Ministry of Higher Education
And Scientific Research
University of Diyala
College of Engineering
Computer and Software
Engineering Department

Programming and operating of Omni-Directional Mobile Robot

A project
Submitted to the Department of (Computer and Software Engineering Dep.)-University of Diyala-College of Engineering, in Partial Fulfillment of the Requirements for the Degree of Bachelor (Computer and Software) Engineering

By

Abeer Jassam Mhmood & Zainab khazal

Supervised

MSc. Hussein Sultan Radhi

2016

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

(قَالَ رَبِّ اشْرَحْ لِي صَدْرِي (25) وَيَسِّرْ لِي أَمْرِي (26) وَاحْلُلْ عُقْدَةً مِنْ لِسَانِي (27) يَفْقَهُوا قَوْلِي (28))

صدق الله العظيم

(سورة طه)

DEDICATIONS

For our Beloved Families ……

…… Friends

…… Classmates

…… Lecturers

ACKNOWLEDGEMENTS

In the name of Allah, the Most Beneficent, the Most Merciful

We would like to express our heartiest thanks to our supervisor **"MSc Hussein Sultan Radhi."** for his guidance, patience, advice and devotion of time, throughout our research. He taught us how to be a researcher with his wonderful personality. We have the honor to work under his supervision and we will ask Allah to keep him safe, and support him with good health and the power to help the students with knowledge and his scientific abilities. We are forever indebted to them for excellent guidance.

Finally, family for their unconditional care until we reach to this point, without their continuous support and precious prayers we could not have been able to finish our research. We know their blessings will always be with us in all our endeavors and we dedicate this success to them. Thanks our beloved family.

SUPERVISOR CERTIFICATION

I certify that this project entitled **"Programming and operating of Omni-Directional Mobile Robot"**, was prepared under my supervision at the Computer and Software Engineering Department/College of Engineering by (**Abeer Jassam Mhmood & Zainab khazal**) as a partial fulfillment of the requirements for the degree of B. Sc. in Computer and Software Engineering.

Signature:

Name: MSc; **MSc. Hussein Sultan Radhi**

Title: Asst. Lecturer

Date: / /

I certify that I have carefully read and corrected the final draft of this project for errors in punctuation, spelling and usage.

Proof reader's signature:

Name: MA. Areej S.Dawood

Title: Lecturer

Date: / /

In view of the available recommendations, I forward this project for debate by the examining committee.

Signature:

Name: DR. ALI J. ABBOUD

(Head of Department)

Title: Lecturer

Date: / /

CERTIFICATION OF THE EXAMINATION COMMITTEE

We certify that we have read this project entitled **"Programming and operating of Omni-Directional Mobile Robot"** and as examining committee examined the students (**Abeer Jassam Mhmood&**zainab khazal) in its contents and that in our opinion it meets the standards of a project for the degree of B. Sc. in Computer and Software Engineering.

Signature:

Name :

Title :

(Member)

Date : / / 2016

Signature:

Name :

Title :

(Member)

Date : / / 2016

Signature:

Name :

Title :

(Chairman)

Date : / / 2016

Approved for Computer and Software Engineering Department.

Signature:

Name: DR. ALI J. ABBOUD

(Head OF Department)

Title: Lecturer

Date: / / 2016

ABSTRACT

This project presents the movement of the robot on black line based on infrared radiation (IR) also is make robot to avoid obstacles based on Position Sensitive Detector (PSD). The Omni-directional Mobile Robotics system model is designed and implemented with AVR Studio4, liquid crystal display (LCD) minimum screen, DC Geared Motor, IR sensing and PSD sensing. The proposed system is designed using C++ language. Current applications of mobile robots are broad and include domestic and public cleaning, transport of goods in hospitals, factories, ports and warehouses, exploration of in hospital terrains such as space or oceans, mining, defusing explosives, entertainment and performing inspections and security patrols.

TABLE OF CONTENTS

Subject Page

TABLE OF CONTENTS

LIST OF FIGURE

Subject Page

Abbreviations

AVR……Acidification/Volatilization/Reneutralization

PSD……. Position Sensitive Detector

API……..Application Programming Interface

ISP ……..In System Programming

LCD ……Liquid Crystal Display

LED…….Light Emitting Diode

RISC…… Reduced Instruction Set Computing

ICE……In Circuit Emulator

JTAG……Joint Test Action Group

Declaration

we hereby declare that my project entitled "**Programming and operating of Omni-Directional Mobile Robot**" is the result of our own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text, and is not substantially the same as any that we have submitted, or, is concurrently submitted a degree or diploma or other qualification at the university of diyala or any other university or similar institution except as declared in the preface and specified in the text. We further state that no substantial part of our thesis has already been submitted, or is concurrently submitted for any such degree, diploma, or other qualification at the university of diyala or any other university or similar institution except as declared in the preface and specified in the text.

The names of students:-

1-**Abeer Jassam Mhmood**

2- Zainab khazal

Chapter One

Introduction

CHAPTER 1

INTRODUCTION

1.1 Background:

The robot is a man instinctively shows interest in moving and it is no wonder many people are fascinated by a moving robot. Wider range of knowledge and experience, however, are required to learn, design and maintain principles of robots movements. The robot platform is based on an optimal design for ED-7275 mobile robotics and is also designed to cope with various practices and tasks for application. This robot platform can be used for conducting a basic practice on various sensor theories and features based on a microprocessor and learning a robot control system using various functional sensors. It consists of the API (Application Programming Interface) to help users understand and use the most basic robot-driving part and sensors easily. ED-7275 robot is characterized by its driving mechanism, which enables a robot to turn in all directions at a standstill based on an omni-directional mobility mode, so that users can understand and learn the robot easily. Also the robot can be easily manipulated with sensors and its platform can be applied to various tasks for application [1].

The term Robotics was newly coined in a science fiction Isaac Asimov first published in 1942, where the writer described Asimov had both a negative side and a positive one for a robot. Besides Asimov has set the following basic three rules as robot characteristics:

1. A robot cannot do any harm to a man and must not keep a man neglected in an urgent situation by not moving itself.

2. A robot must obey all commands given by a man, expect for the ones that is against the first rule.

3. A robot must protect itself as long as it stays within the first and second rule.

1.2 Literature Survey:

This section presents a brief survey of the work presented in the literature, where related to the field of this project. The knowledge of the existing technology is important for the proper design of any robotic system. Mechanical architecture and sensor technology are important fields to consider for the development of a mobile robot for any specific task. Understanding of the advantages and shortcomings of various systems is important to choose a system for the specific task. An omni-directional mobile robot is a type of holonomic (the relationship between the controllable and total degrees of freedom of a given robot) called (Robotics and Autonomous Systems 56 (2008) 461–479). It has the ability to move simultaneously and independently in translation and rotation. The inherent agility of the omni-directional mobile robot makes it widely studied for dynamic environmental applications [2,3]. The annual international Robocop competition in which teams of autonomous robots compete in soccer-like games, is an example where the omni-directional mobile robot is used.

The second robot is called (3WD Triangular 100mm omni wheel mobile robotics car 10003 [4]**)**.

Features of 3WD Triangular 100mm omni wheel mobile robotics car 10003 is:-

Three-wheel drive. Aluminum alloy body. Omni wheel. Easy to assemble. DC motors with encoders.

While the robot is currently characterized by:-

Enabling users to learn a basic theory on robot control and have practices on its application by using the Embedded Microprocessor, 3-Axis motor, IR sensor, PSD and omni-directional drive used in a mobile robot. Enabling overall practices on a mobile robot by using a floor-detecting distance sensor (PSD: Positive Sensitive Detector). Consisting of a unique mechanism by using omni-wheels attached at an interval of 120 to drive a robot in an omni-directional. Including five IR sensors to detect a certain color and deliver information to Host PC by using an interrupt mode. Including three PSD sensors basically and can also have a combination of six sensors including PSD sensor. Providing a related library (API) to control a robot directly by using a user application program (C language). Enabling users to check a robot status easily by having a User button and blue character LCD to control hardware attached to the robot.

1.3 Research Objectives:

Research objectives of this project are summarized below:-

1. Identify the components and the program designed by and the parts that need to be programmed.
2. To design a robot programming using C++ language and AVR studio4.
3. To download the programming to the robot by using ISPAVR usb.

1.4 Outline of the Project:

The Project is basically divided into four chapters.

Chapter 1 provides general background to the field of the search, and feature of robot.

Chapter 2 describes the field of the project software; by including introduction of the AVR Studio 4 C programming language, the software algorithms and features, programming of ATMega128 and how to programming parts of Omni-directional Mobile Robotics.

Chapter 3 presents the project results and the simulation results.

Chapter 4 presents a conclusion of the entire design and results as well as the recommendation for future work.

Chapter Two

Robot Design & Tools

CHAPTER 2

ROBOT DESIGN & TOOLS

2.1 Introduction:

The software for this project is built with simplicity, flexibility and robustness in mind. The most important aspects of the software is the control on the robot to do the missions that require it to do. It would be used in the design program that controller on robot using ATmega128 because it has better properties than the other types due to the popularity of ATmega128 with developers. It includes powerful atmel ATMega128 microcontroller with 128kb internal flash program memory, wide availability, operating speed at 16MHz, power LED, reset button and AVR Studio 4 C programming language to program the ATMega128 [5].

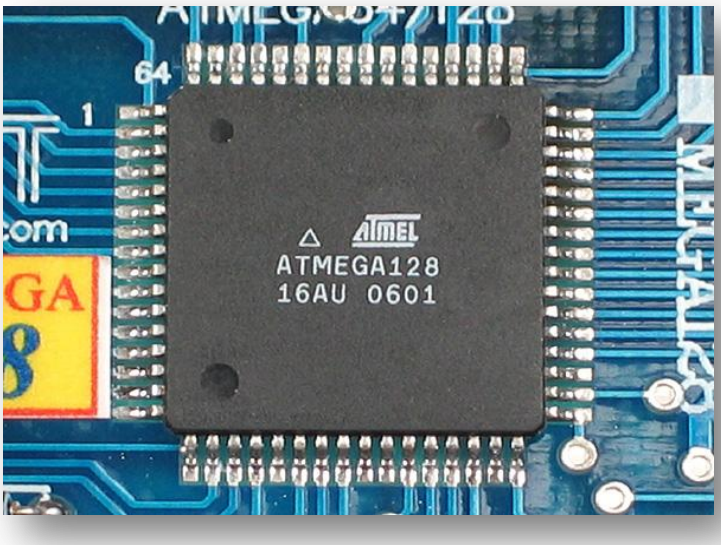


Figure (2.1): ATMega128 Chip for robot [5].

2.2 Software Algorithms and Feature:

The software offers high degree of flexibility with many integrated features When the program is start, it checks the black line it was :-

1-Found.

2-Not found.

If the black line is the robot will move and continue to move to the end of the black line and then back to the starting point, stops, and this move depends on the IR sensor as well as motor. And the robot avoids obstacles using PSD sensor.

