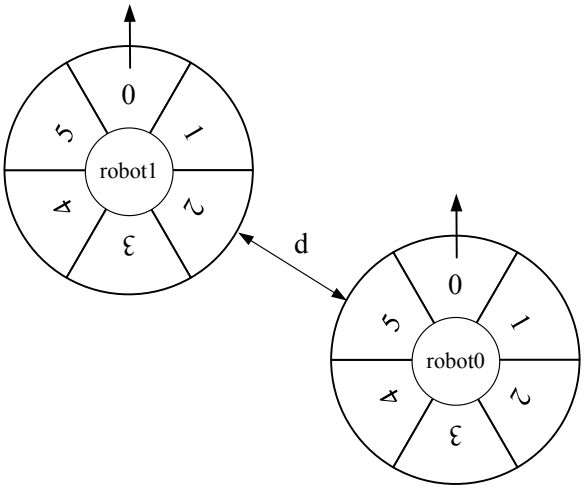


Figure 5.1: Formation shape. The relative direction is robot0's sector5 pointing to the robot1's sector2. The *d* represents the relative distance, 12*cm*.

5.1.1 Formation generation

Two robots are initialized to be moving freely. If they meet each other, two robots are supposed to perform formation. Formation are defined by two factors as mentioned above, including relative direction and distance.

Formation generation can be realized by two approaches. Two robots can perform formation with each other or only one moving robot performs formation with the other stop one.

Using the first approach, the relative direction is much easier to be corrected by two robots adjusting in the meanwhile. They only need to rotate themselves to make sure the correct sector points to the other robot. But, there exists a problem when adjusting the distance. Because in most cases, the communicating IR couples on two robots do not point to each other in a straight line, the sensors' readings for the same distance will be different for two robots. Therefore, two robots will stick on the process of distance adjusting.

Using the second approach, the distance is only confirmed by the moving robot. It can avoid the problem existing in the first approach. However, for direction adjusting, the moving robot needs to perform more motion to find the expected relative direction and it will take more time than the time spent in the first approach.

As a compromise, two robots can use the first approach to do direction adjusting and then use the second approach to do the distance adjusting. It is relative ideal when they generate the formation. However, when they are then moving together after formation is generated, in order to keep the direction relative to each other, two robots have to

change the moving direction dynamically. Actually, the further objective of swarm robots formation is for doing other specific tasks. Otherwise, only doing formation has no value in application. Therefore, if the moving direction were changed frequently, it would be no sense for specific tasks.

Therefore, the second approach is used to do implementation. The complete expected procedures in formation generation is stated in the following:

1. Two robots are moving freely.
2. If they meet each other, the robot0 asks the robot1 to stop.
3. The robot0 adjusts direction $00to12$.
4. If direction $00to12$ is ok, the robot0 adjusts distance.
5. If distance is ok, the robot0 adjusts direction $05to12$.
6. If both direction and distance are ok, they go to the next step.

The main tasks in the above procedures are direction and distance adjustments. The algorithms for realizing these two processes are developed in the following.

Direction adjustment

The expected relative direction is that the robot0's sector5 points to the robot1's sector2, which is shortly represented by $05to12$ and the same notation is used in this report. Take the 05 for example, the first number 0 represents the robot ID and the second number 5 represents the sector number.

The Algorithm 15 is developed for adjusting the direction of $00to12$. The approach to do direction adjustment consists of two steps. Firstly, the robot0 has to move to the relative direction of the robot1's sector2. Secondly, the robot0 adjusts its own direction to realize that its own sector0 points to the robot1's sector2.

This algorithm is developed only for one specific relative direction. However, the algorithms for other relative direction adjustments will not be written since the approach is the similar.

Algorithm 15 Direction adjusting(00 to 12) for Robot0

```
1: if Neighbor_Robot_Channel == (0||1||5) then
2:     if My_Received_Channel == 1 then
3:         Move forwards()
4:     else if My_Received_Channel == (2||3||4) then
5:         Turn right(60°)
6:     else if My_Received_Channel == (0||5) then
7:         Turn left(60°)
8:     end if
9: else if Neighbor_Robot_Channel == (3||4) then
10:    if My_Received_Channel == 5 then
11:        Move forwards()
12:    else if My_Received_Channel == (0||1||2) then
13:        Turn right(60°)
14:    else if My_Received_Channel == (3||4) then
15:        Turn left(60°)
16:    end if
17: else if Neighbor_Robot_Channel == 2 then
18:    if My_Received_Channel == 0 then
19:        Stop()
20:    else if My_Received_Channel == (1||2||3) then
21:        Turn right(60°)
22:    else if My_Received_Channel == (4||5) then
23:        Turn left(60°)
24:    end if
25: end if
```

Distance adjustment

The distance is decided by the robot0. Because the motors are easy to be slipping, the distance can not be adjusted precisely by controlling the robot's moving and turning in the meanwhile. That is also why two robot need to realized direction $00to12$ firstly. After the direction adjustment $00to12$ is done, the robot0 can only move forwards or backwards to adjust the distance. The Algorithm 16 is developed for adjusting the distance to be the reference distance by using a P-controller. If the distance is farther, the robot0 will keep moving forwards with a lasting time defined by the *P_gain* and error distance. If the distance is shorter, the robot0 will keep moving backwards with a lasting time. A dead zone is set, in which, the robot0 believes the relative distance is satisfied.

Algorithm 16 Distance adjusting() for Robot0

1: $Motion\_Time \Leftarrow P\_gain \cdot |Distance\_to\_Neighbor\_Robot - reference\_distance|$
2: **if** $Distance\_to\_Neighbor\_Robot > reference\_distance + deadzone\_distance$ **then**
3: Move forwards()
4: **else if** $Distance\_to\_Neighbor\_Robot < reference\_distance - deadzone\_distance$ **then**
5: Move backwards()
6: **else**
7: Stop()
8: **end if**
9: Delay ms($Motion_T ime$)
10: Stop()

5.1.2 Formation keeping

After the two robots finish formation generation, they are supposed to move forwards together with keeping the formation.

Keeping the formation implicates two robots have to keep the relative distance and direction during the motion. Due to the truth that the distance decided from the sensors are much influenced by the deflected angle, the robot0 will judge frequently that the distance is wrong. But on the implementation, the relative directions for doing distance adjustment and for this specific formation are not the same. Therefore if the robot0 thinks the distance is not correct, it needs to perform `Direction adjusting(` $00to12$`)`, `Distance adjusting()` , and then `Direction adjusting(` $05to12$`)`. Actually in reality, each process is not easy to be performed due to the problems of losing communication and slippage between the wheel and pedrails. Therefore, on the implementation, only the

relative direction is ensured to be kept during the motion. The whole formation procedures will be performed again only if the relative direction is lost. However, in the simulation, both the relative direction and distance can be ensured to be kept during the formation. Classified by these two approaches to keep the formation, the realization of formation will be discussed in next two sections, respectively.

5.2 Formation – case one

When the robot0 finishes formation generation, it only makes sure that the relative direction is correct during they are moving. Only if the relative direction is lost, they have to perform formation again. This case will be discussed in this section. The separated algorithms for each robot are developed with respect to their different roles. Also, the algorithms are validated in both simulation and experiments.

5.2.1 Algorithms

The Figure 5.2 shows the flow chart of the complete algorithm for both robot0 and robot1. Hardware is initialized firstly and then a loop runs infinitely. During the loop, the robot will measure distance and receive signal firstly. The data got from those two processes will be analyzed to decide whether or not to do obstacle avoidance. If an obstacle is avoided, the loop will go back to the beginning. Otherwise, the robot will perform an expected behavior. The two robots have different behaviors according to their own responsibilities. The procedures to perform a specific behavior for both robots are the same. The robot has to analysis the information got from sensors firstly. Then, the robot will make a decision and perform the decided behavior. Thereafter, the whole loop will be repeated.

The algorithm `BEHAVIOR()` is the main part to realize an expected behavior, inside which different algorithms have to be developed by considering the role of each robot. This main algorithm is realized by three sub-algorithms. In the algorithm `Signal analyzing()` , the pre-extracted data from received signals will be analyzed to further extract the useful information depending on different aims. In the algorithm `Decision making()` , a decision will be made that is judged by its previous state and the information got from the last algorithm. In the algorithm `Executing()` , an expected behavior decided by the last algorithm will be executed. In the following, the algorithms for robot0 and robot1 to realize the formation will be described in detail.

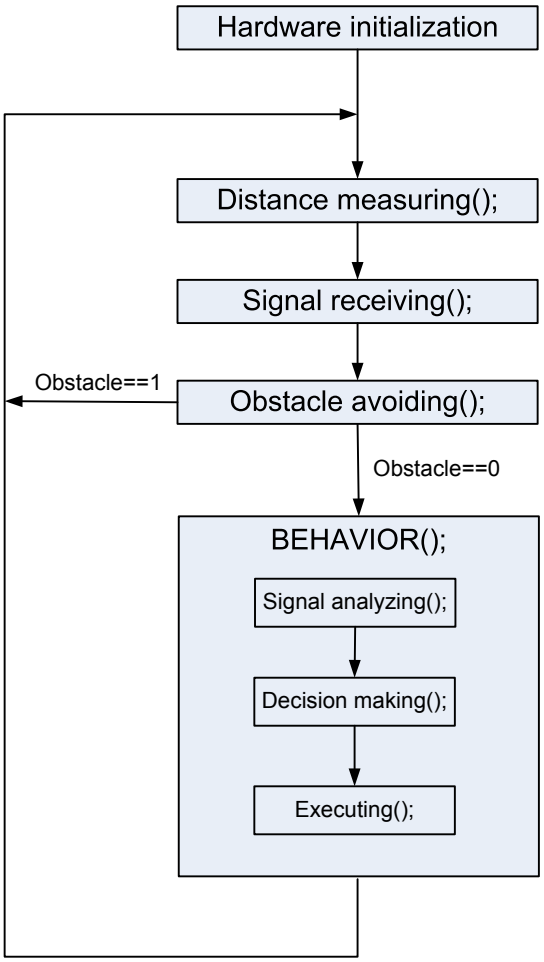


Figure 5.2: Flow chart for both robots in formation-case one.

Robot0

The robot0 has to generate and keep the formation with robot1. The relative direction and distance are both adjusted by the robot0, which plays an important role to realize the formation.

Algorithm 17 Signal analyzing() for Robot0 in formation-case one

```
1: No_signal ⇐ 0
2: Direction_00to12_Ok ⇐ 0
3: Direction_05to12_Ok ⇐ 0
4: Distance_Ok ⇐ 0
5: if There_is_a_Neighbor_Robot == 1 then
6:    Signal ⇐ Status_of_Neighbor_Robot
7:    if Neighbor_Robot_Channel == 2 then
8:       if My_Received_Channel == 0 then
9:          Direction_00to12_Ok ⇐ 1
10:      else if My_Received_Channel == 0 then
11:         Direction_05to12_Ok ⇐ 1
12:      end if
13:   end if
14:   if Distance_to_Neighbor_Robot ∈ [reference_distance ± deadzone_distance] then
15:      Distance_Ok ⇐ 1
16:   end if
17: else
18:   No_Signal ⇐ 1
19: end if
```

The Algorithm 17 illustrates the signal analyzing process. Five useful variables are taken out from the received signals, involving the information of data, relative direction and distance. The variable, *Signal*, is used to record the status of the other robot, which also represents the main information that robot1 want to tell robot0.

The Figure 5.3 is the Finite State Machine (FSM) diagram to illustrate the transitions between states and the actions in each state. During every main loop, only one state in the Figure 5.3 can be executed. Then the function will be returned to begin the next main loop. The state which will be executed in the current loop is decided by the last state and the five variables from the previous signal analyzing.

The formation is mainly achieved by states from 2 to 5 with respect to the excepted formation process. If the robot0 believes the formation is correct, it will send out the signal to let the robot1 move. Otherwise, it will always ask the robot1 to be stopped. *Counter* is to count the times of losing direction $05to12$. If the number is bigger than a threshold,

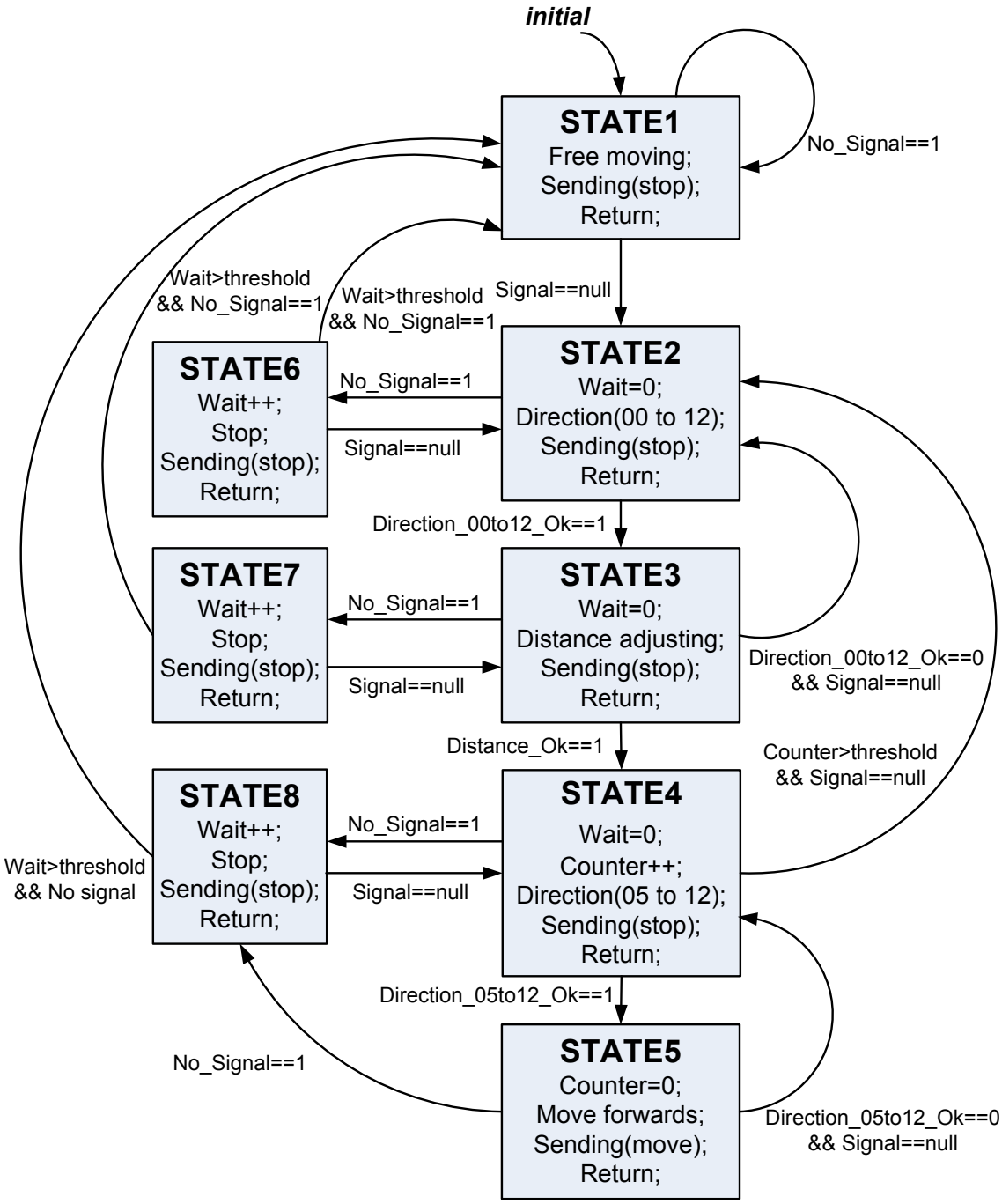


Figure 5.3: FSM of robot0 in formation-case one.

the previous adjusted distance is not believable. Thus, the robot0 has to adjust direction and distance from the beginning. There are also three assistant states 6, 7 and 8 which are used for the robot0 to stay when no signal is got in the current loop but there was signal before. A variable *Wait* is to counter the waiting times. If the robot0 have waited more times than a threshold, it will go back to move freely. If there is signal got again during waiting, the robot0 will go back to the previous one of formation states.