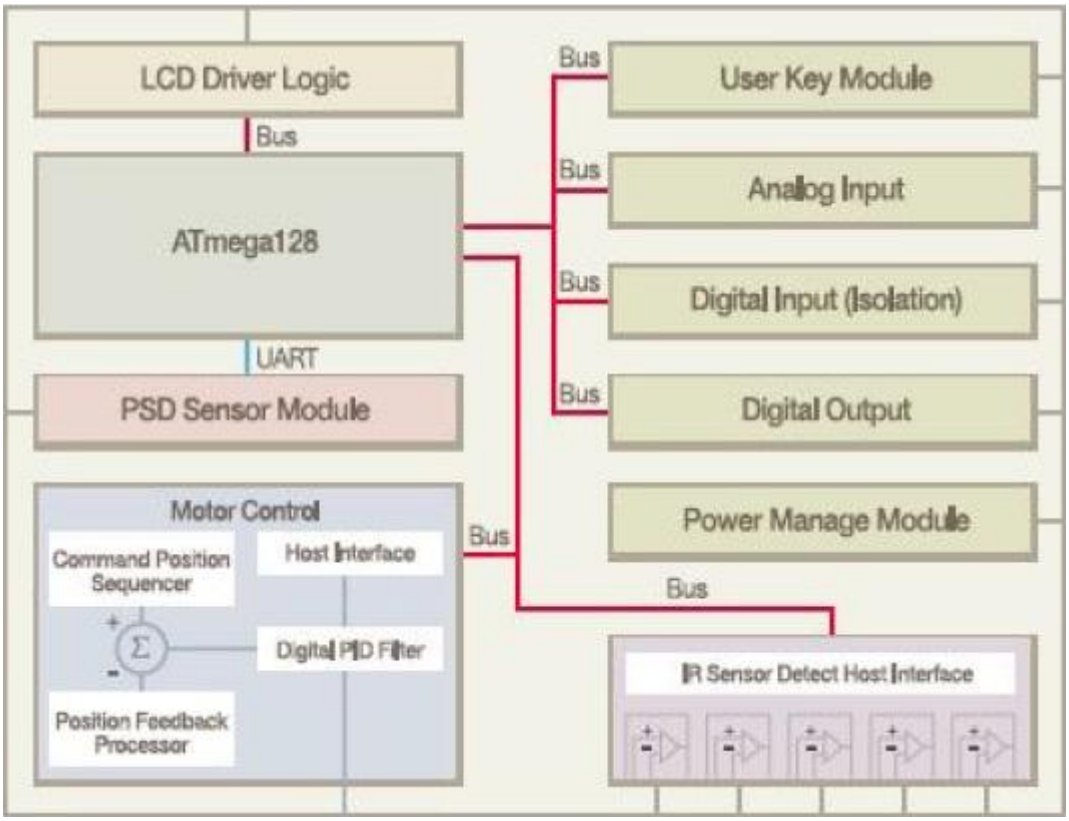


Figure (2.2) Whole Block Diagram [1].

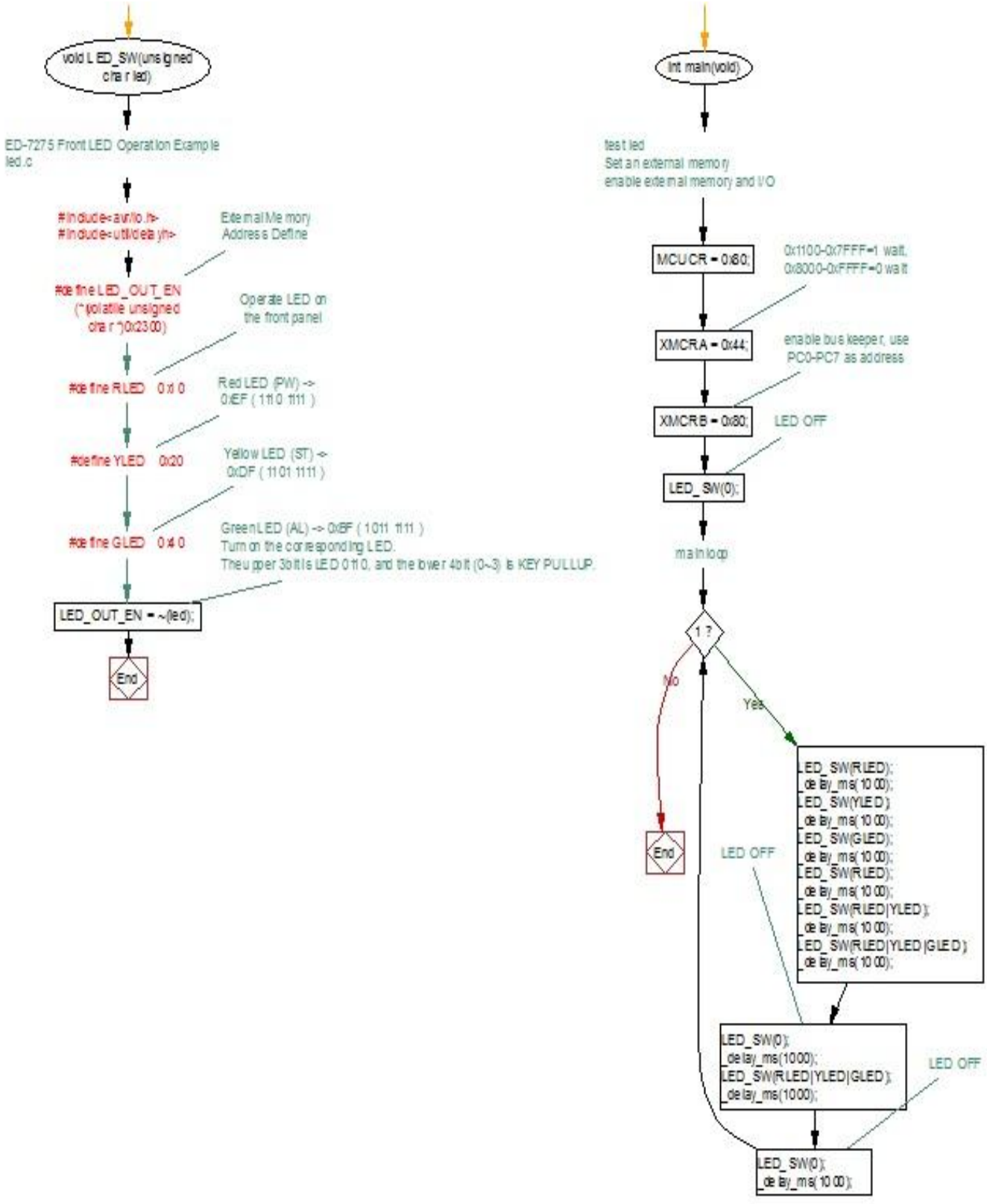


Figure (2.3) Flowchart Of Lcd.

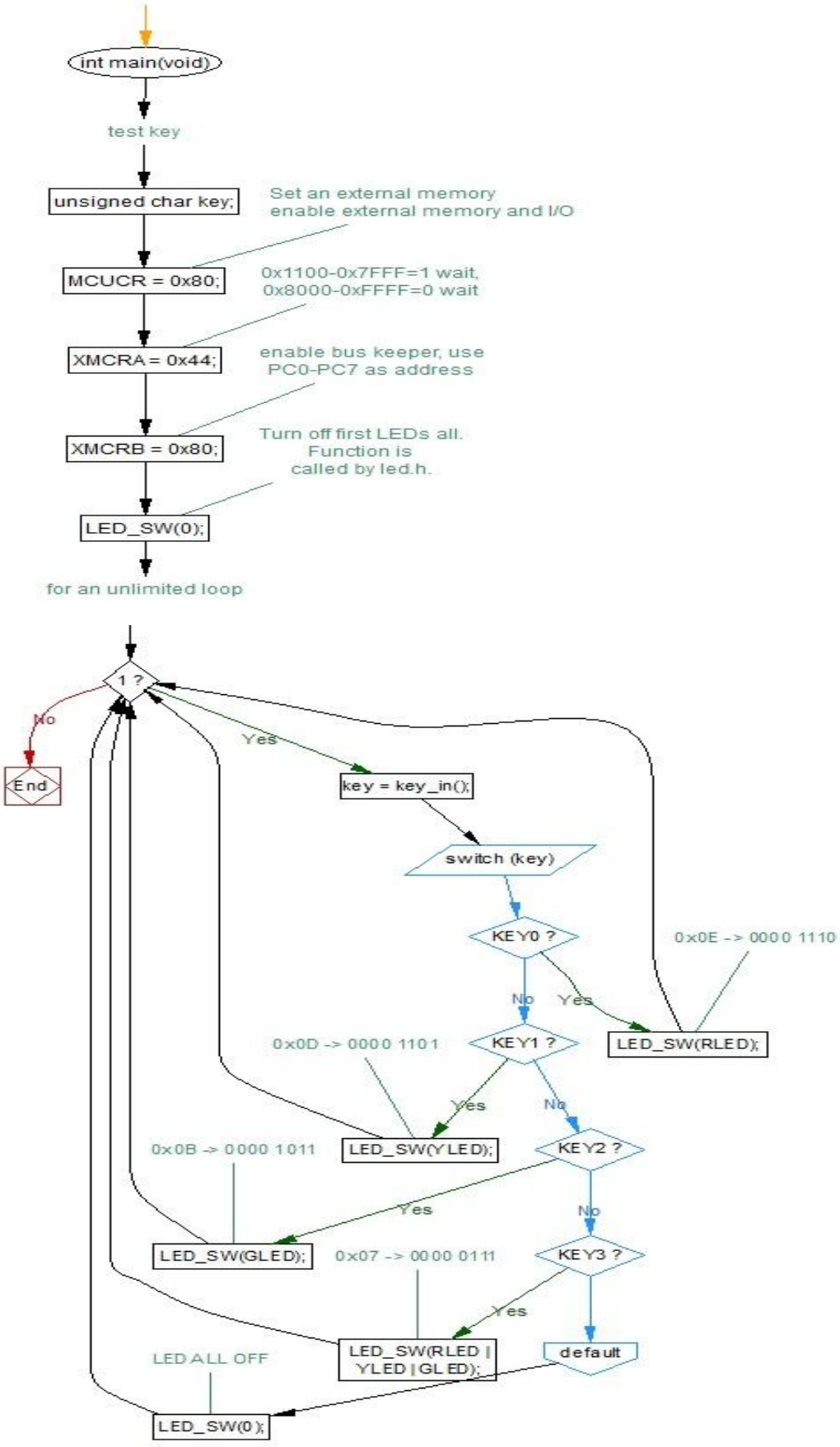


Figure (2.4) Flowchart Of Key.