This state machine has been written in Algorithm 18 for deciding a state and Algorithm 19 for executing an action.

Robot1

During the formation generation, the robot1 performs like a landmark and waits for the robot0 to finish formation. Then when they are moving, the robot1 behaves as a leader and let the robot0 keep formation with itself.

The Algorithm 20 shows the signal analyzing for the robot1. The robot1 only needs to know whether there exists signal or not. The received signal will only include the information to let it move.

The Figure 5.4 shows the FSM for robot2 including the transitions between four states and the actions in each state. The formation states are state 2 and 3. The robot1 will be stopped in state 2 when the robot0 performs formation. The state 3 represents the situation that the robot1 is moving with formation of the robot0. The state 4 is for waiting and the state 1 is for free moving without formation. During all the four states, the robot1 always sends out the signal *null*. It is noted that *null* does not mean no signal, which represents there is no special request on the other robot and the contents of the data are only robot ID and channel ID. Only one state will be executed during one loop. Then after return, the loop will be repeated again.

Referring to FSM in Figure 5.4, the Algorithm 21 and 22 shows the decision making and executing process, respectively.

5.2.2 Testing results

The algorithms are implemented in both simulation and experiments. During the distance adjustment, the dead zone is defined from $9cm$ to $15cm$ and the P_gain is set to be 1.5.

Algorithm 18 Decision making() for Robot0 in formation-case one

```
1: if Last_State == 1 then
2:     if Signal == null then
3:         State ⇐ 2
4:     end if
5: else if Last_State == 2 then
6:     if No_Signal == 1 then
7:         State ⇐ 6
8:     else if Direction_00to12_Ok == 1 then
9:         State ⇐ 3
10:    end if
11: else if Last_State == 3 then
12:    if No_Signal == 1 then
13:        State ⇐ 7
14:    else if Distance_Ok == 1 then
15:        State ⇐ 4
16:    else if Direction_00to12_Ok == 0 then
17:        State ⇐ 2
18:    end if
19: else if Last_State == 4 then
20:    if No_Signal == 1 then
21:        State ⇐ 8
22:    else if Direction_05to12_Ok == 1 then
23:        State ⇐ 5
24:    else if Counter > threshold then
25:        State ⇐ 2
26:    end if
27: else if Last_State == 5 then
28:    if No_Signal == 1 then
29:        State ⇐ 8
30:    else if Direction_05to12_Ok == 0 then
31:        State ⇐ 4
32:    end if
33: else if Last_State == (6||7||8) then
34:    if Signal == null then
35:        State ⇐ Last_State − 4
36:    else if Wait > threshold&&No_Signal == 1 then
37:        State ⇐ 1
38:    end if
39: else
40:    State ⇐ 1
41: end if
42: Last_State ⇐ State
```

Algorithm 19 Executing() for Robot0 n formation-case one

1: **if** $State == 1$ **then**
2: Free moving()
3: **else if** $State == 2$ **then**
4: $Wait \Leftarrow 0$
5: Direction adjusting($00 to 12$)
6: **else if** $State == 3$ **then**
7: $Wait \Leftarrow 0$
8: Distance adjusting()
9: **else if** $State == 4$ **then**
10: $Wait \Leftarrow 0$
11: $Counter$++
12: Direction adjusting($05 to 12$)
13: **else if** $State == 5$ **then**
14: $Counter \Leftarrow 0$
15: Move forwards()
16: **else if** $State == (6||7||8)$ **then**
17: $Wait$++
18: Stop()
19: **end if**
20: **if** $State == 5$ **then**
21: Signal sending($stop$)
22: **else**
23: Signal sending($move$)
24: **end if**
25: **Return**

Algorithm 20 Signal analyzing() for Robot1 in formation-case one

1: $No\_signal \Leftarrow 0$
2: **if** $There\_is\_a\_Neighbor\_Robot == 1$ **then**
3: $Signal \Leftarrow Status\_of\_Neighbor\_Robot$
4: **else**
5: $No\_Signal \Leftarrow 1$
6: **end if**

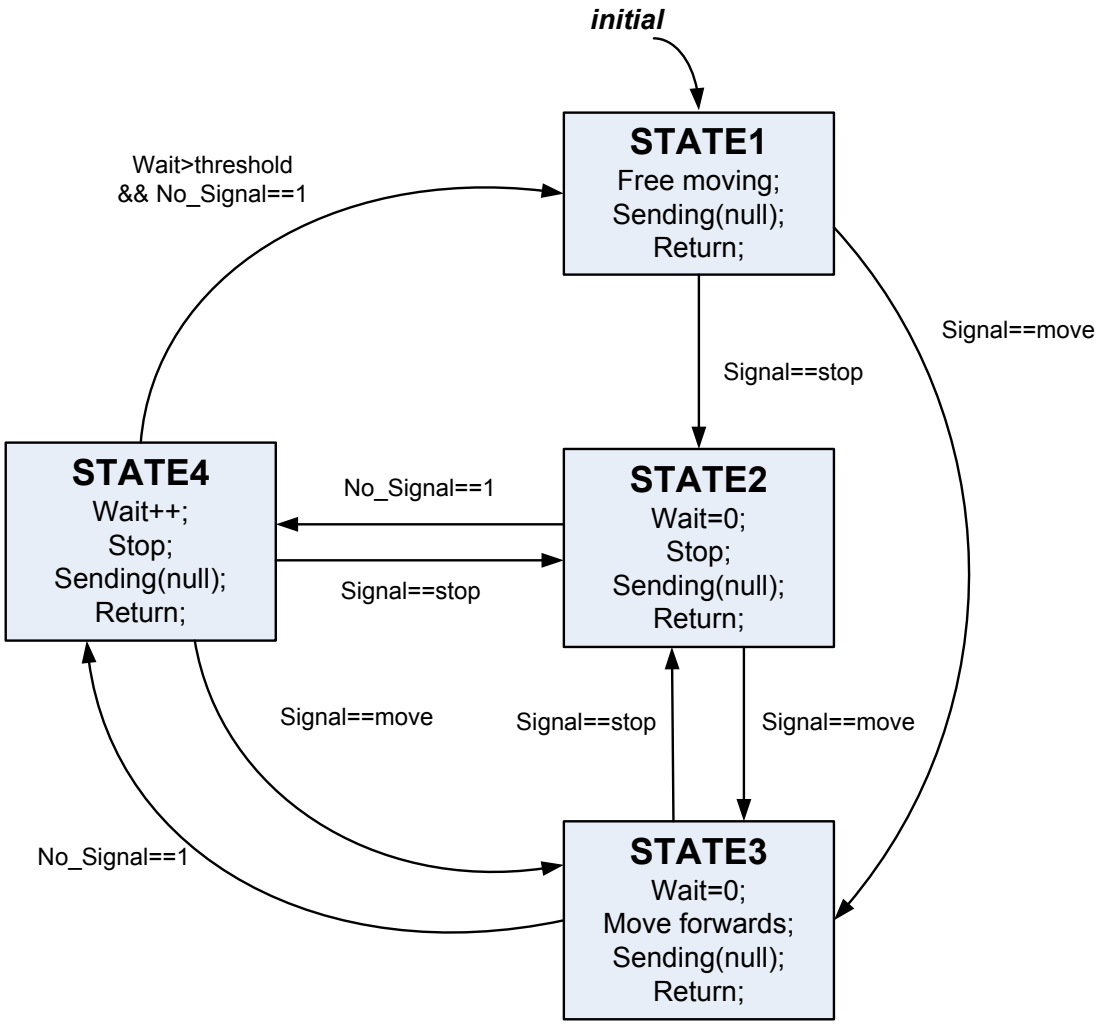


Figure 5.4: FSM of robot1 in formation-case one.

Algorithm 21 Decision making() for Robot1 in formation-case one

```
1: if Last_State == 1 then
2:    if Signal == stop then
3:       State ⇐ 2
4:    else if Signal == move then
5:       State ⇐ 3
6:    end if
7: else if Last_State == 2 then
8:    if No_Signal == 1 then
9:       State ⇐ 4
10:   else if Signal == move then
11:      State ⇐ 3
12:   end if
13: else if Last_State == 3 then
14:   if No_signal == 1 then
15:      State ⇐ 4
16:   else if Signal == stop then
17:      State ⇐ 2
18:   end if
19: else if Last_State == 4 then
20:   if Signal == stop then
21:      State ⇐ 2
22:   else if Signal == move then
23:      State ⇐ 3
24:   else if No_signal == 1&&Wait > threshold then
25:      State ⇐ 1
26:   end if
27: else
28:   State ⇐ 1
29: end if
30: Last_State ⇐ State
```

Algorithm 22 Executing() for Robot1 in formation-case one

```
1: if State == 1 then
2:    Free moving()
3: else if State == 2 then
4:    Wait ⇐ 0
5:    Stop()
6: else if State == 3 then
7:    Wait ⇐ 0
8:    Move forwards()
9: else if State == 4 then
10:   Wait++
11:   Stop()
12: end if
13: Signal sending(null)
14: Return
```

Results of simulation

The Figure 5.5 shows the moving traces of two robots. The ◯ and × represent the initial positions of robot0 and robot1, respectively. It can be clearly seen that two robots are keeping formation and moving forwards after 100*s*.

The Figure 5.6 shows the distance and relative direction between two robots. The Figure 5.6(b) shows the number of channel from which the communication signal is received. If the receiving channel number for the robot0 is 5 and for the robot is 2, it implicates the relative direction is correct. The Figure 5.6(a) shows the distance between two robots. It can be see that at the time around 55*s*, both the distance and direction are correct. But the distance is still increasing after 55*s* until they lose the formation. It results from the reason that the robot0 only ensures the relative direction is right during the motion. The proof for judging the right direction is that the IR receiver on the robot0's sector5 receives the signal from IR transmitter on the robot1's sector 2. If the two IR couples do not point to each other in a straight line, the moving directions for two robots are a bit different. Therefore, two robots will go farther or closer to each other. Finally, they will lose formation and then they need to perform formation again. After the 100*s*, two robots keep formation precisely since two IR couples straightly point to each other.

The two robots also have the abilities to avoid the obstacles when they are moving with formation. The Figure 5.7 is a series of video clips to show the behaviors when they meet the wall two times. Every time, the robot1 detects the wall and tries to avoid it. The robot0 always tries to keep the formation with robot1 except the case when there is any