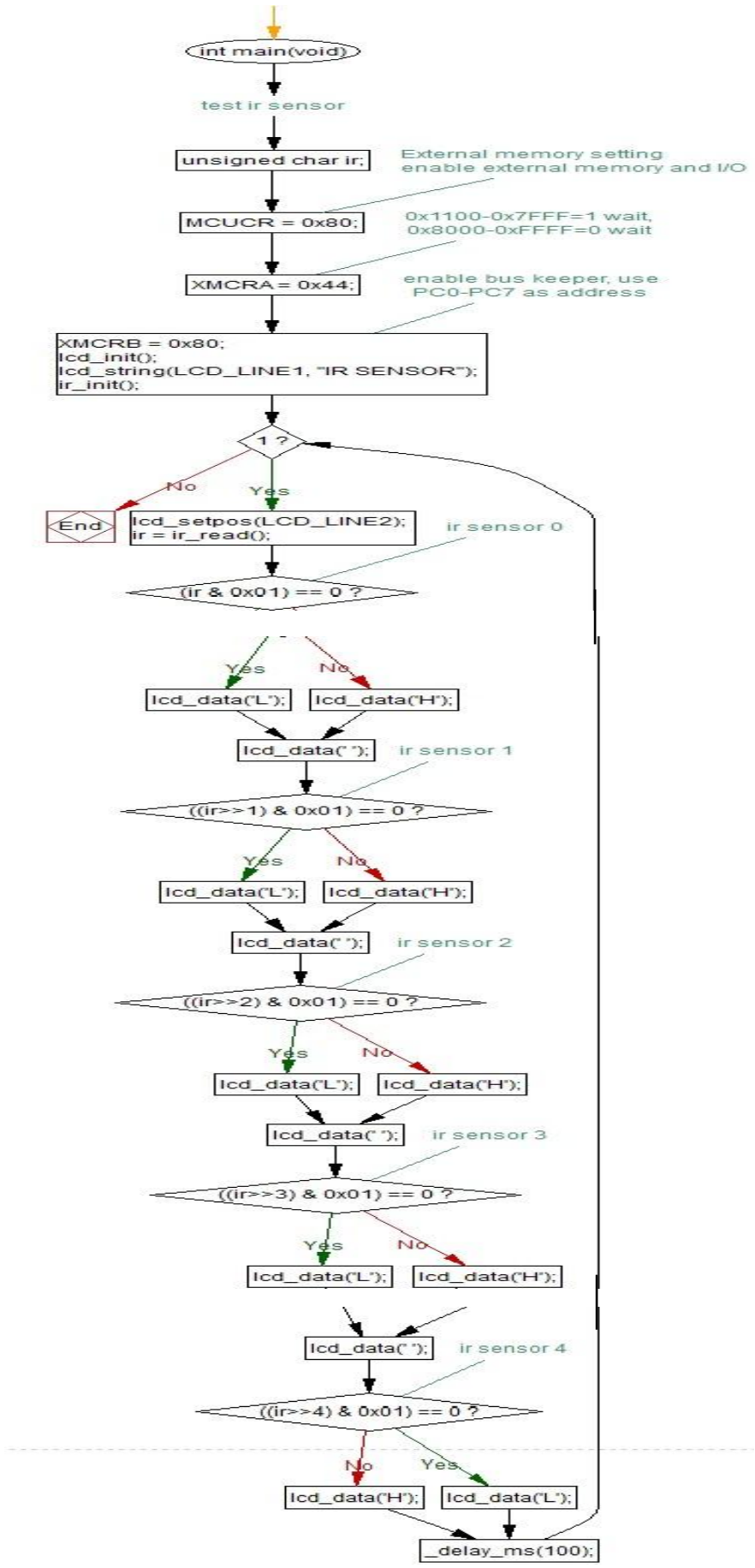


Figure (2.5) Flowchart Of IR.

2.3 AVR Studio 4 C Programming language:

The AVR Studio 4 is an Integrated Development Environment for debugging AVR software. The AVR Studio allows chip simulation and in-circuit emulation for the AVR family of microcontrollers. The user's interface is specially designed to be easily used and to give complete information overview. The AVR uses the same user interface for both simulation and emulation providing a fast learning curve. AVR Studio 4 has many interactive debugging features. For the purposes of this class the two basic methods you will use to debug are the AVR Simulator and the JTAG. At the bottom of your screen AVR Studio displays what device/chip it will interact with when debugging and which debugger it will use. Below is a picture of the Status Bar showing that the device is an "ATMega16" and the debug platform is the "AVR Simulator" [6]. To change the target device for debugging and/or the debug platform go to the "**Debug"** menu and select the menu option **"Select device and debug platform. "** In this window you can select AVR simulator if you want to debug in simulation or if you want to debug directly on your hardware choose either JTAGICE or JTAGICE mkII (depending on which JTAG you have at your station) [6].

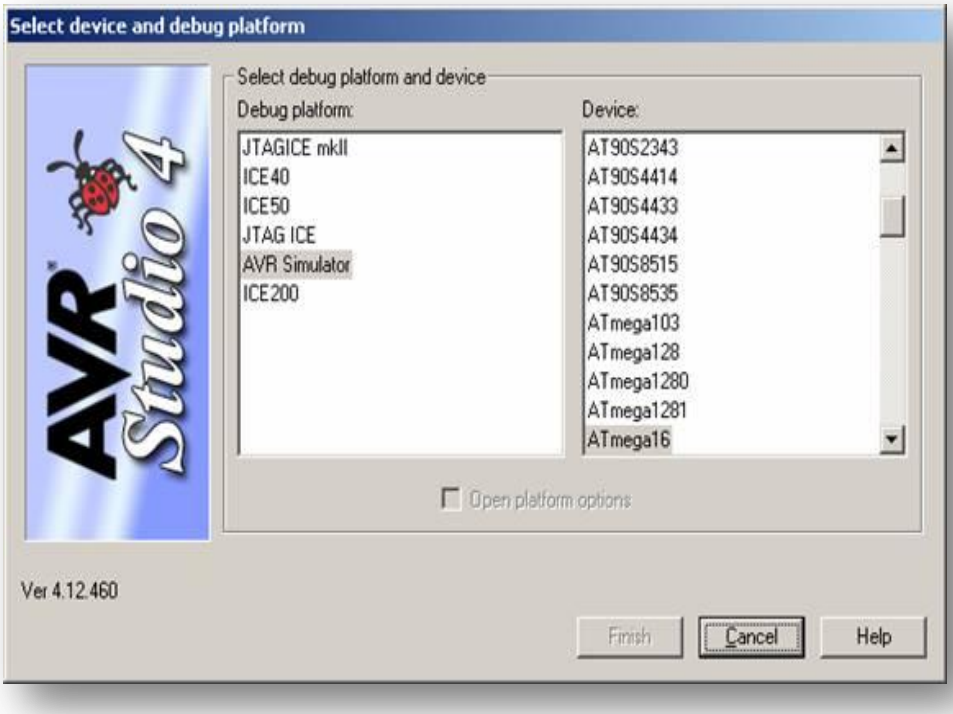


Figure (2.6): Select device and debug platform [6].

One of the most powerful features of the debugger is the ability to view the current states of any of the registers. You can view the current value of any register in the "**I/O view"** tab.

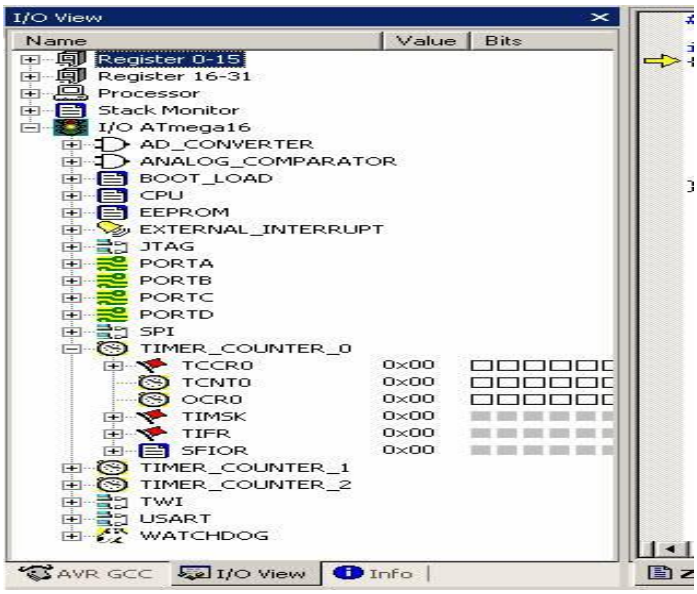


Figure (2.7): I/O View for AVR studio4 [6].

Another useful feature of the AVR Studio 4 debug system is the watch window which allows you to see what the value any variable is in the RAM. To view the current value of a variable use, add the variable to the watch window [6].

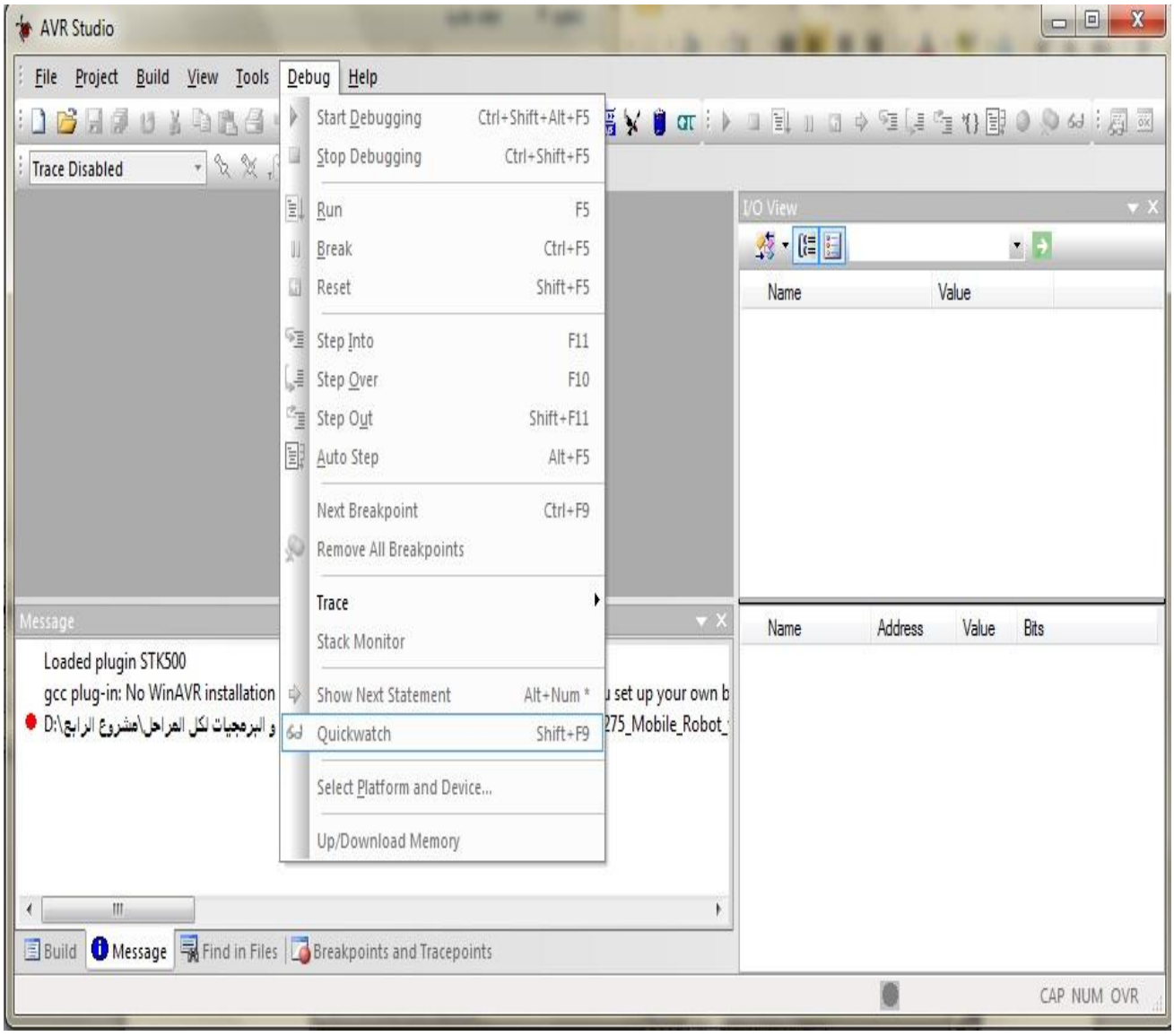


Figure (2.8): watch window for AVR studio4 program [6].

2.4 Parts Of Robot:

This project is the entry of parts and upon which we conduct experiments and are (IR sensor, DC motor, PSD sensor, Battery and charger) which will be explained:-

1- IR sensor.

An infrared radiation (IR) sensor is attached to the bottom of a robot in order to detect a certain color of a floor. The IR sensor detects a certain color as its light-emitting element sends light and its light-receiving element measures an amount of that light reflected and returned correspondingly, as shown in the figure below.

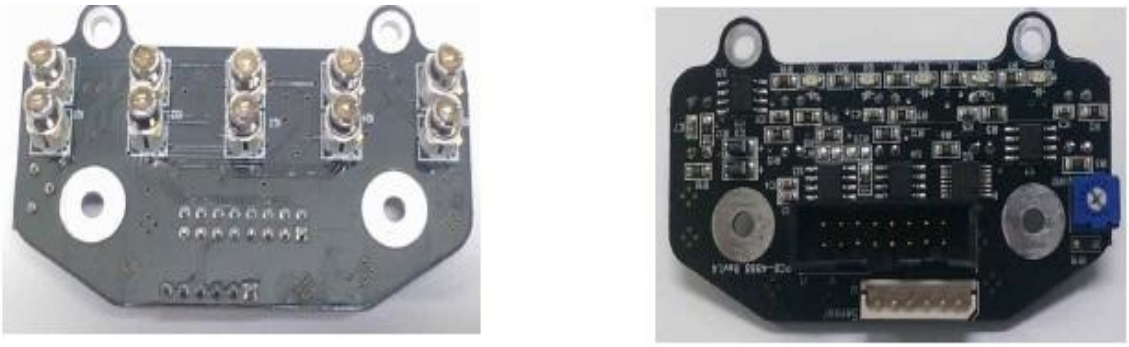


Figure (2.9): IR sensor for robot.

2- DC motor.

The driving method of ED-7275 consists of an unique type of mechanism in which a robot drives with omni wheels dynamically in all directions. This product is attached with three omni-wheels at an interval of 120°, as shown in the figure below.

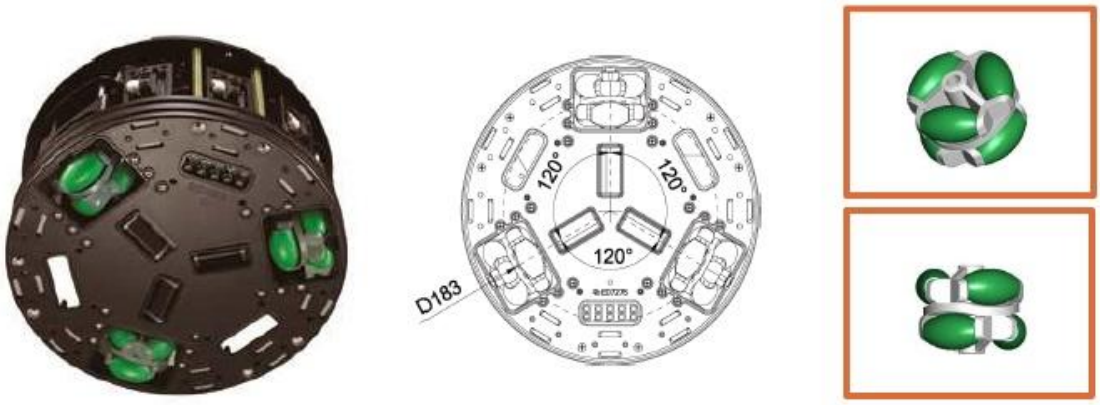


Figure (2.10): omni-wheels for robot.

3- PSD sensor.

PSD (Position Sensitive Detector) is a semiconductor that responds to light. It is possible to configure a distance sensor for detecting a position by using a triangulation as one of the most representative properties of the PSD. A distance at a light-sending lens and light receiving lens, and a distance to the PSD are reduced according to a proportional relationship of a lens system, so a distance a to an object can be calculated according to the following formula, as shown in the figure below.

Figure (2.11): PSD sensor for robot.

4- Battery and charger.

①Battery Check Switch : Checks a current battery level.

②Battery Level Indicator : Indicates a current battery level in five steps. When an amount of battery remains is less than 10%, an LED on the bottom turns on every one second, as shown in the figure below.

Figure (2.12): Battery for robot.

①Battery connecting terminal : Connecting with a smart battery-charging module. Charging time (2ah) : 1.5h, charging time (4ah) : 3h, Charging

②Check Indicator LED : Indicates Red while a current battery is being charged, and displays Green when its charging is finished.

③Charging plug : Even while charging a battery, users can supply external power to a robot with a charging plug. Users can charge a battery and drive a robot while supplying power, as shown in the figure below.

Figure (2.13): Charger for robot.

2.5 Programming the ATMega128-chip:

How ATMega128 chip programming will be done by four points and is:-

1- It is to set an emulator to be used as a debugger. Because a debugger is not used now, select ATmega128 in Device only and click Finish.

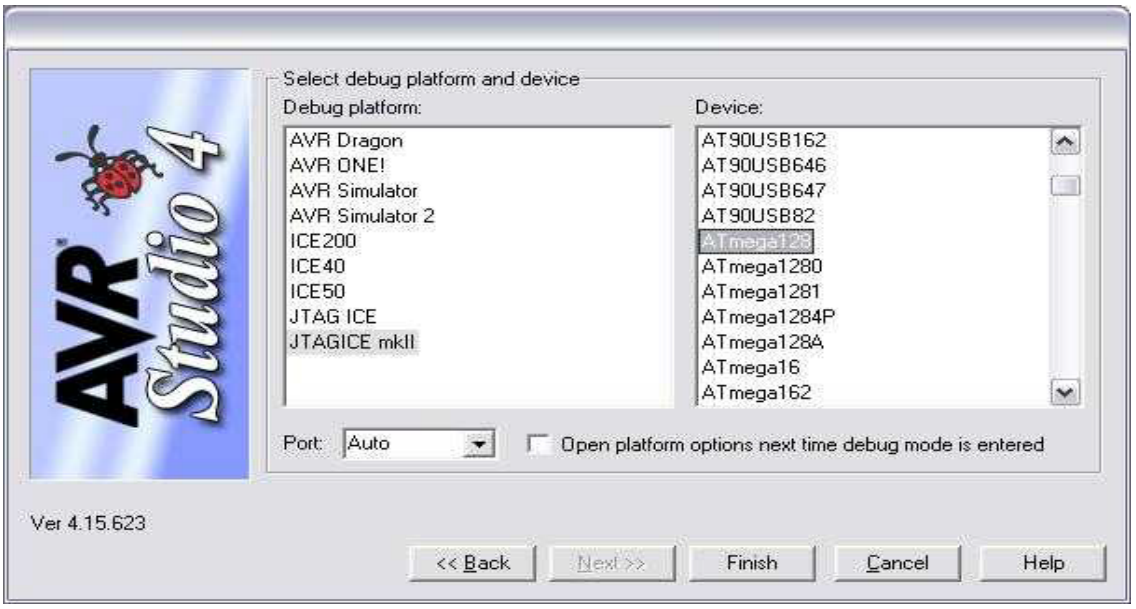


Figure (2.14): The beginning of the window for AVR Studio 4 Program.

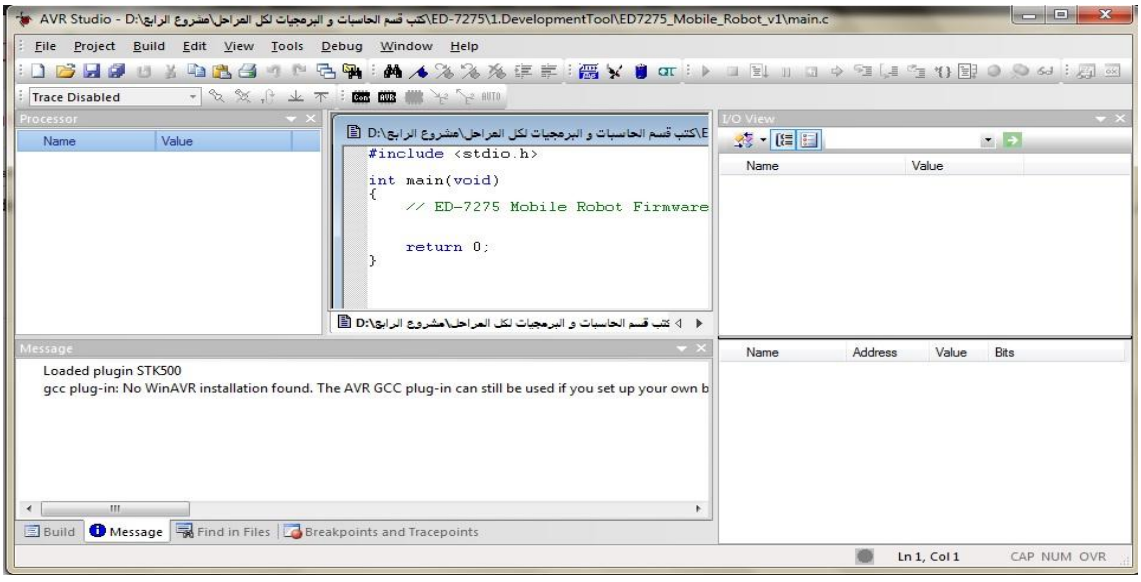


Figure (2.15): Windows of new project for AVR studio4 program.

2- Open the code file to be written in the AVR Studio 4.