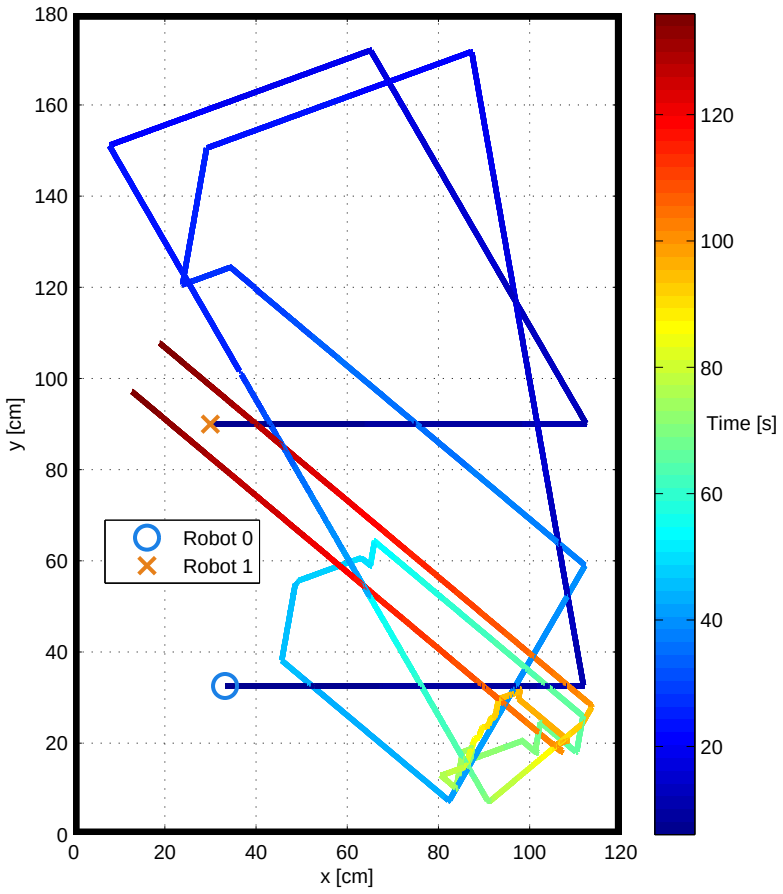
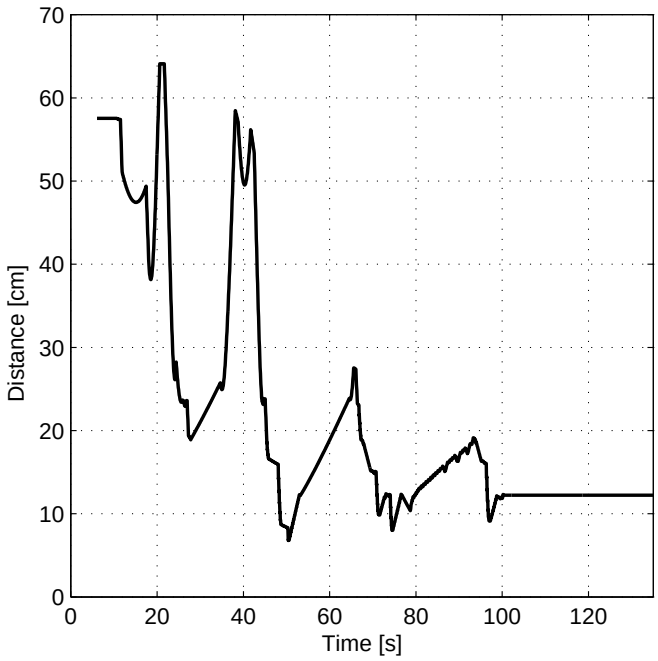
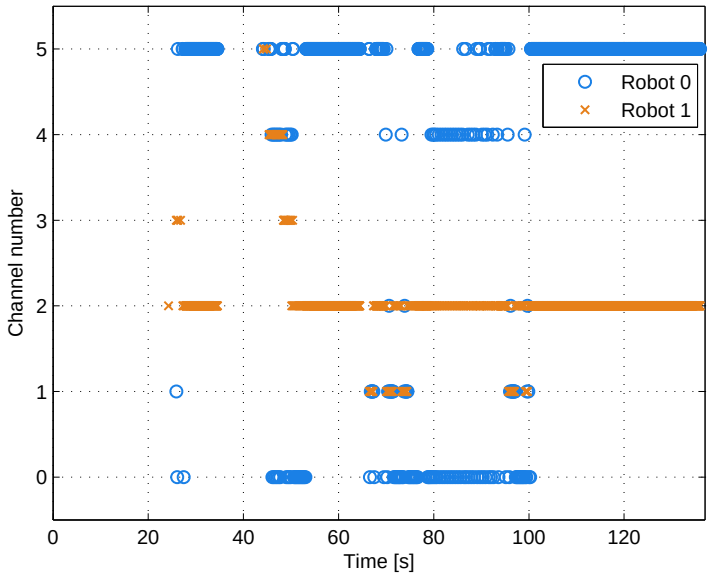


Figure 5.5: Simulation result of two robots' moving traces in formation-case one.

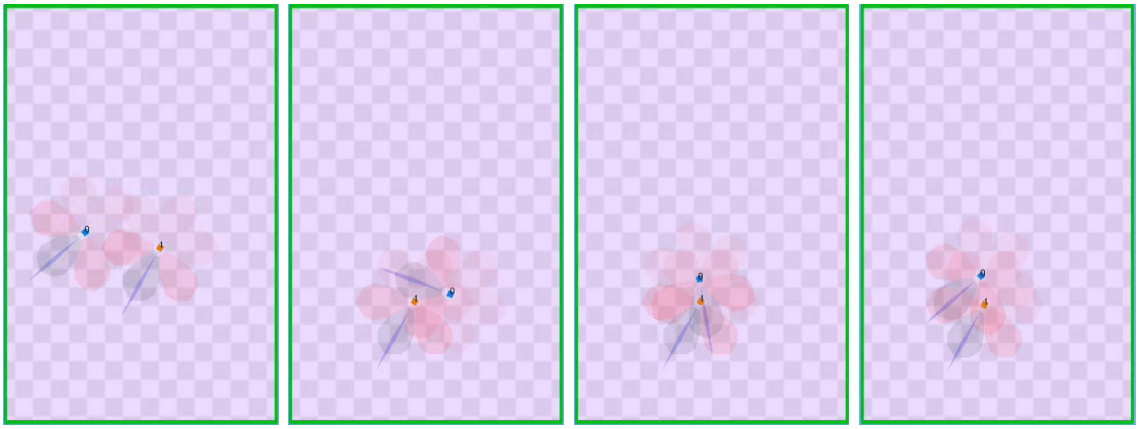


(a) The distance between two robots.



(b) The signal received channel numbers of two robots.

Figure 5.6: Relative direction and distance from simulation in formation-case one.

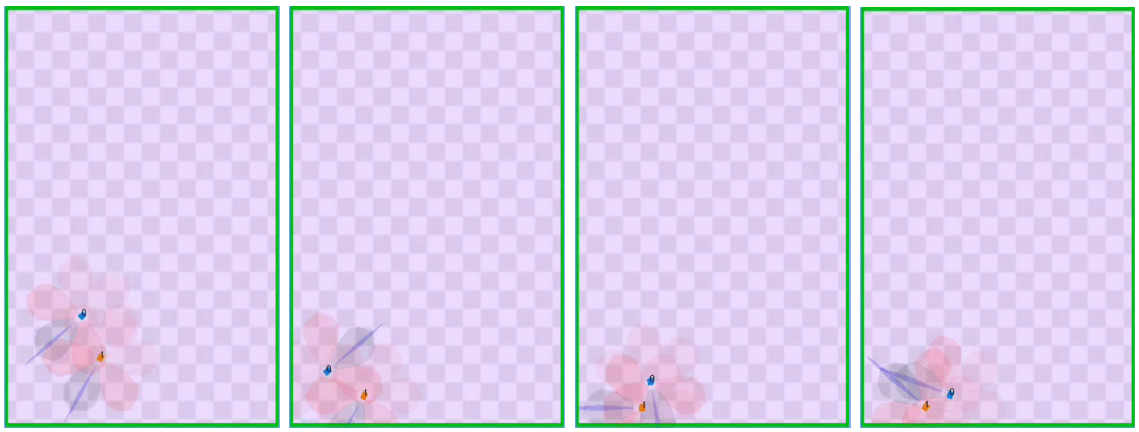


(a) 36.35s. Two robots are moving freely.

(b) 47.41s. R0 asks R1 to stop and is adjusting the direction $00to12$.

(c) 51.86s. R0 is adjusting the distance.

(d) 53.62s. R0 is adjusting the direction $05to12$.



(e) 60.88s. Finish formation and move forwards with deflected directions.

(f) 67.21s. R1 meets the obstacle.

(g) 72.66s. R1 is avoiding the obstacle and R0 performs formation.

(h) 76.95s. R1 has avoided the obstacle and R0 performs formation.

(i) 83.49s. Finish formation and move forwards.

(j) 97.19s. R1 meets the obstacle again and R0 performs formation.

(k) 101.15s. Finish formation and move forwards.

(l) 106.26s. Keep the formation and move forwards with the same direction.

Figure 5.7: The video clips from simulation in formation-case one. The number represents the time in simulation. R0 and R1 represent the robot0 and the robot1, respectively.

obstacle suffered by itself. It can be seen that the two robots can succeed to avoid the wall and quickly recover the formation.

Results of experiments

The formation algorithm is validated on experiments without a scenario. The Figure 6.4 shows the sequential video clips from the test. It can be seen that the robot0 needs to take much longer time than the time spent in simulation to finish the process of adjusting direction $00to12$, which is shown from Figure 5.8(c) to 5.8(h). Finally, they finish formation can move forwards together.

One problem can not be avoided is that they will lose formation since only relative direction is ensured when they moving together, which also exists in the simulation result. However, two robots can not perform so well as the behaviors in the simulation. They will lose formation when losing communication. Also, the actuation system can not carry out the behavior as expected. Therefore, two robots need to generate formation much more frequently than in the simulation.

5.3 Formation – case two

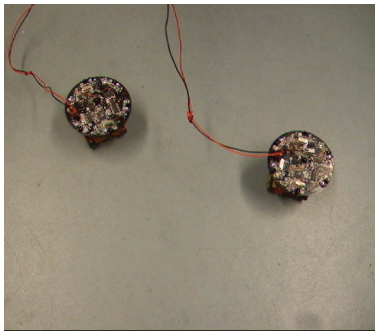
In the second case of formation, when the robot0 finishes formation generation, it will keep both relative direction and distance during they are moving. When either direction or distance is not correct, the robot0 has to perform formation again. This case will be discussed briefly in this section.

5.3.1 Algorithms

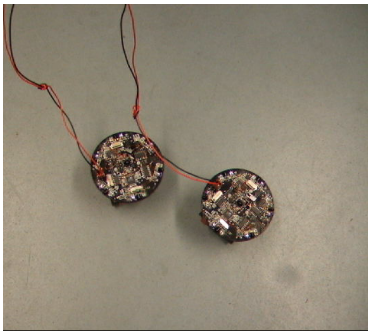
The algorithms for two robot are similar to the algorithms in case one, which will be described briefly in the following.

The main loop are the same as in case one, which has been shown in Figure 5.2. The behaviors for the robot1 are also the same as before, which has been written in Algorithm 20, 21 and 22.

For the robot0, it has to perform much heavier behaviors than in the case one. The signal analyzing is the same as the Algorithm 17. The Figure 5.9 shows the FSM in this case. Compared with the FSM in Figure 5.3, the number of the state and the actions in each state

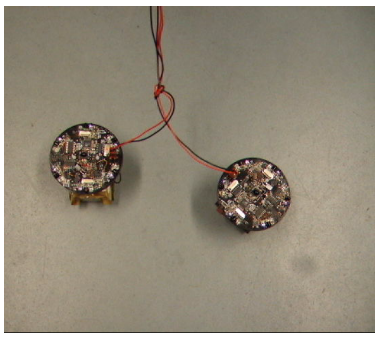


(a) Two robots are moving freely .

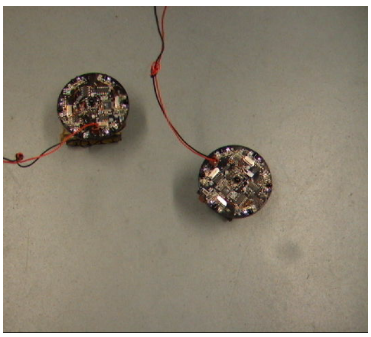


(b) R0 meets R1 and asks R1 to stop.

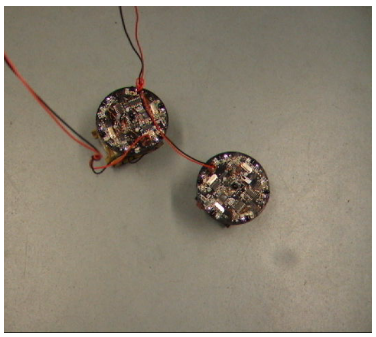
(c) R0 is adjusting the direction $00to12$.



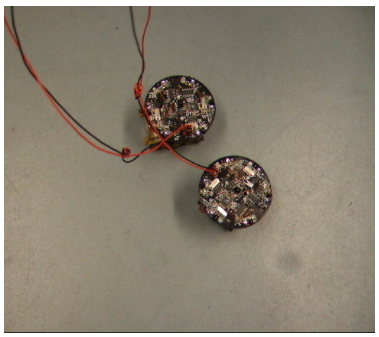
(d) R0 is adjusting the direction $00to12$.



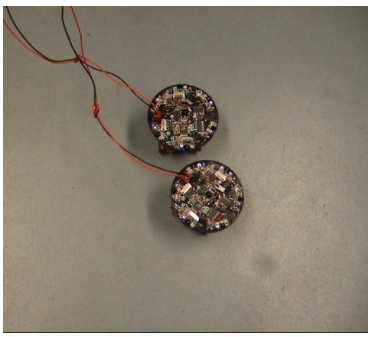
(e) R0 is adjusting the direction $00to12$.



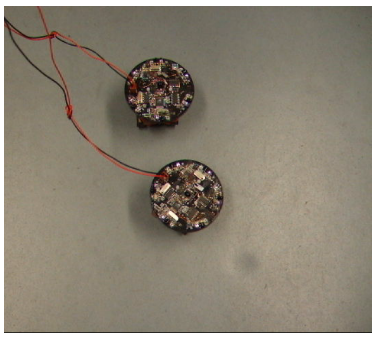
(f) R0 is adjusting the direction $00to12$.



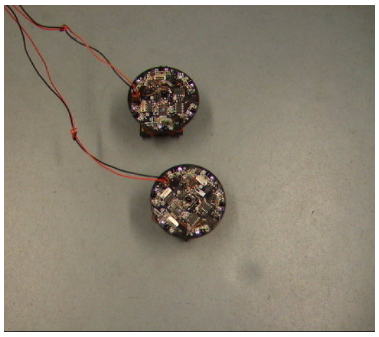
(g) R0 is adjusting the direction $00to12$.



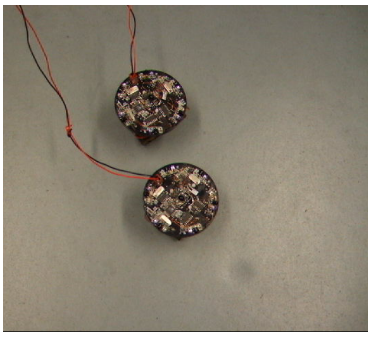
(h) R0 finishes adjusting the direction $00to12$.



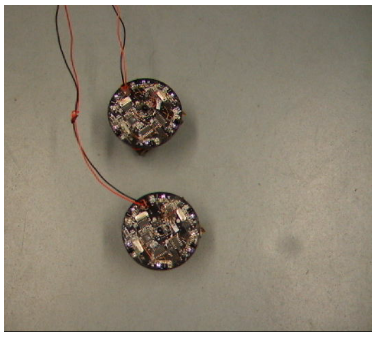
(i) R0 is adjusting the distance.



(j) R0 finishes adjusting the distance.



(k) R0 is adjusting the direction $05to12$.



(l) Keep the formation and move forwards.

Figure 5.8: The video clips from the experiment in formation-case one.