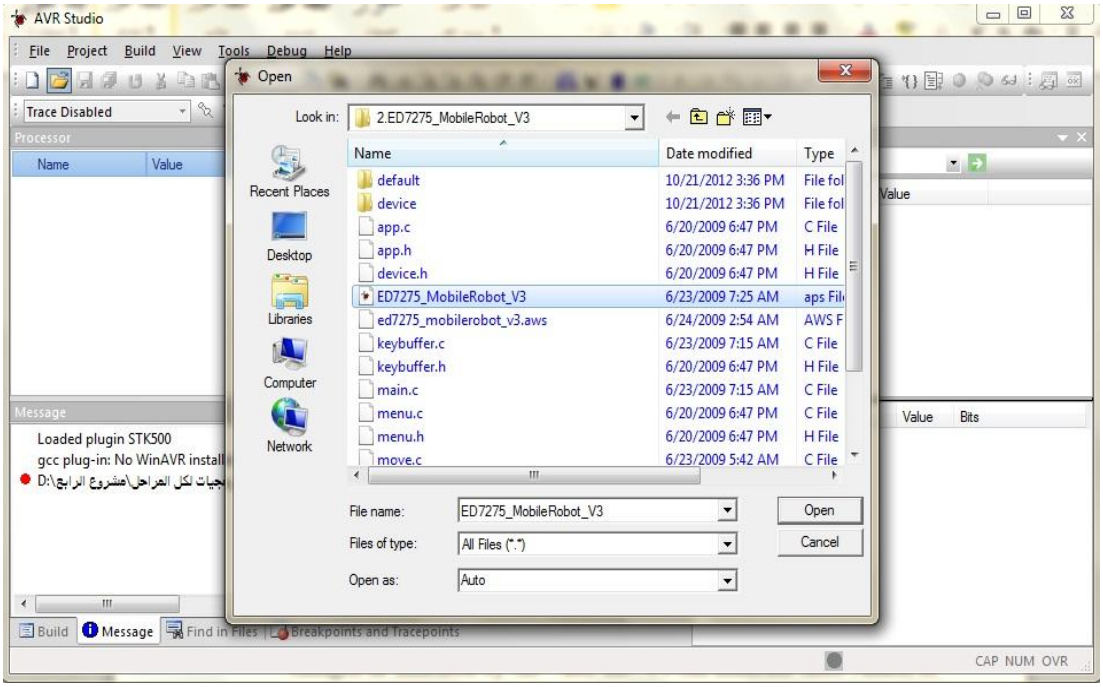


Figure (2.16): Windows of project File selection for code of the robot.

Code is stored in format of *.hex file, select the ED7275-Mobile Robot_v1.hex file, and click the Open button. We can maintain these values, or change them later if we have problems during the programming process [3].

3- Verifying the code.

Click the icon  or Select Menu →Build→build to compile a source.

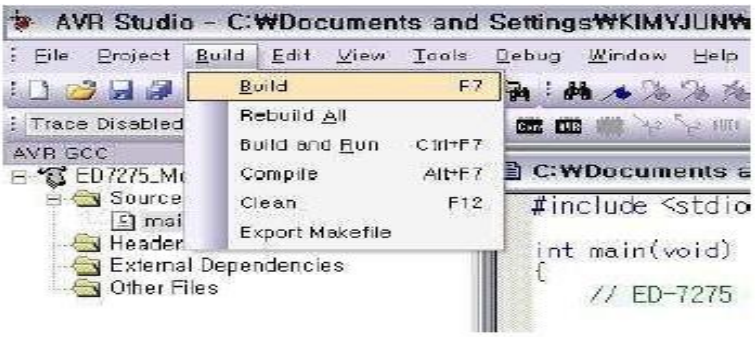


Figure (2.17): Build the program.

Check if Build is completed normally from the following window.

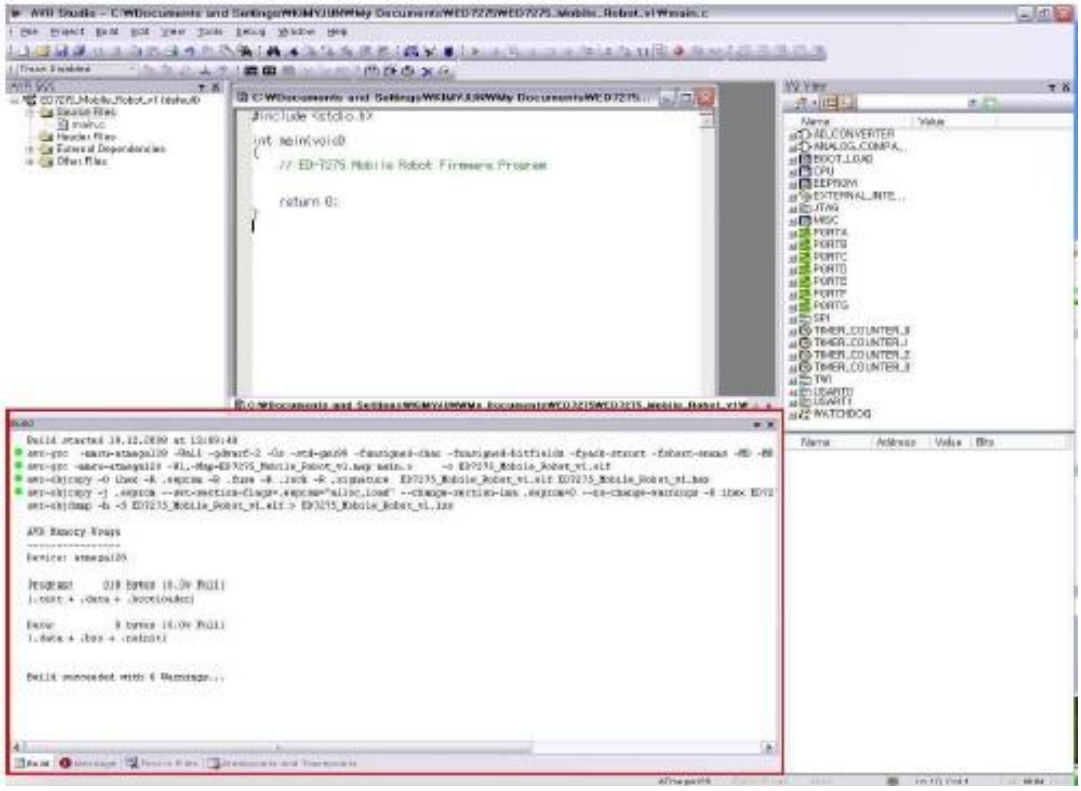


Figure (2.18): Red Line is (Output Window).

When Build is done normally, let's check a Hex file created correspondingly. Check if there is an ED7275_Mobile_Robot_v1.hex file from a default folder in a folder in which there is a created project.

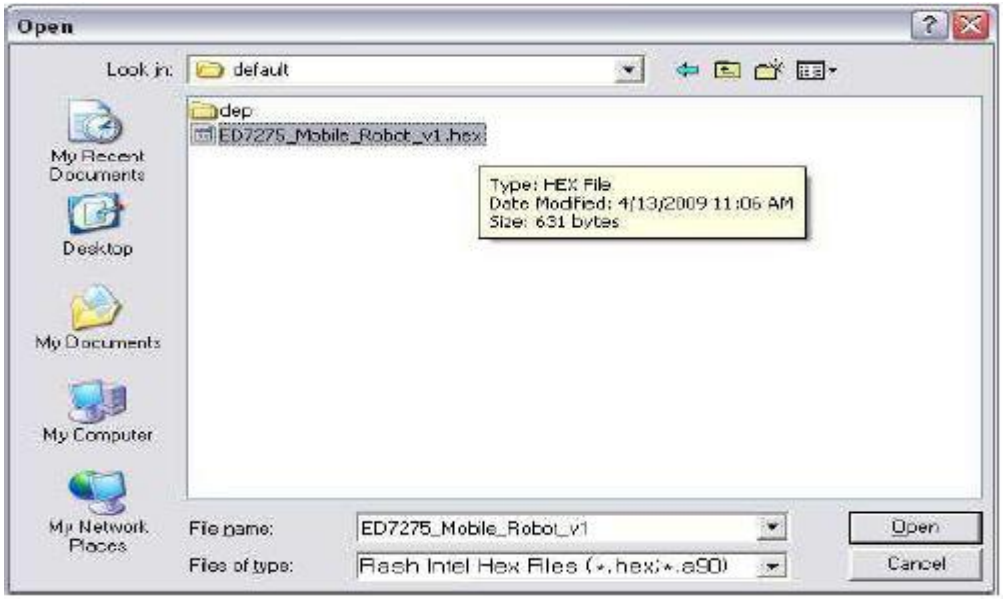


Figure (2.19): *.HEX File Code.

4- download the HEX file created to a robot.

connect the USB AVR ISP to PC through an USB, turn on the robot power, and connect the 6 pin connector of USB AVR ISP to an ISP terminal of the robot.

Figure (2.20): USB AVR ISP for robot.

Then, open the AVR Studio 4 window again, select an icon as follows or click the Menu->Tools->Program AVR->Auto Connect.

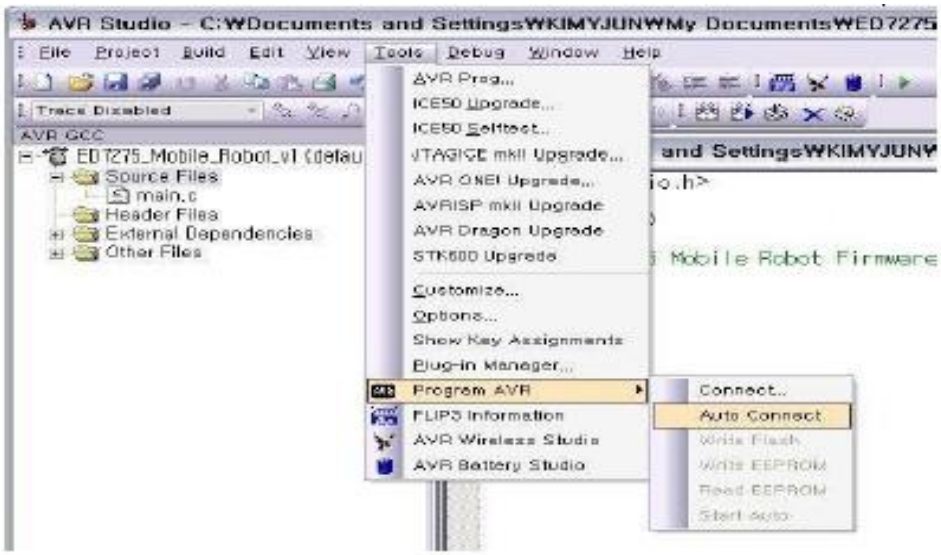


Figure (2.21): Transfer the program to robot.

Select "STK500 or AVRISP" for Platform: and select a COM port number, which was changed when setting the USB AVR ISP for Port:. Then, click the Connect button.

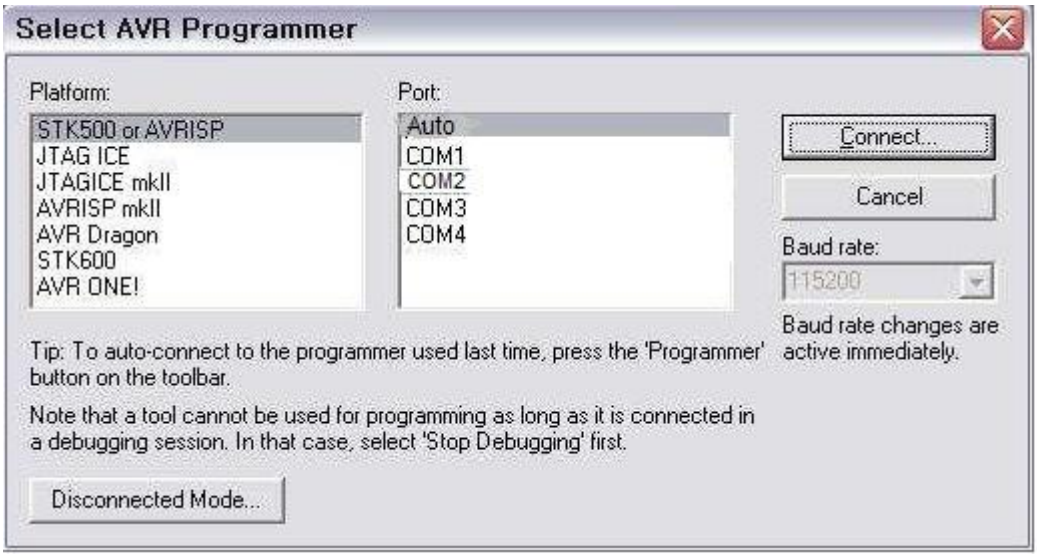


Figure (2.22): An ISP selection window for downloading is displayed.

When connected normally, the following window appears. Here, make sure that you select the "Cancel" button.

Figure (2.23):Message window of AVR studio4 program.

If the message above does not appear, check the USB AVR ISP connection again.

Open a No device selected dialog box of the Device and Signature Bytes group from the following window, and select ATmega128.

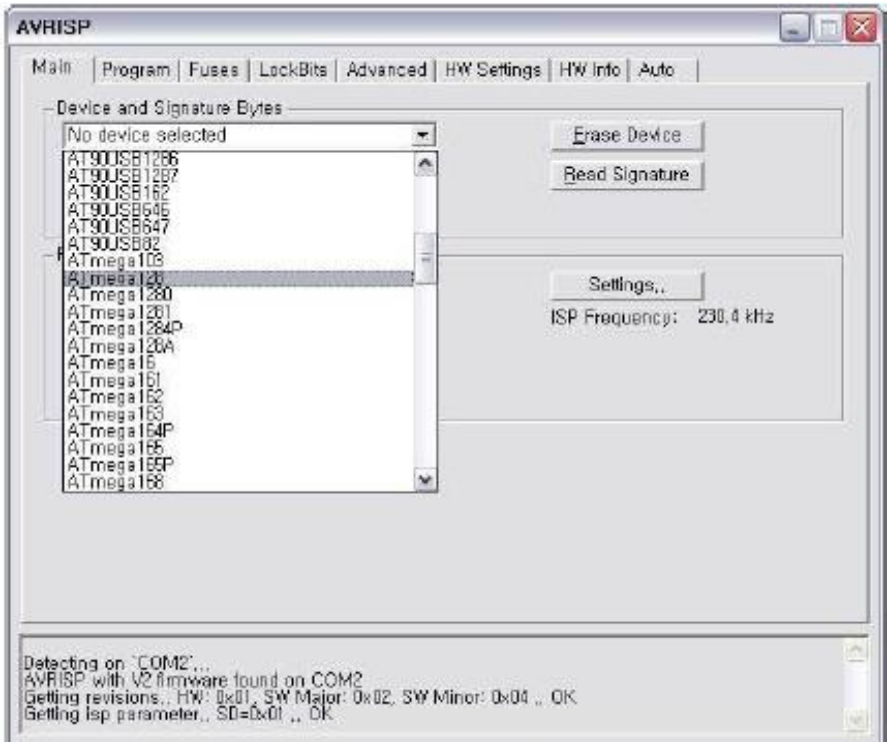


Figure (2.24):AVR ISP Window.

Select the Program of the second tab to download a HEX file.

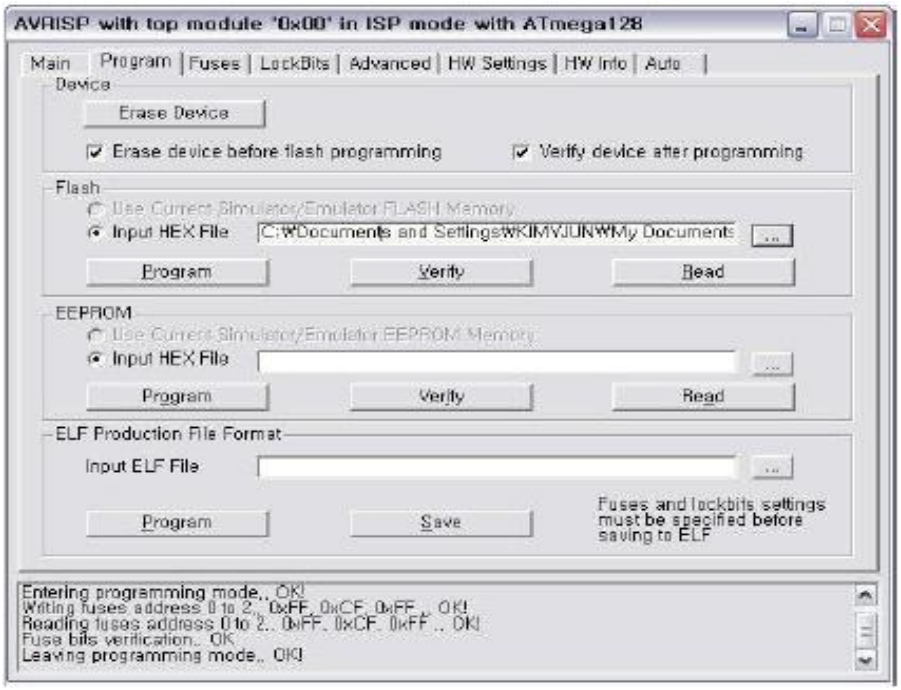


Figure (2.25): AVR ISP Window.

Click the "..." button in the right side of the Input HEX File of the Flash ROM, select the ED7275_Mobile_Robot_v1.hex file, and click the Open Button [3].

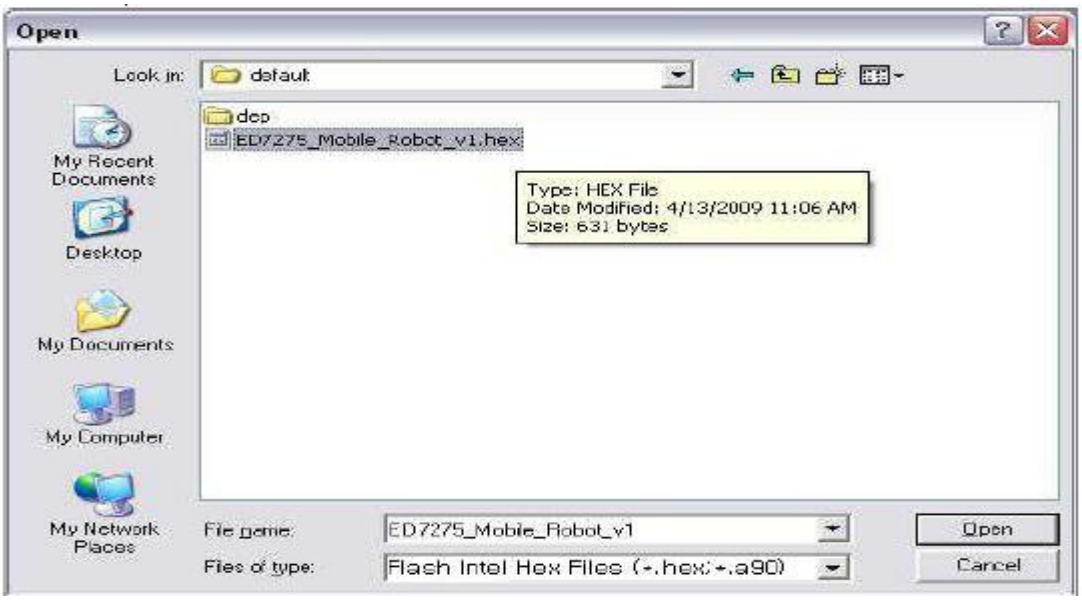


Figure (2.26): Chose *.HEX File.

Chapter Three

Results & Discussion

CHAPTER 3

RESULTS & DISCUSSION

3.1 Introduction:

This chapter is divided into three parts; the first section presents the programming results of the proposed design using AVR Studio 4 C language for the purpose of implementation on ATMega128 chip. The second part presents the custom design results, this part is very necessary to ensure the success of the hardware design. The last part present the Experiments conducted on robot using the parts of robot such as DC-motor, IR sensing and PSD-sensor.

3.2 Programming of the Design Code:

AVR Studio 4 C language was used as a programming language in this project; this language is easy to use and is applicable for the purpose of implementation on the ATMega128 controller. Advantages of C language in AVR Studio 4 are **:-**

1.C provide powerful functions and exhibit flexibility.
2.C shows an excellent portability.
3.C provides an easy program configuration by using only some words called keyword.
4.C is based on a module.