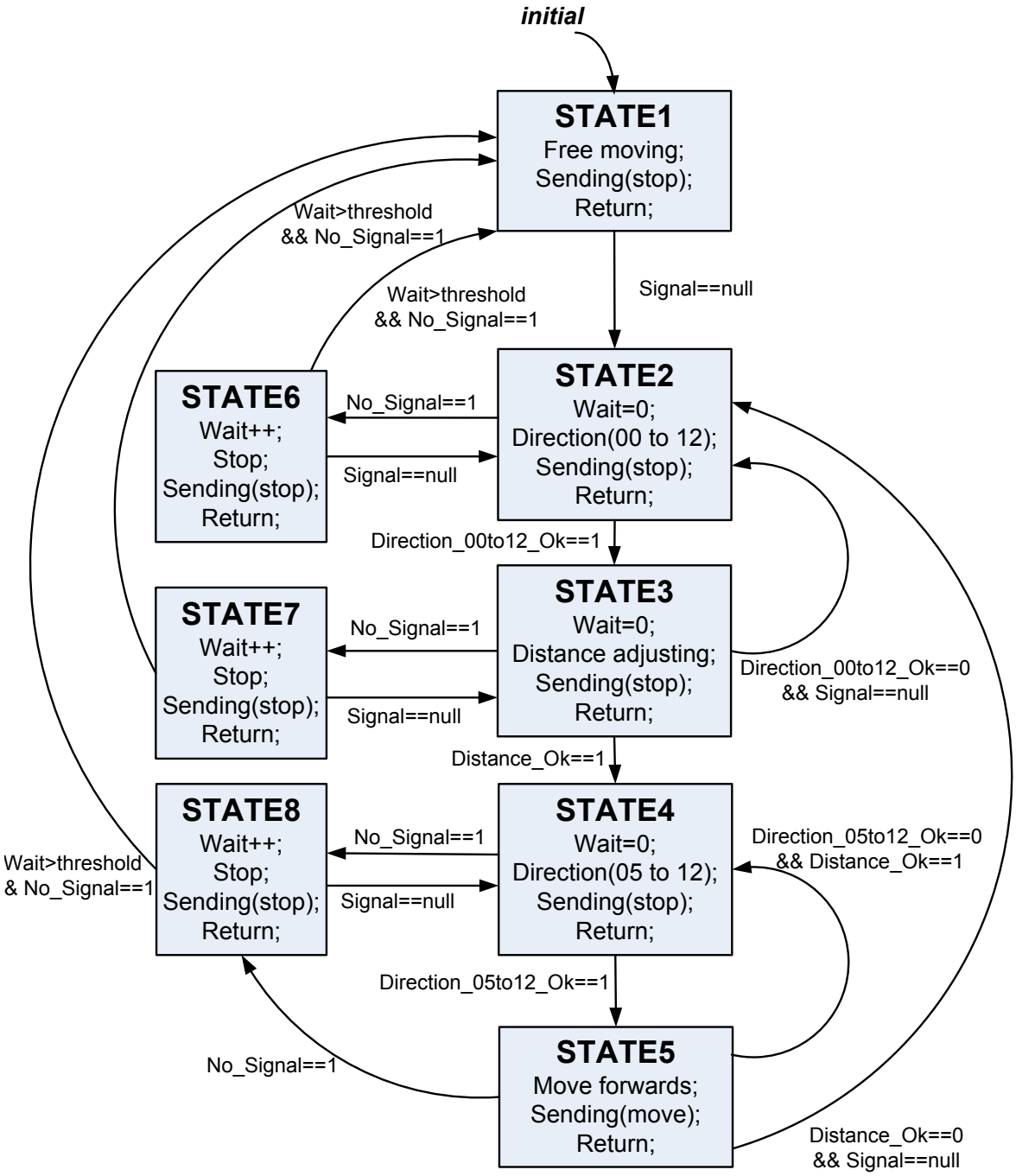


Figure 5.9: FSM of robot0 in formation-case two.

are the same, while transitions are a bit different. One assistant variable, *Counter*, is not needed any more since the robot0 will go to the state to adjust the distance immediately when the distance is not correct.

5.3.2 Testing results

In reality, the robot can not response so much quickly and accurately as in simulation. Therefore, the algorithms in this case are only tested in simulation.

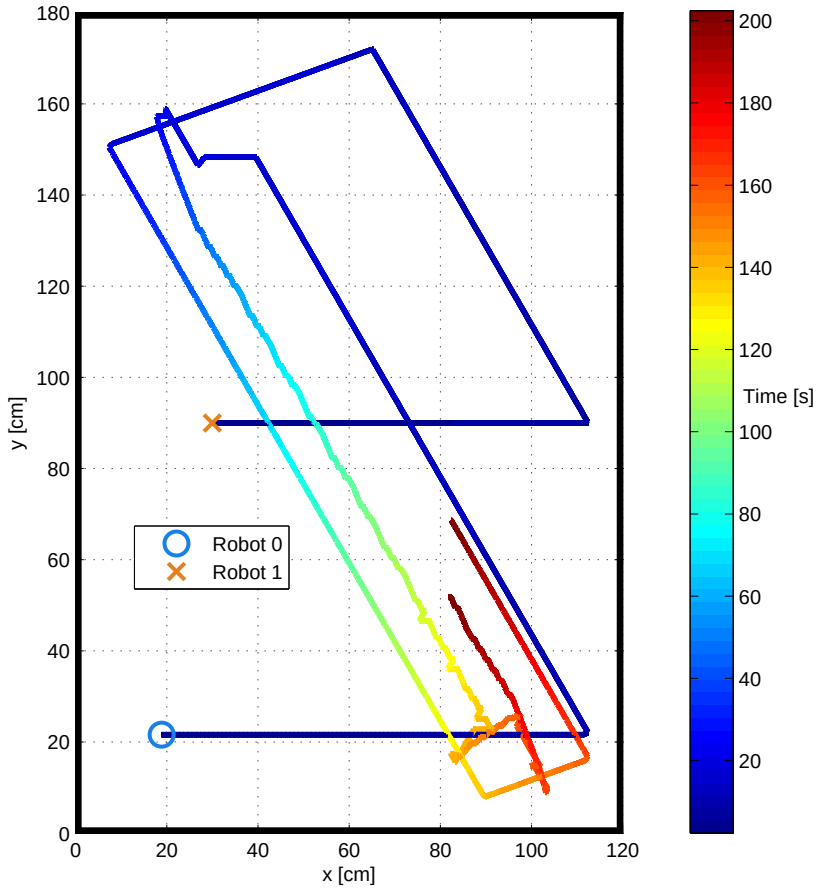
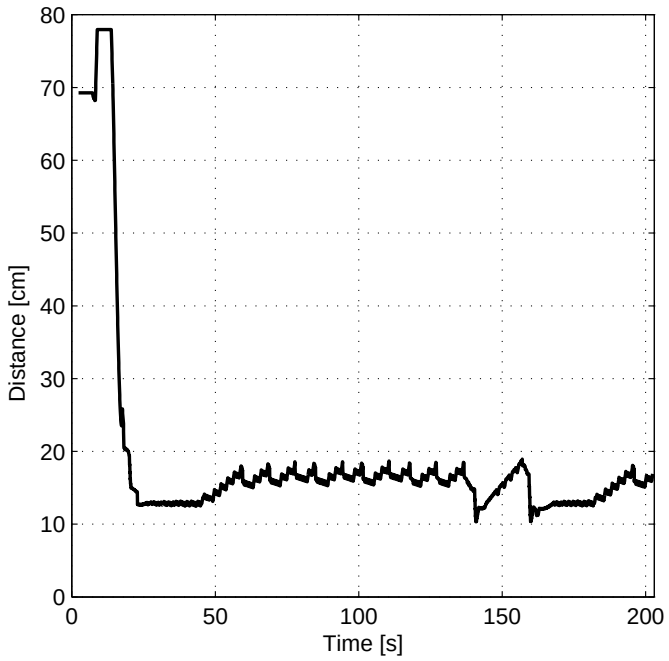


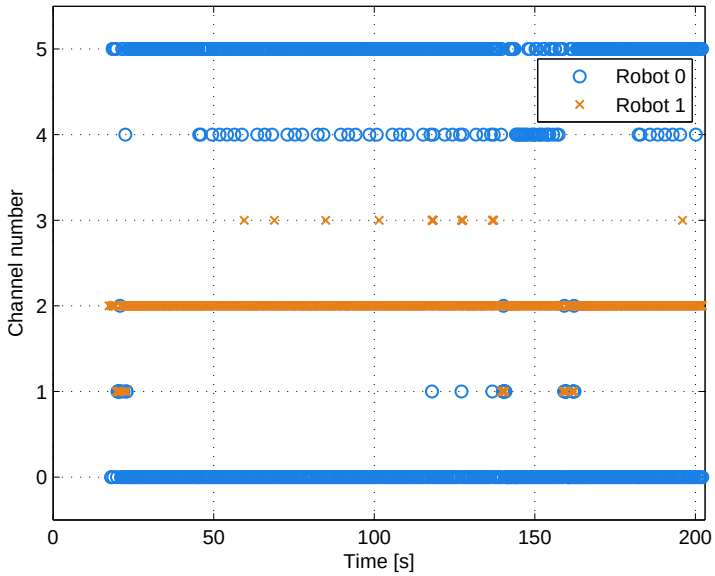
Figure 5.10: Simulation result of two robots' moving traces in formation-case two.

The Figure 5.10 shows the moving traces of two robots. It can bee seen that two robots can avoid the obstacle and move forwards with formation. The robot0's trace is not so smooth as the trace of robot1. It dues to the fact that the robot0 adjusts the relative direction and distance frequently in order to keep the formation.

Figure 5.11(a) shows the distance between two robots. It can be seen that when the distance begins to increase, it will decrease again by the effort from the robot0. The



(a) The distance between two robots.



(b) The signal received channel numbers of two robots.

Figure 5.11: Relative direction and distance from simulation in formation-case two.

Figure 5.11(b) shows the two robots' channel numbers which receive signal. It shows that the channel number for the robot0 switches between 0 and 5 frequently in order for adjusting the distance and direction.

From the simulation result, it proves that the algorithms are good at keeping formation. While it expenses much more motions of robot0. Therefore, maneuverability and flexibility of the robots are two necessary conditions to realize the formation by using this approach.

5.4 Summary and discussion

In this chapter, the algorithms of formation by two robots are developed. There are two cases are discussed separately.

In Section 5.1, the formation shape and the expected procedures to realize this formation are described firstly. Then, two algorithms for realizations of direction and distance adjustments are developed, which are two important processes in the formation.

In Section 5.2, the case that only relative direction is kept when two robots move together is discussed. The separated algorithms for the robot0 and the robot1 are developed with respect to their different roles. The algorithms are validated in both simulation and experiments. The two robots can keep the formation as well as obstacle avoidance. However, they will lose formation if their moving directions are not exactly the same.

In Section 5.3.1, the case that both relative direction and distance are kept when two robot are moving with formation. The algorithm for the robot1 is the same as in the case one. While the algorithm for the robot0 is newly developed. The algorithms are only validated in simulation since there is a high requirement on the robot's maneuverability and the current robots can not achieve. The simulation result shows that two robots can be more robust to keep the formation when moving. Even though there is an obstacle, two robots can succeed to avoid it and recover formation again.

For the swarm robots, it suggests to use one algorithm for all the robot members. While on the implementation of this project, separated algorithms for each robot are developed. But it does not disobey the suggested rule. Actually, the separated algorithms are developed for different roles rather than for different robots. Because only two robots are implemented in this project, one role has already been assigned to one robot in the beginning. For a large group of swarm robots, it prefers to integrate the algorithms of different roles' behaviors into one algorithm and fixed the same algorithm onto all the robots since they

are not so distinguishable as only two robots. When robots doing communication with others, they can decide which roles they should play. Thus, every robot will perform one role's behavior based on the same integrated algorithm. One advantage to distribute roles beforehand in this project is to save flash memories of MCU, which is quite important for the current used MCU.

Chapter 6

Collective Behavior

Another swarm robots behaviors implemented in this project is to let the robots perform a collective task. In this chapter, the collective behavior is implemented by two robots. Each robot plays a different role and has different responsibility with respect to a specific collective task. The collective task will be planned and the corresponding algorithms will be developed for each robot. The algorithms are validated in simulation or experiments.

6.1 Task planning

For different collective tasks, the robots have to be assigned with different behaviors. Therefore, a specific collective task has to be planned firstly. Then according to that planned task, the different duties can be distributed on different robots.

One complete collective task is planned and written sequentially in the following:

1. The robots are searching the food
2. If food is found by one robot, this robot will ask the other robots to help carrying the food to their home.
3. Some robots are carrying the food to home and some robots are guarding them.

In the plan of this task, each job is supposed to be conducted by a group of robots. Different common responsibilities will be assigned to different groups, involving searching food, carrying food, finding the way to home and security guarding. However, only two robots have been built until now and these two robots have to take on all the jobs above.

Therefore, in order to realize collective behaviors by two robots. The planned collective task has to be separated into two subtasks. For each subtask, different robots have the different responsibilities.

In the subtask one, one robot is responsible for searching the food and then carrying the food and the other robot will help to carry the food.

In the subtask two, one robot is carrying the food and finding a way to go home and the other robot has to guard the first robot.

These two subtasks will be discussed in the following sections and the algorithms will be developed.

6.2 Subtask one

In this subtask, two robots need to find the food and carry the food. The object with red light will be treated as the food that the robots want. Due to the structure of the robot, there is neither a gripper to hold the food nor a place to keep the food. Therefore, the food can be moved only if the robots push it. On the implementation, the expected procedures to perform this subtask are written in the following.

1. The robot1 is searching an object with red light.
2. If the robot1 finds the object, it will stop and ask for help from another robot.
3. If the robot0 gets the help signal, it will perform formation with the robot1. When the formation is finished, the robot0 will ask the robot1 to move the object together.
4. When the robot1 receives the signal from the robot0 and knows the robot0 is ready to move, it will move the object.

6.2.1 Algorithms

The main loop for both robots is similar to the case of formation, which is illustrated in Figure 6.1. However, one more judgement is added depending on a variable, *Last_Food*. This variable is used to distinguish food and obstacle. If the robot believes there is a food in front of itself, it will directly perform the expected behavior according to that food. Otherwise, the robot has to judge whether there is an obstacle or not.

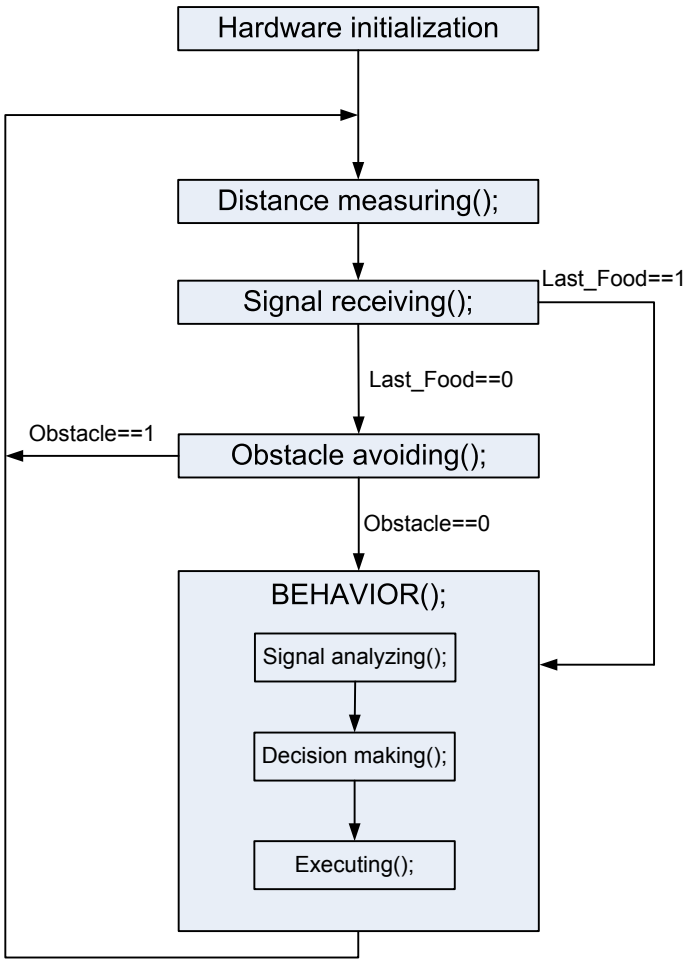


Figure 6.1: Flow chart for both robots in collective subtask one.

The specific behaviors for robot0 and robot1 are different, which will be discussed respectively.

Robot0

The Algorithm 23 shows the useful information extracted from the communication signal. Three variable are important for the decision making, which are *No_signal*, *Signal* and *Direction_05to11_Ok*. The variable, *Last_Food* is initialized to be 0, which will be changed if there is food.

Algorithm 23 Signal analyzing() for Robot0 in collective subtask one

```
1: Last_Food ⇐ 0
2: No_signal ⇐ 0
3: Direction_05to11_Ok ⇐ 0
4: if There_is_a_Neighbor_Robot == 1 then
5:    Signal ⇐ Status_of_Neighbor_Robot
6:    if Neighbor_Robot_Channel == 1&&My_Received_Channel == 5 then
7:       Direction_05to11_Ok ⇐ 1
8:    end if
9: else
10:   No_Signal ⇐ 1
11: end if
```

The Figure 6.2 shows the FSM about the transitions between states and the actions in each state. If a help signal is received, the robot0 will perform the formation with the robot1. The relative direction $05to11$ is only ensured for this formation. With this formation, the two robots almost stand shoulder by shoulder and face to the food. If the formation is finished, the robot0 will send ready signal to the robot1. Meanwhile, the Robot0 believes there is food in front and tries to move the food. If there is no help signal received accidently, the robot0 will wait until it believes that there is no help needed from others. The corresponding processes are written in Algorithm 24 for decision making and 25 for behavior executing.

Robot1

In the Algorithm 26, the robot1 actives the color sensor and analyzes the color. If detected color is red, it represents food in front. Furthermore, the robot1 will judge the food is close to itself if the output frequency from the red filter is higher than a threshold.

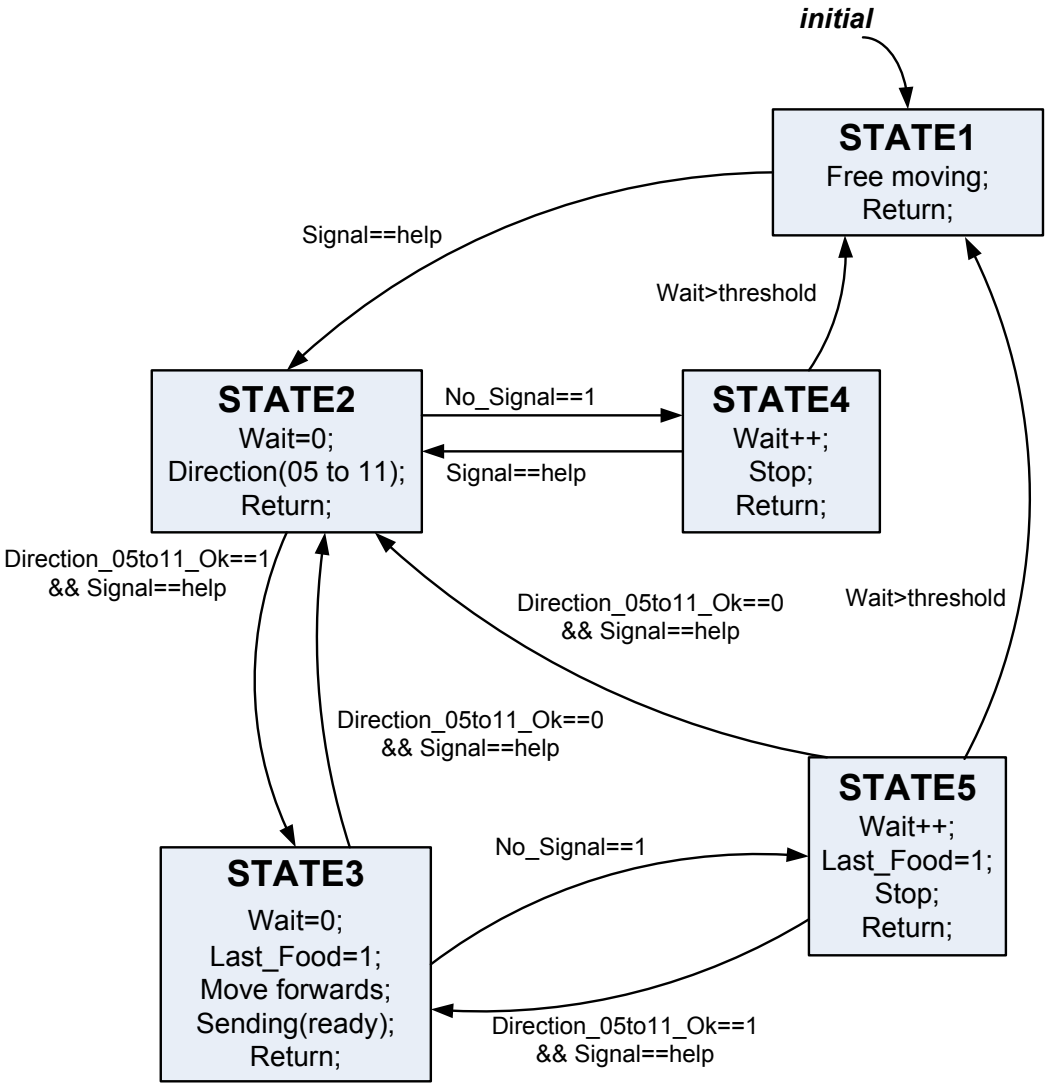


Figure 6.2: FSM of robot0 in collective subtask one.

Algorithm 24 Decision making() for Robot0 in collective subtask one

```
1: if Last_State == 1 then
2:     if Signal == help then
3:         State ⇐ 2
4:     end if
5: else if Last_State == 2 then
6:     if No_signal == 1 then
7:         State ⇐ 4
8:     else if Direction_05to11_Ok == 1&&Signal == help then
9:         State ⇐ 3
10:    end if
11: else if Last_State == 3 then
12:    if No_Signal == 1 then
13:        State ⇐ 5
14:    else if Direction_05to11_Ok == 0&&Signal == help then
15:        State ⇐ 2
16:    end if
17: else if Last_State == 4 then
18:    if Signal == help then
19:        State ⇐ 2
20:    else if Wait > threshold then
21:        State ⇐ 1
22:    end if
23: else if Last_State == 5 then
24:    if Direction_05to11_Ok == 0&&Signal == help then
25:        State ⇐ 2
26:    else if Direction_05to11_Ok == 1&&Signal == help then
27:        State ⇐ 3
28:    else if Wait > threshold then
29:        State ⇐ 1
30:    end if
31: else
32:    State ⇐ 1
33: end if
34: Last_State ⇐ State
```

Algorithm 25 Executing() for Robot0 in collective subtask one

```
1: if State == 1 then
2:     Free moving()
3: else if State == 2 then
4:     Wait ⇐ 0
5:     Direction adjusting(00to12)
6: else if State == 3 then
7:     Wait ⇐ 0
8:     Last_Food ⇐ 1
9:     Move forwards()
10:    Signal sending(ready)
11: else if State == 5 then
12:    Wait++
13:    Stop()
14: else if State == 5 then
15:    Wait++
16:    Last_Food ⇐ 1
17:    Stop()
18: end if
19: Return
```

Algorithm 26 Signal analyzing() for Robot1 in collective subtask one

```
1: Color deciding()
2: Food ⇐ 0
3: Food_Near ⇐ 0
4: if Color == red then
5:     Food ⇐ 1
6:     if Red_Filter_Frequency > threshold_of_frequecny then
7:         Food_Near ⇐ 1
8:     end if
9: end if
10: Last_Food ⇐ Food
11: No_signal ⇐ 0
12: if There_is_a_Neighbor_Robot == 1 then
13:     Signal ⇐ Status_of_Neighbor_Robot
14: else
15:     No_Signal ⇐ 1
16: end if
```

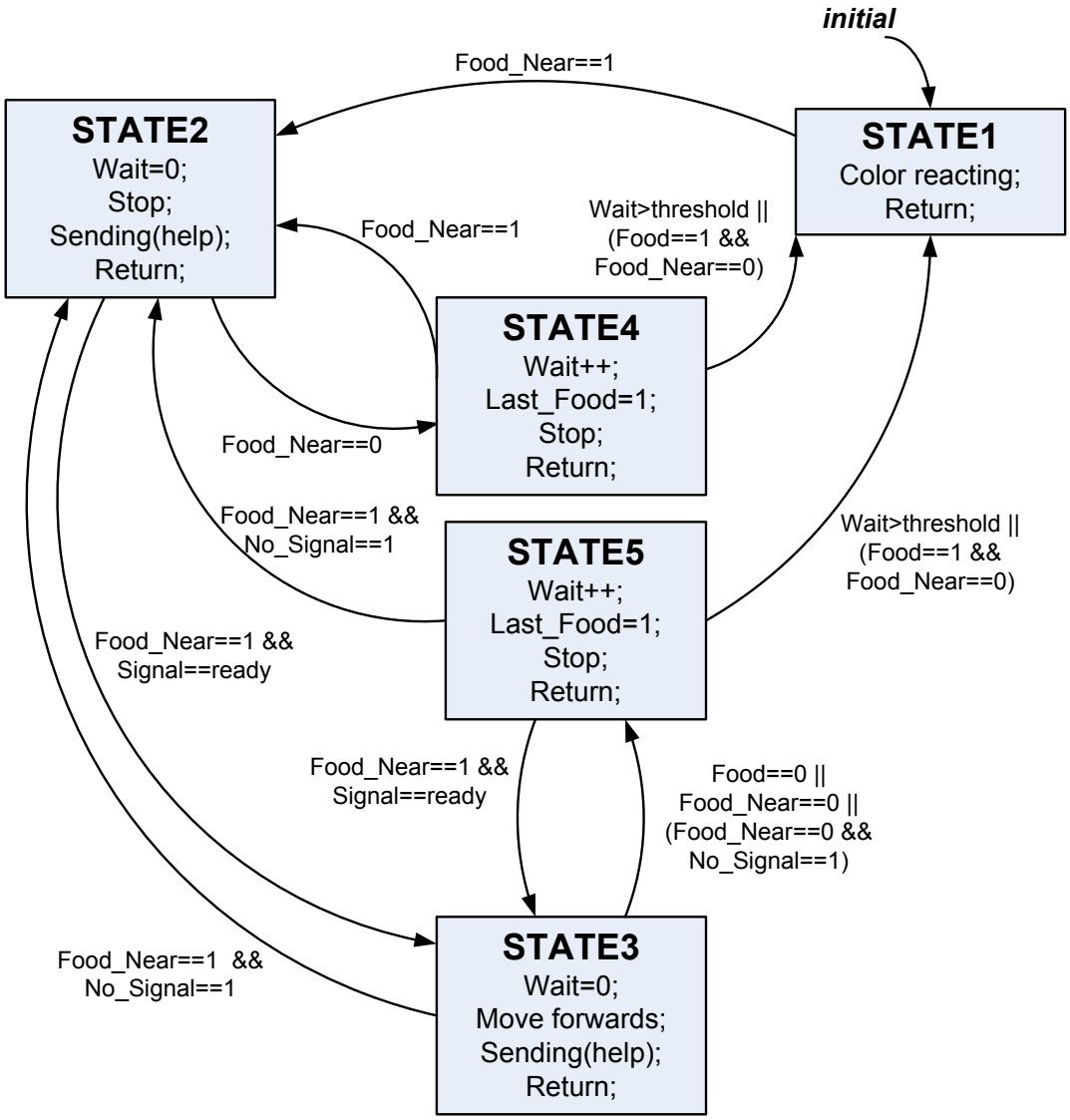


Figure 6.3: FSM of robot1 in collective subtask one.

The Figure 6.3 shows the FSM for the robot1. If the object with red light is detected by the robot1, it will approach it. Then, the robot1 will stop close to the object and send out signal for help. Finally, the robot1 will move the objects if there is a ready signal received from another robot. The state 4 and 5 are used for waiting in the cases of accidently changing of sensor's readings or communication signal. The corresponding processes are written in Algorithm 27 for decision making and 28 for behavior executing.

6.2.2 Testing results

The algorithms are validated in the experiments. The result is shown in Figure 6.4. It can be seen that the robot1 gets the help from the robot0 and they cooperate to move the object with red light.

6.3 Subtask two

In the second subtask, one robot is assumed to be carrying the food and tries to find a way to go home. The responsibility of the other robot is to guard the first robot. The latter robot plays a role as a security person, who should always stay around the first robot and ensure no accidence for the food. The home will be imitated by a light source. The bulb used before as a light source is not possible in this case, because the communication can not work with the light from the bulb. Therefore, a white light from a LED array will be treated as the home.

On the implementation, the expected procedures to perform this subtask are written in the following:

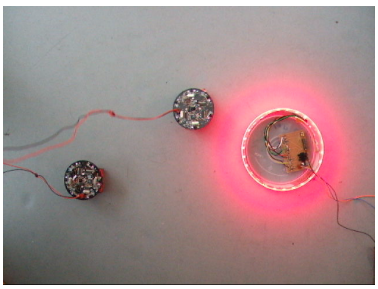
1. The robot1 is searching a way to home and asking for security guard.
2. The robot0 gets the signal and stays around the robot1.
3. The robot1 goes home and the robot0 guards it.
4. When robot1 arrives home, the robto0 finishes its task and moves freely to see if there are other robots need to be guarded.