3.3 Custom design Results:

This section shows the work we have done for the success of the project work **:-**

- Field
  - 405cm x 405cm
  - Floor & wall color : White
  - Blocks for configuring a game path
  - 16 basic blocks (for 30cm)
  - 16 basic blocks (for 50cm)
  - Block specifications
    - Material: Wood or plastic or corks
    - Color: White (dove color)
    - (Same color as a field floor)

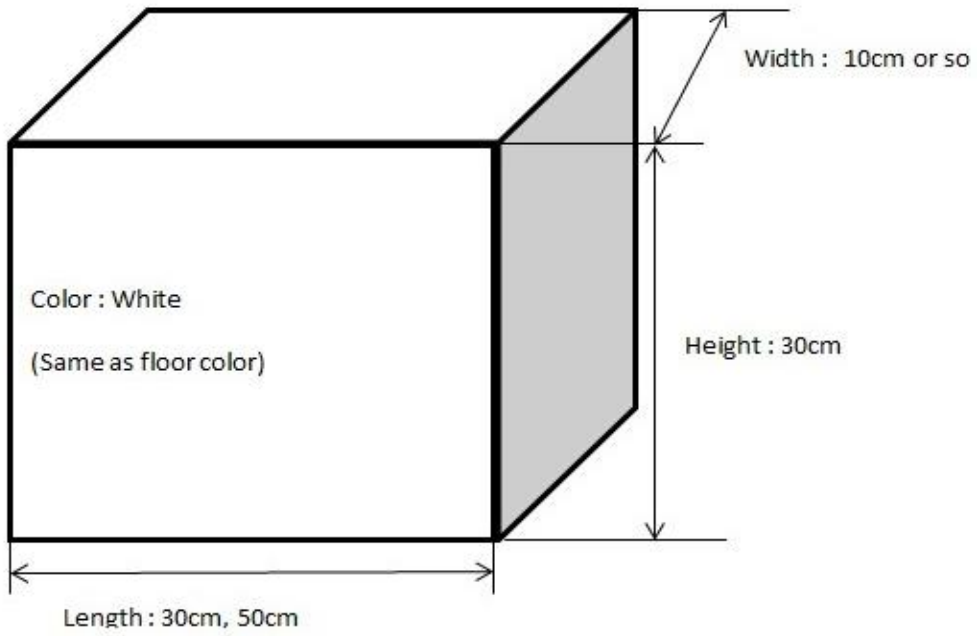
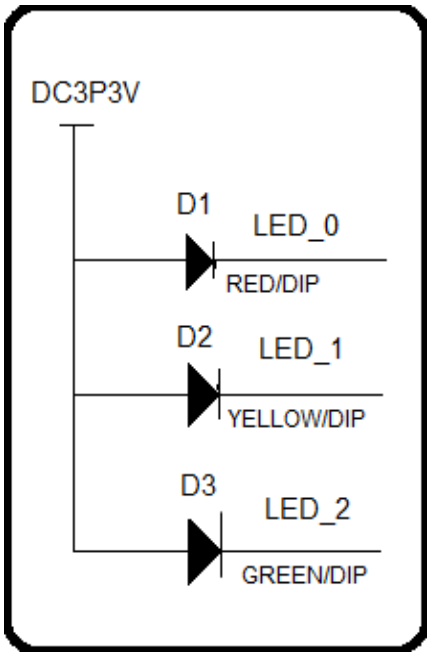


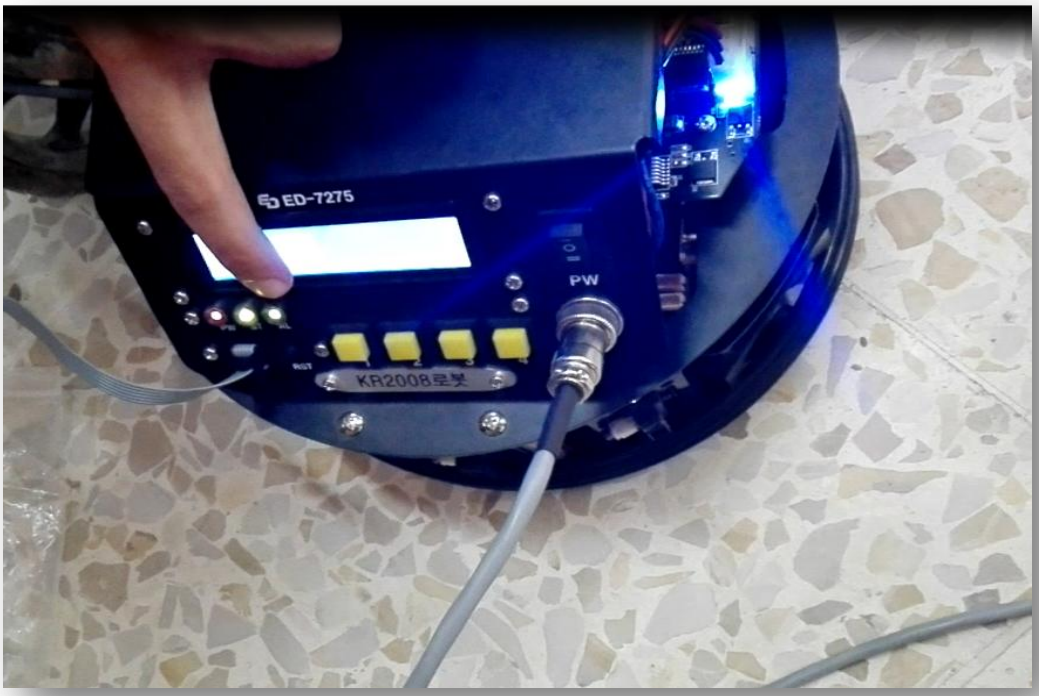
Figure (3.1): Blocks for configuring a game path.

3.4 Operate the led in robot:

In this experiment, we will explain operate the led in the robot:-

- Open the AVR and select new project.
- Select the W0.Device_Example_SourceW1.LED folder from sample program provided and open an example project.
- When project is open normally .user can check the following source that control the LEDs .
- Then select Build to run the program and check the program if contain any error
- Load the program to the ED-7275 [source destination most peripheral devices of ED-7275 are connected by external memory interface ,which is available by connecting a memory or device.

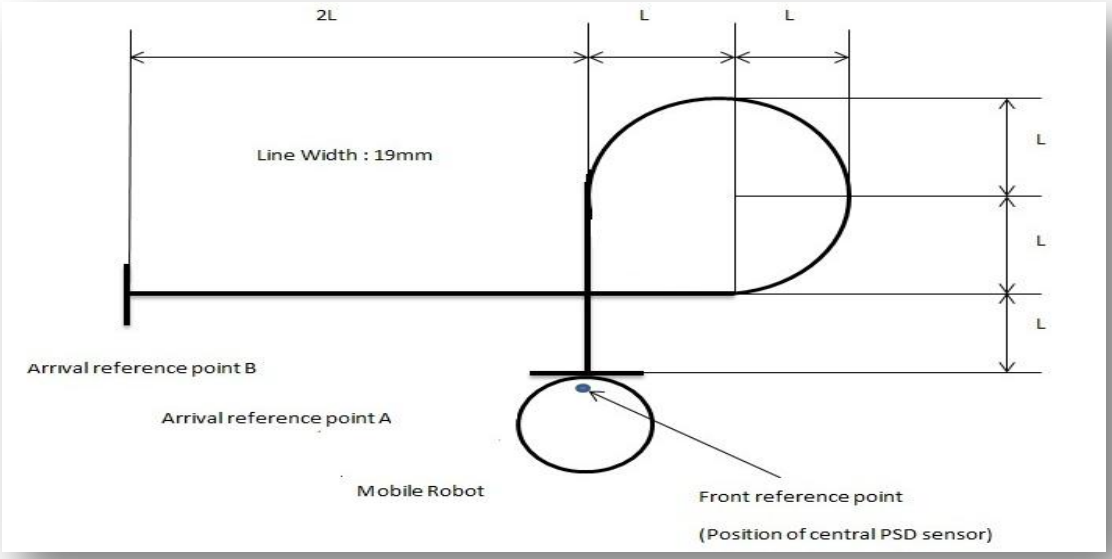




Fig(3.2) Example Of Task No.1.

3.5 The movement of the robot at a certain angle:

Participants must make a program in which a mobile robot starts from a starting line and follows a given path, mixed with other straight lines and curves. The no.2 task sets a goal at accomplishing a fast and smooth driving with information of a sensor. Moreover, its sensor par must stop at a stop line accurately. A length L is 50cm in the figure, finally speed of robot is 16, and this experiment codes found in Appendix A, This experiment, which was applied to the robot located inside the disc and the name of experience is 2.



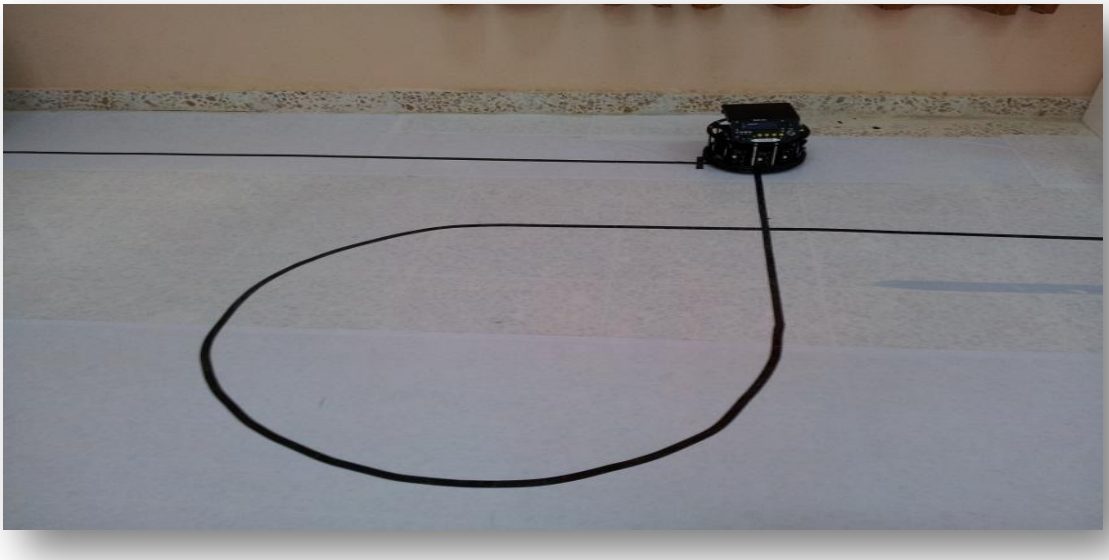


Figure (3.4): Example of Task No.2.