6.3.1 Algorithms

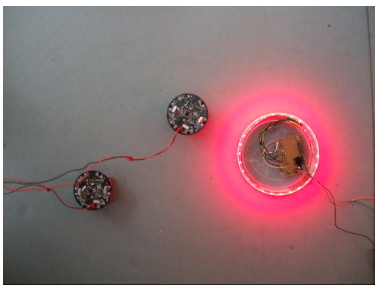
The algorithms for the robot0 and robot1 are developed respectively in the following.

Algorithm 27 Decision making() for Robot1 in collective subtask one

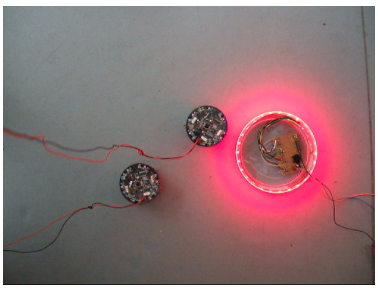
```
1: if Last_State == 1 then
2:    if Food_Near == 1 then
3:       State ⇐ 2
4:    end if
5: else if Last_State == 2 then
6:    if Food_Near == 0 then
7:       State ⇐ 4
8:    else if Food_Near == 1&&Signal == ready then
9:       State ⇐ 3
10:   end if
11: else if Last_State == 3 then
12:   if Food_Near == 1&&No_Signal == 1 then
13:      State ⇐ 2
14:   else if Food == 0||Food_Near == 0||No_Signal == 1 then
15:      State ⇐ 5
16:   end if
17: else if Last_State == 4 then
18:   if Food_Near == 1 then
19:      State ⇐ 2
20:   else if Wait > threshold||(Food == 1&&Food_Near == 0) then
21:      State ⇐ 1
22:   end if
23: else if Last_State == 5 then
24:   if Food_Near == 1&&No_Signal == 1 then
25:      State ⇐ 2
26:   else if Food_Near == 1&&Signal == ready then
27:      State ⇐ 3
28:   else if Wait > threshold||(Food == 1&&Food_Near == 0) then
29:      State ⇐ 1
30:   end if
31: else
32:   State ⇐ 1
33: end if
34: Last_State ⇐ State
```



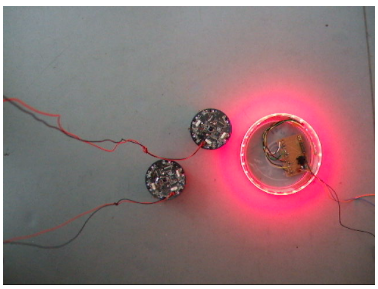
(a) R1 is searching the object with red light.



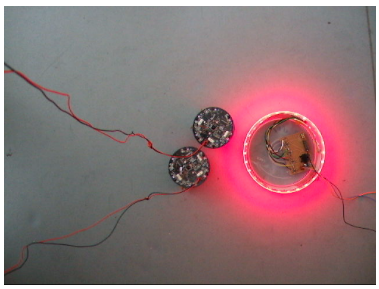
(b) R1 detects the object.



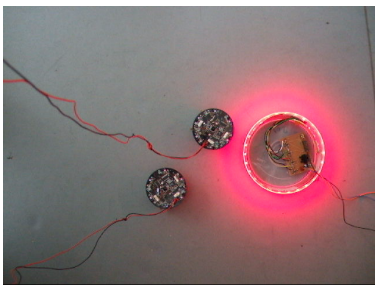
(c) R1 approaches and stops in front of the object, then asks for help.



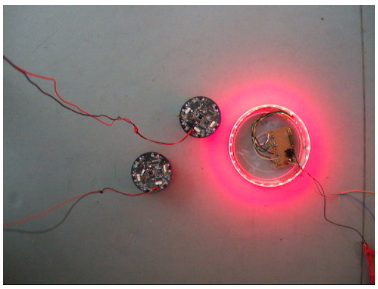
(d) R0 gets the help signal and performs formation.



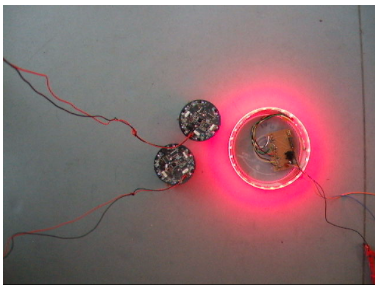
(e) R0 is too close to R1.



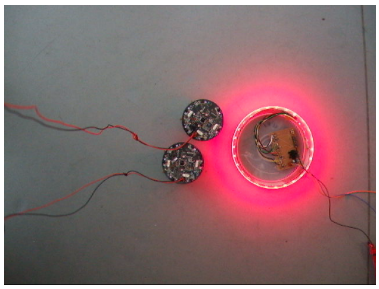
(f) R0 moves backwards to avoid R1 and performs formation again.



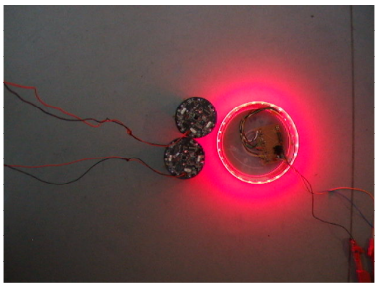
(g) R0 performs formation.



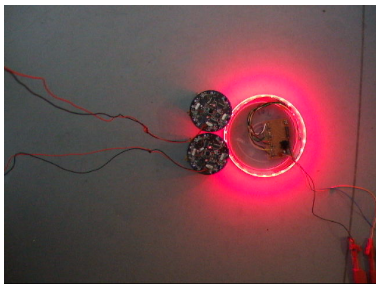
(h) R0 performs formation.



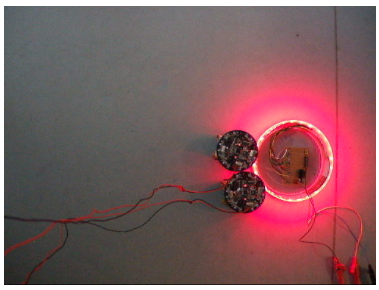
(i) R0 performs formation.



(j) Finish the formation and begin to move the object.



(k) Move the object.



(l) Move the object.

Figure 6.4: The video clips from an experiment in collective subtask one. R0 and R1 represent the robot0 and the robot1, respectively.

Algorithm 28 Executing() for Robot1 in collective subtask one

```
1: if State == 1 then
2:     color reacting()
3: else if State == 2 then
4:     Wait ⇐ 0
5:     Stop()
6:     Signal sending(help)
7: else if State == 3 then
8:     Wait ⇐ 0
9:     Move forwards()
10:    Signal sending(help)
11: else if State == (4||5) then
12:    Wait++
13:    Last_Food ⇐ 1
14:    Stop()
15: end if
16: Return
```

Robot0

The main loop for the robot0 is the same as the loop shown in Figure 5.2. The behaviors including the signal analyzing, decision making and executing, are all written inside one Algorithm 29. From the communication signal, the robot0 only needs to judge whether existing robots need to be guarded. If there is a request from another robot, the robot0 will move close to guard it. Otherwise, the robot0 moves freely. These behaviors have been shown in the Figure 6.5, which is the FSM to illustrate state transitions and actions.

The Algorithm 30 shows the approach to realize the security guarding. The robot0 has to make sure that the robot1 is always in front of itself. If the distance is too far, the robot0 will speed up to move close to the robot1. If the distance is too short, the robot0 will stop to avoid crash. Otherwise, the robot0 will keep normal speed to move with the robot1.

Robot1

The Figure 6.6 shows the flow chart of the main loop for robot1 in this subtask. The robot1 has to judge the home after signal receiving. If the robot1 believes it is on the way to home or the home has already been arrived, it will perform the specific behaviors directly. Otherwise, the robot1 will analyze about the obstacle existence firstly.

The detailed behavior including the signal analyzing, decision making and executing, are all written by the Algorithm 31. During signal analyzing, the robot1 will detect the

Algorithm 29 BEHAVIOR() for Robot0 in collective subtask two

```
1: {Signal analyzing()}
2: No_signal ⇐ 0
3: if There_is_a_Neighbor_Robot == 1 then
4:    Signal ⇐ Status_of_Neighbor_Robot
5: else
6:    No_Signal ⇐ 1
7: end if
8:
9: {Decision making()}
10: if Last_State == 1 then
11:    if Signal == guard then
12:       State ⇐ 2
13:    end if
14: else if Last_State == 2 then
15:    if No_Signal == 1 then
16:       State ⇐ 3
17:    end if
18: else if Last_State == 3 then
19:    if Signal == guard then
20:       State ⇐ 2
21:    else if Wait > threshold&&Nosignal == 1 then
22:       State ⇐ 1
23:    end if
24: else
25:    State ⇐ 1
26: end if
27: Last_State ⇐ State
28:
29: {Executing()}
30: if State == 1 then
31:    Free moving()
32: else if State == 2 then
33:    Security guarding()
34: else if State == 3 then
35:    Wait++
36:    stop()
37: end if
38: Return
```

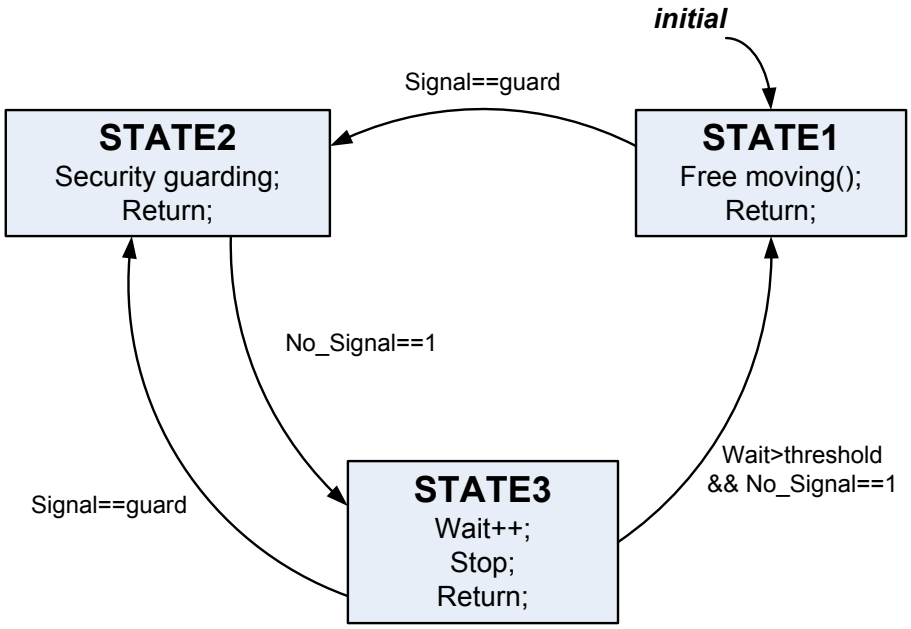


Figure 6.5: FSM of robot0 in collective subtask two.

Algorithm 30 Security guarding() for robot0 in collective subtask two

```
1: if My_Received_Channel == (1||2) then
2:     Turn right(60°)
3: else if My_Received_Channel == (3||4||5) then
4:     Turn left(60°)
5: else if My_Received_Channel == 0 then
6:     if Neighbor_Robot_Channel = (0||1||5) then
7:         Stop()
8:     else if Neighbor_Robot_Channel = (2||3||4) then
9:         if Distance_to_Neighbor_Robot > far_distance then
10:            Move forwards() with speed up
11:        else if Distance_to_Neighbor_Robot > near_distance then
12:            Stop()
13:        else
14:            Move forwards()
15:        end if
16:    end if
17: end if
```

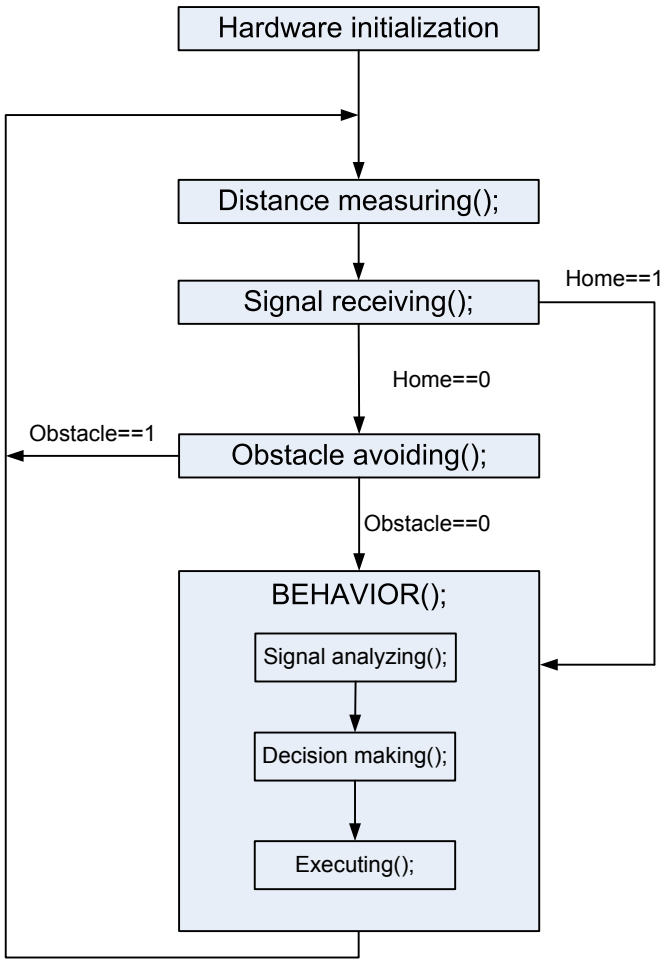


Figure 6.6: Flow chart for the robot1 in collective subtask two.

ambient light firstly and then judge that whether existing a home. If a home is existed, it will further decide whether the home has been arrived by analyzing the irradiance of the light. The more irradiant, the closer to home.

Algorithm 31 BEHAVIOR() for Robot1 in collective subtask two

```
1: {Signal analyzing()}
2: Ambient light detecting()
3: Home ⇐ 0
4: Home_Arrived ⇐ 0
5: if Max_Lux > Min_threshold_of_Lux then
6:     Home ⇐ 1
7: end if
8: if Max_Lux > Max_threshold_of_Lux then
9:     Home_Arrived ⇐ 1
10: end if
11:
12: {Decision making()}
13: if Last_State == 1 then
14:     if Home_Arrived == 1 then
15:         State ⇐ 2
16:     end if
17: else if Last_State == 2 then
18:     if Home_Arrived == 0 then
19:         State ⇐ 1
20:     end if
21: else
22:     State ⇐ 1
23: end if
24: Last_State ⇐ State
25:
26: {Executing()}
27: if State == 1 then
28:     Light source approaching()
29:     Signal sending(guard)
30: else if State == 2 then
31:     Stop()
32: end if
33: Return
```

The decision making and actions are indicated in Figure 6.7. The robot1 will keep approaching the light source and sending the request for guard. If the home is arrived, it will stop and have no request any more.

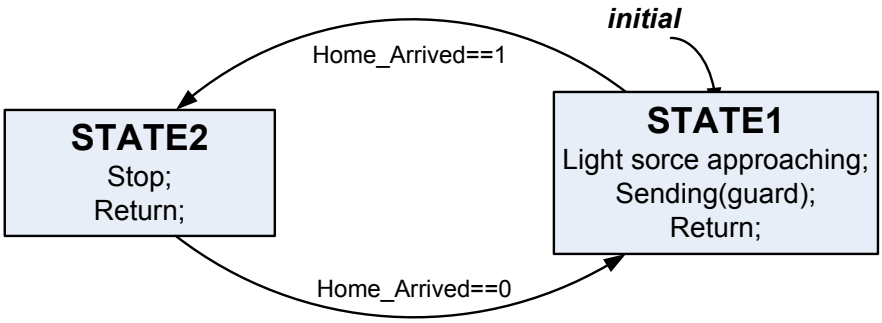


Figure 6.7: FSM of robot1 in collective subtask two.

6.3.2 Testing results

The Algorithm 30 has been validated in simulation. Because there is no simulated light source, the robot1 is assumed to be moving freely. The Figure 6.8 shows the moving traces of two robots. The smooth curve represents the behavior of the robot1. While the curve for the robot0 is rough since its duty is to guard the robot1 and its motion is mainly decided by the robot1. It can been seen that after the robot0 meets the robot1, the robot0 can succeed to keep staying around the robot1. The Figure 6.9 indicates the distance between two robots. It can been clearly seen that the distance can be always kept lower than 30cm after 35s.

An experiment has been done to validate the complete algorithms and the result is shown in Figure 6.10. It proves that the robot0 can succeed to guard the robot1 until the robot1 arrives home.

6.4 Summary

In this chapter, the collective behavior is implemented by two robots. The algorithms are developed and validated in simulation and experiments.

In Section 6.1, the complete collective task is planned and separated into two subtasks. For each subtask, different robots have different responsibilities.

In Section 6.2, the subtask one is discussed. One robot needs to find the object with red light, which is treated as food. Then this robot will ask for help from the other robot. The object will be moved with the contributions from two robots. Separated algorithms are developed with respect to each robot's duty. Also, the algorithms have been proved in the real experiment.

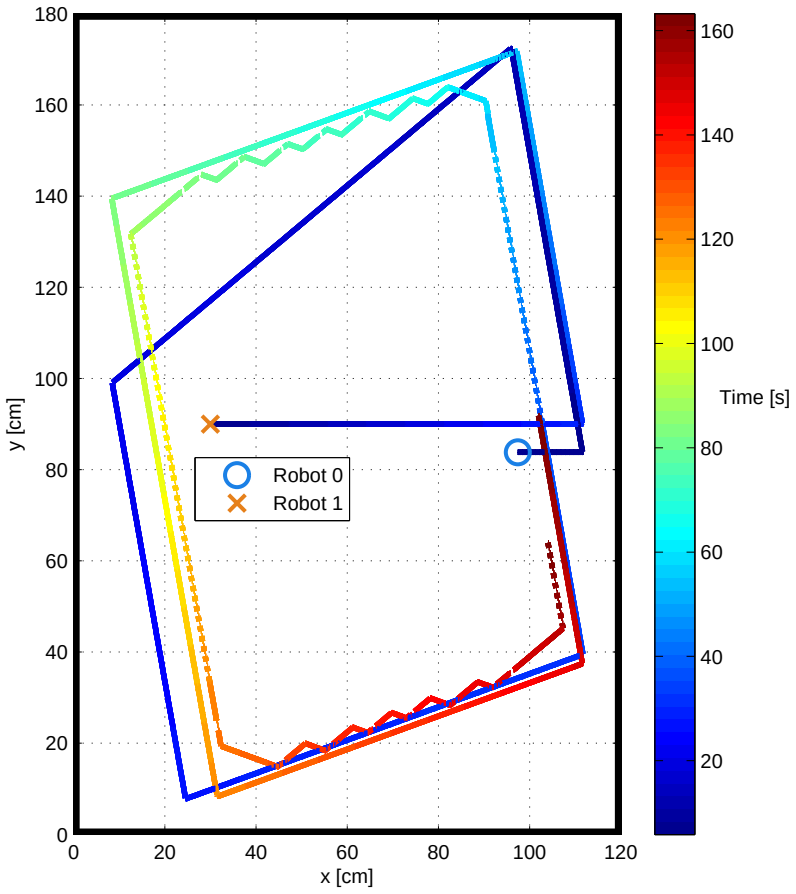


Figure 6.8: Simulation result of two robots' moving traces in security guard behavior.

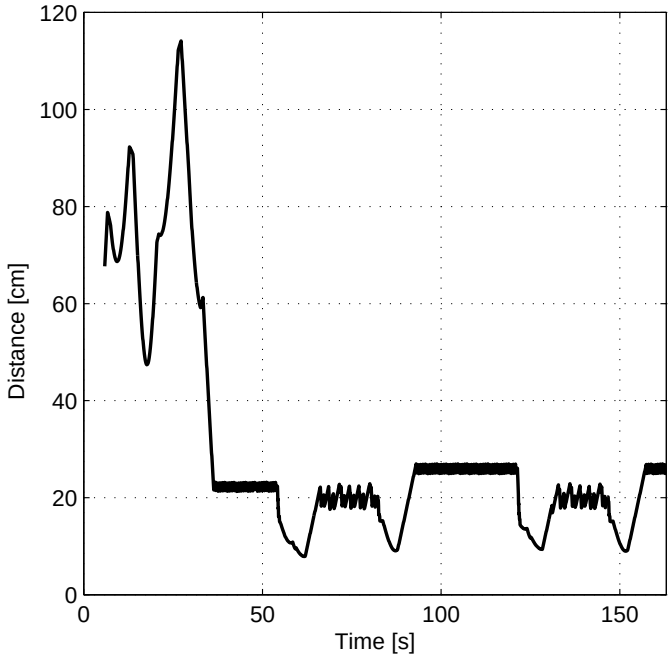
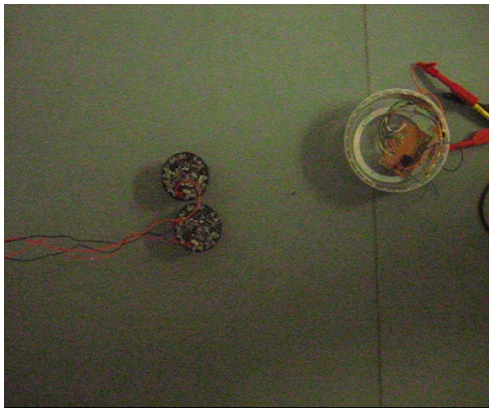
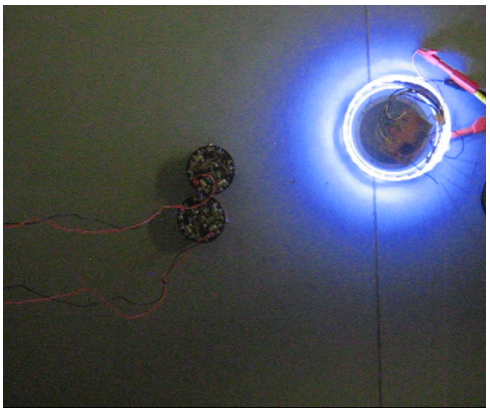


Figure 6.9: The distance between two robots from simulation in security guard behavior.

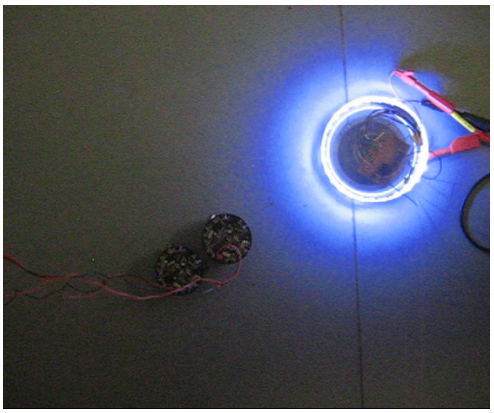
In Section 6.3, the subtask two is discussed. One robot is assumed to be carrying the food and has to find a way to home. Meanwhile, this robot needs the guard from the other robot. The other robot has the responsibility to guard the robot to return home. The algorithms are developed separately for two robots and validated in the simulation and real experiments.



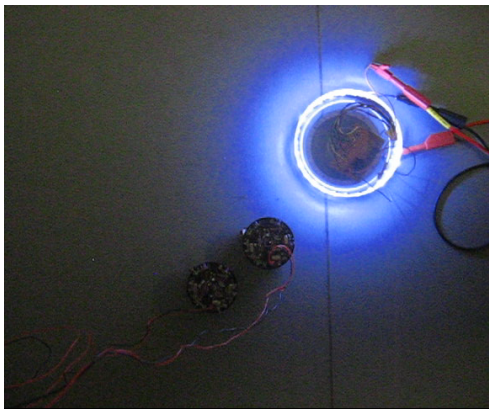
(a) R1 is searching the home and R0 is guarding it.



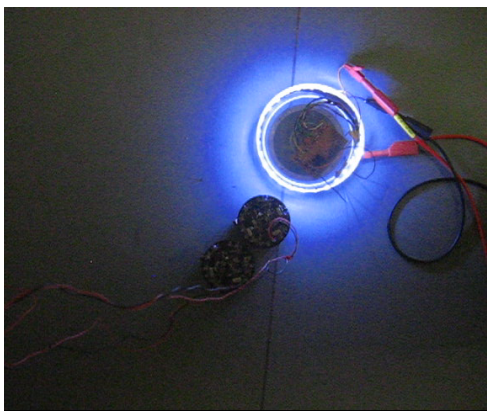
(b) Light is turned on and R1 finds the home.



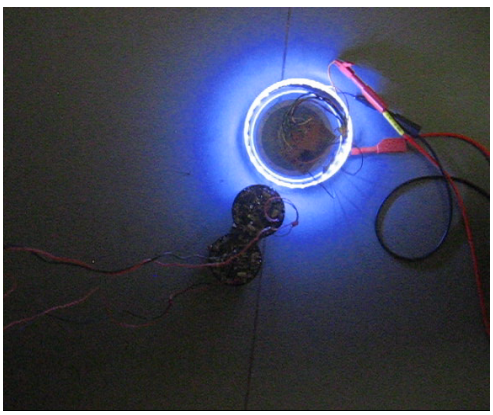
(c) R1 is approaching the the light source and R0 is guarding it.



(d) R1 is approaching the the light source and R0 is guarding it.



(e) R1 is approaching the the light source and R0 is guarding it.



(f) R1 arrives home. R0 finishes this guard.

Figure 6.10: The video clips from an experiment in collective subtask two.

Chapter 7

Closure

7.1 Conclusion

The objectives of this project are focused on the swarm robots implementation. The project starts from a circuit diagram of the robot to the end of realization of swarm robots behaviors. During this period, three main objectives have been accomplished:

1. Two robots were built. The hardware functionalities based on the current frame were analyzed. Especially, a lot of effort was devoted to improving the communication quality and distance measurement ability.

2. The algorithms for signal robot behaviors and swarm robots behaviors are developed, mainly including formation and collective behavior.

3. The developed algorithms are validated by the physical robots system or simulation.

The detailed tasks conducted in this project have been described in this thesis. In the following, there will be a brief conclusion of the whole project.

The robots used in this project was designed by Trung Dung Ngo. For the aim of swarm robots implementation, the robot is supposed to be micro and identical. Each robot was equipped with six couples of IR transmitter and receiver, six ambient light sensors, one color sensor and one Global IR receiver. The actuation system was taken from the chassis of a toy tank. The homemade robot was established by combining sensor system and actuation system. A MCU was used to cooperate those systems with respect to different tasks. Two robots were built in this project.

An object was built in order to play different roles for implementation of different robot's behaviors. The object was round and decorated by four LED arrays with different colors. Programs were developed for remotely controlling these LED arrays by radio frequency. The realization of this function mainly relies on the developed radio frequency modules.

All the sensor except the global IR receiver were tested separately and used for different functionalities of the robot. The communication between robots was realized by using the six IR couples. The protocol agreed by the robots was established based on the UART with the baud rate of $9600bps$. The ability of distance measurement was also achieved by using IR couples. The robot can detect the distance to the wall by measuring the voltage level of reflected IR signal. While the distance to another robot is not measurable by using this approach since there is no big reflecting surface on the top of robot. Therefore, it prefers to estimate the distance to another robot by detecting the low voltage level of the communication signal. However, the estimated distance was much influenced by the direction deflected from the direction of perpendicularly pointing to a wall or a robot. A light source can be detected by six ambient light sensors and the red and blue color detections were realized by the color sensor.

With the above functionalities provided by the hardware, the algorithms for communication and distance measurement were developed firstly. Then the algorithms of three single robot behaviors were developed and validated. A robot can avoid both the wall and another robot by considering the information from reflected IR and communication signal synthetically. The algorithm to estimate the direction of the light source was developed by analyzing the ADC values detected by the six ambient light sensors. A P-controller was designed to control the robot to approach the light source in a smooth way. The algorithm of reacting to the different colors of lights were implemented. The robot can succeed to approach the red light and escape from the blue light.

After the signal robot's behaviors were ensured, two robots were used to implement two swarm robots' behaviors.

The first one was the formation. Two robots were supposed to have a specific formation shape. When two robots meets each other, one robot will ask the other robot to be stopped and perform formation with it. When it performs formation, the robot needs to do direction and distance adjustment sequentially. After formation is generated, two robots are supposed to move together with the formation. There are two approaches to keep the formation when moving. One is to keep the relative direction only. The separated algorithms by using this approach were developed for each robot and validated in both simulation and experiments. The result shows that the two robots can keep the formation

if the moving directions are exactly the same when formation is generated. The other approach is to keep both the relative direction and distance during moving. The algorithms were developed for this case and validated in the simulation. It shows that the robot needs high requirements of maneuverability and flexibility in order for adjusting direction and distance momentarily. In both cases, the developed algorithms can also ensure the robots to avoid obstacles successfully.

The second one was the collective behavior. A collective task was planned and implemented by two subtasks. The first subtask requires one robot finds the food and move the food with the help of the other robot. The object with red light is to simulate the food that robots interest. The corresponding algorithms were developed and proved by the experiments. The second subtask is to let one robot guard the other robot to return home. The other robot is assumed to be carrying the food and its duty is to take back the food to home. The light source from the object with white light is simulated to be the robots' home. The algorithms were also validated by the simulation and real experiments.

As a conclusion, in this project, two robots were built and different algorithms were developed to realize swarm robots' formation and collective behavior. All the algorithms are validated by implementations of the physical robots system.

7.2 Future work

The realization of swarm robots behaviors much relies on the ability of each robot member. The more abilities provide to every single robot, the more complicated swarm behaviors can be realized. However the robot used in swarm robots is supposed to have a small size, which leads to the drawback of limited capabilities. Because of this tradeoff, the single robot's design should be always paid attention to in the future work on swarm robots. The main discussion in this section will be based on the current implemented robots.

The independent power system is relative important for distributed robots system. The power used for the robots is taken from an power supply instrument. A rechargeable battery has not built into the robot and has to be added in the future. A suitable rechargeable battery should be selected by mainly considering the characteristics of capacity, voltage and size related with the energy dissipation and structure of the robot.

The global IR receiver was tested during this project, but it can not function as well as expect. The features of communication distance and the aperture angle are neither satisfied. In expectation, with this global IR receiver, the robot can have more chance to

receive IR signals globally and save the energy spent on switching the local IR channels. This global IR receiver needs to be investigated and improved in the future.

The communication is significantly important to implement swarm robots. For current robots' communication, six IR couples have to be switched one by one. Two robots can only communicate when a switch-on IR transmitter of one robot points to a switch-on IR receiver of the other robot. If the communication speed were increased, during every one second, the chance to get communication signal would be increased. In order to increase the communication speed, the most effective suggestion for the current design is to replace IR phototransistors with IR photodiode. The photodiode has the less response time than the phototransistor. If the ability of communication is expected be largely enhanced, the hardware layout in the communication part should be redesigned. Actually, seen from Figure A.2, the communication part is relatively improved.

From the communication signal, the robot can also know the relative direction to the other robot. There are six IR couples in the current robot, so the maximal resolution to decide the direction is 60°. If adding more IR couples with smaller aperture angle, the resolution would be smaller and the direction could be decided more precisely. As a tradeoff, the probability to get the communication signal would be even less than the current $\frac{1}{36}$.

For measuring distance, the current robot has a problem to detect the distance to the other robot since there is no obvious reflector can be used on the height of IR couples. Thus, it suggests to vertically fix six partitions between six sectors. These partitions can be used not only as reflectors but also to isolate the interference from the neighboring IR couples.

The actuation system is not robust in the current assembled robot, which can not be controlled accurately. Therefore, a new actuation should be designed by synthetically considering the motor system, gear system, wheel system and placement.

Because only one MCU is used for one robot, every functionality has to be executed sequentially depending on the program. For the current developed algorithm, in order to get valid communication signal, the communication function has to take much more time than other function. In order to control the robot close to the real time, it suggests to let the other slaver MCU take in charge of the most time-consuming or heaviest function in the master MCU. The Two Wire Interface (TWI) interface has already designed in the PCB and can be used to transfer the data from slaver MCU to the master MCU. Also, the flash memories are enlarged.

The current behaviors of swarm robots are only implemented by two robots. For the real meaning of swarm robots, two robots is not enough. Therefore, it is expected to build

more micro robots and implement them. If more robots were playing in a limited scenario, there would be more interference on the robots' communication. One solution is to give each robot a characteristic to distinguish from each other. The distinctive characteristics can be realized by using different modulated communication signals for different robots. One approach is to use the different frequencies of carrier waves to carry the original signals. The robot on the receiving side only extracts the signal with its interested carrier wave.

All the suggestions mentioned above are expected to be implemented in the future by a large group of the next generation micro robots according to the circuit diagram in Figure A.2.

Acronyms

AAU Aalborg University

ADC Analog-to-Digital Converter

DCE Data Circuit-terminating Equipment

DTE Data Terminal Equipment

FSM Finite State Machine

IR Infrared Ray

ISP In-System Programming

ISR Interrupt Service Routine

MCU Microcontroller Unit

MIPS Million Instructions Per Second

PCB Printed Circuit Board

PWM Pulse Width Modulation

RISC Reduced Instruction Set Computer

SRAM Static Random Access Memory

TWI Two Wire Interface

UART Universal Asynchronous Receiver/Transmitter

USART Universal Synchronous/Asynchronous Receiver/Transmitter

Bibliography

[adp04] Si9986 datasheet, 2 2004.

[atm06] Atmega8l datasheet, 10 2006.

[BA98] Tucker Balch and Ronald C. Arkin.
Behavior-based formation control for multirobot teams.
1998.

[bre] Breve.
http://www.spiderland.org/ .

[CL94] Q. Chen and J. Y. S. Luh.
Coordination and control of a group of small mobile robots.
1994.

[cmp] Communications protocol.
http://en.wikipedia.org/wiki/Communications _protocol .

[cri] Crispavr-usb.

[DRJ95] B. Donald D. Rus and J. Jennings.
Moving furniture with teams of autonous robots.
1995.

[EYM94] J. Ota E. Yoshida, T. Arai and T. Miki.
Effect of grouping in local communication system of multiple mobile robots.
1994.

[Gag92] D. W. Gage.
Command control for many-robot systems.
1992.

[idw] idwarf-168 datasheet.

[inf] idwarf-net firmware.

[jas] Jasmine robots.
http://www.swarmrobot.org/ .

[lit] idwarf-hubboard.

[lmv99] Lmv331m5 datasheet, 8 1999.

[mcu] Mcu.
http://en.wikipedia.org/wiki/Microcontroller .

[NG01] NASA-GSFC.
Satellite formation flying concept becoming a reality.
2001.

[OKC96] K. Chang D. Ruspini R. Holmberg O. Khatib, K. Yokoi and A. Casal.
Vehicle/arm coordination and mobile manipulator decentralized cooperation.
1996.

[Par94] Lynne E. Parker.
Heterogeneous multi-robot cooperation.
1994.

[Par00] Lynne E. Parker.
Current state of the art in distributed autonomous mobile robotics.
2000.

[pwm] Pulse-width modulation.
http://en.wikipedia.org/wiki/Pulse-width_modulation .

[rs2] Rs-232.
http://en.wikipedia.org/wiki/RS-232 .

[SB93] Daniel Stilwell and John Bay.
Toward the development of a material transport system using swarms of ant-like robots.
1993.

[sbt] Swarm-bots.
http://www.swarm-bots.org/ .

[sf4] Sfh487 datasheet.

[sfh99] Sfh309fa datasheet, 2 1999.

[sro] Swarm-robotics.org.
http://swarm-robotics.org/SAB06/ .

[TaS89] H. Ogata T. arai and T. Suzuki.
Collision avoidance among multiple robots using virtual impedance.
1989.

[tcs04] Tcs230 datasheet, 1 2004.

[usa] Usart.
http://en.wikipedia.org/wiki/USART .

[Wan89] P. K. C. Wang.
Navigation strategies for multiple autonomous mobile robots.
1989.

[Yam97] H. Yamaguchi.
Adaptive formation control for distributed autonomous mobile robot groups.
1997.

[ZWN00] T. Takahashi Z. Wang, Y. kimura and E. Nakano.
A control method of a multiple non-holonomic robot system for cooperative object transpotation.
2000.

Appendix A

Circuit Diagram

The Figure A.1 shows the pins configuration of ATmega8L, which is used as the MCU for robots in this project.

The Figure A.2 is the new design of a robot by Trung Dung Ngo, which can be implemented for the next generation of robots. It can be seen clearly from the attached CD.

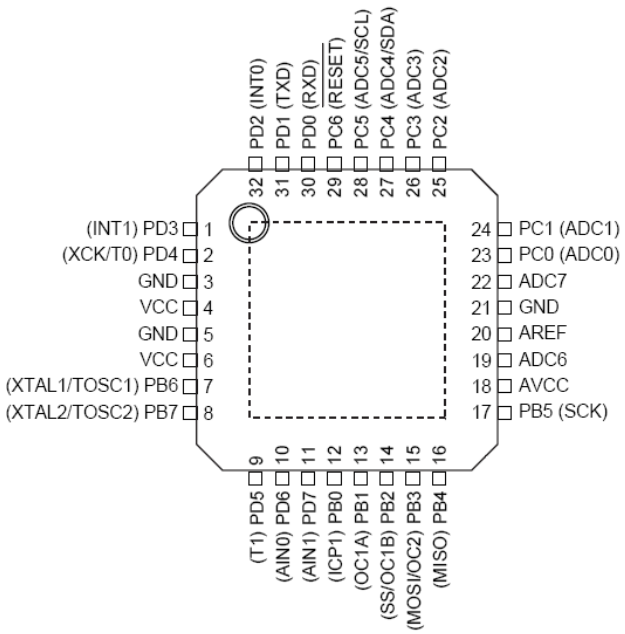


Figure A.1: MLF top view of ATmega8L.

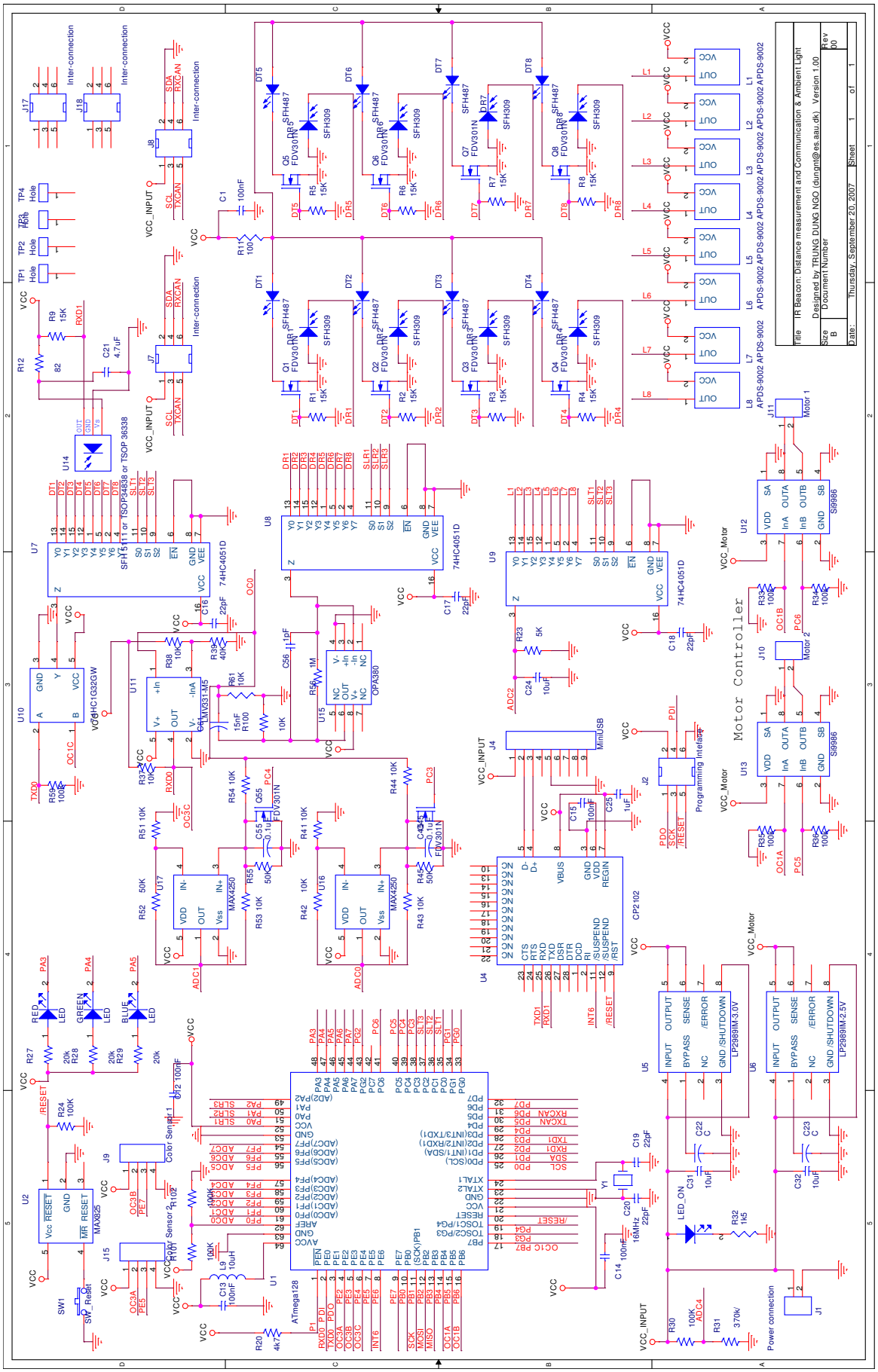


Figure A.2: Circuit diagram of new version of the robot..

Appendix B

LED Control

Four LED arrays, which are used to decorate the object, can be remotely controlled by using the wireless radio modules, **iDwaRF-168**. One module connects to the PCB for the LED arrays and the other one is connected with a host PC. The LED arrays can be switched by commands.

The commands are created by programming based on the **iDwaRF-Net Firmware Version 2.1a**, and can be entered with keyboard by using a terminal program, e.g., **HyperTerminal**. The following commands are mainly used in this project. More command can be found in [inf].

rst

The remote device will be reset and bind again and its ID will be reassigned.

enu

To enumerate the current bound remote device.

cln

To clean up the double entries of same device if it is reset.

snd (*deviceID*) -1 (*color*)

The variable *color* can be characters *r*(red), *b*(blue), *y*(yellow), *w*(white) and *c*(clean). For example, sending *r* first time will turn on the red LEDs, and sending *r* second time will turn off the red LEDs. Sending *c* can turn off all the lights.