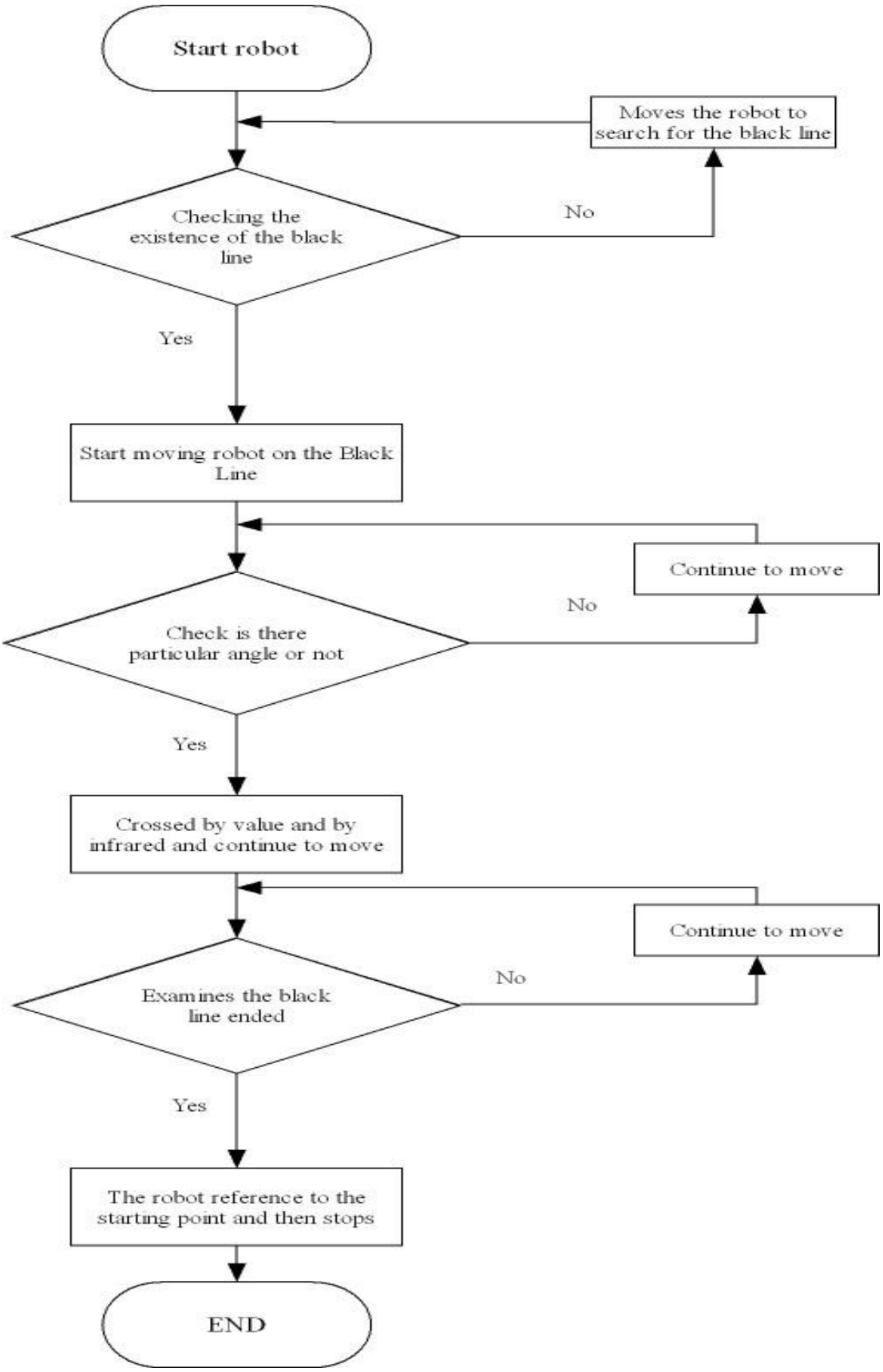


Figure (3.5): Flowchart of move the robot at a certain angle.

3.6 Avoid the Obstacles:

A mobile robot starts from a starting point A, drives along a path of walls, and stops at an arrival reference point B. A given path consists of both sides of a straight line and a gray wall of an arrival point. It sets a goal at driving the robot to its destination based on information of the PSD sensor without colliding with the wall where the arrival point is installed, and stopping the robot at the arrival reference point as close as possible. At this time, the mobile robot must use information of the PSD sensor. The block field consists of gray wall and its length L1 is 210cm, width L2 is 60cm and height L3 is 30cm. And its distance L4 from the wall is 15cm. Another task for mobile robot drives along a path of track and wall in the order of a, b, c and d, and stops at an arrival point B. A block path in a field consists of gray wall and its length L is 60cm and height is 30cm as shown in the picture. As for evaluation of mobile robot operations, a time during which the mobile robot stops in the arrival point B is measured and the robot is given a mark accordingly. Also, the robot is given a mark according to how accurately it arrives at the arrival point D. When the robot collides with the wall during its driving as a whole, it is given a demerit.

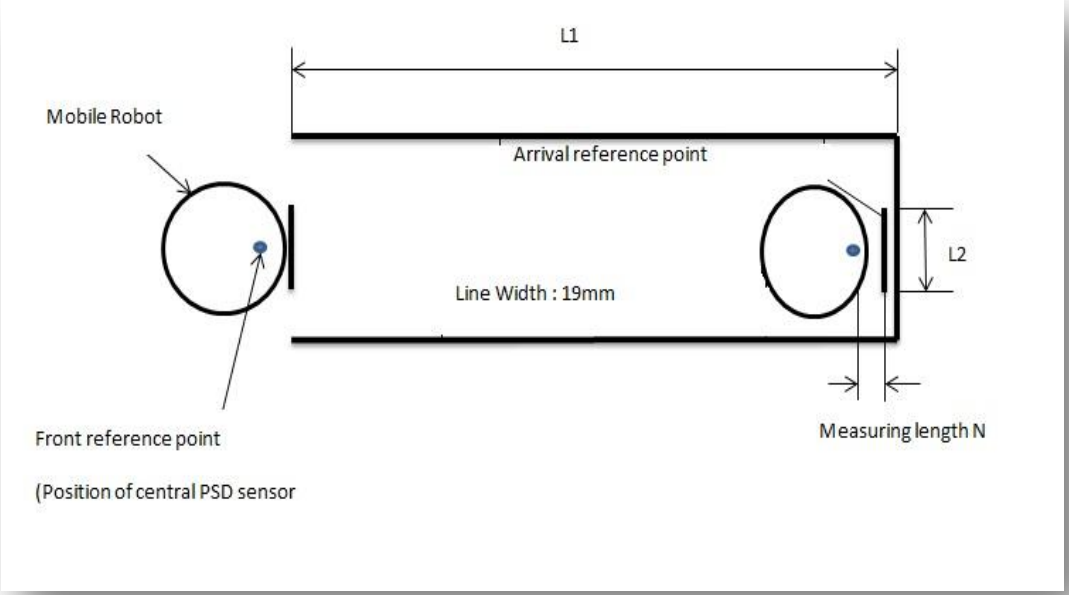


Figure (3.6): Example of task No.3.

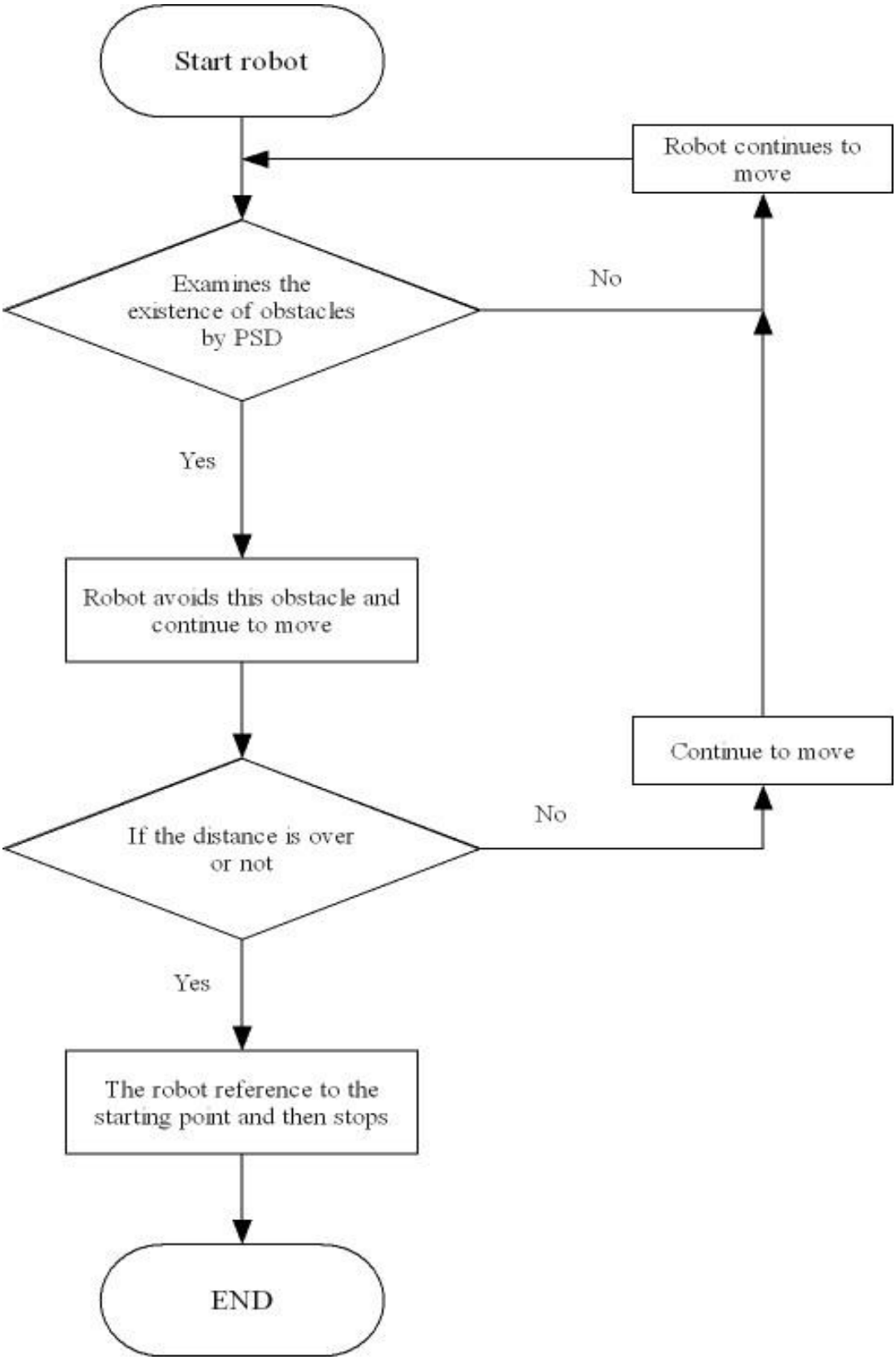


Figure (3.7): Flowchart of robot avoid obstacles.

3.7 Discussion:

Discussion here revolves around the measurements and the problems that confront us each section of the tests:

1- The operate of led is based on the AVR but in C++ language each led in the program is determine duration of time wherever the Red is ox10, the Yellow is ox20 and Green is ox40 than the program is loaded to the ED-7275 .

2- The movement of the robot at a certain angle:-

Here robot moves at an angle and the measurements used for this case speed is 16 and the length of the line is 370 cm and also faced the same problem in the movement of the robot on a straight line, as well as uses the same solution to get rid of the problem, as well as adjust the angle is a semi-circle and measured 30 cm.

3- Avoid the obstacles:-

Avoid obstacles encountered a problem and are using cork material that does not reflect the PSD signal only absorbs reference and including the cork material inside the studios are used to absorb the echo so it did not work this article and to solve the problem using wood material.

Chapter Four
Conclusion and Future Work

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusions:

From the design and simulation results, it can be concluded that Higher execution speed versus small chip size is achieved by designing the proposed circuit with ATMega128-controller. Higher flexibility versus acceptable accuracy is also achieved by designing the proposed system using AVR Studio C language. Sufficient design accuracy can be achieved with AVR Studio C in particular. It is very convenient system for consumers and has extensible and flexible characteristics. It can also be used for cleaning home. Therefore, it can be used in several areas such as explosives and other examination and to carry objects.

4.2 Suggestions for Future Work:

The work in the field of robot is evolving day by day and in this project will be suggested to:-

1- The first suggestion is to add an arm to the robot to move things from one place to another.

2- The second proposal is to solve the problem to avoid the obstacles that we faced, and the use of other material instead of cork, for example, wood material.

REFRENCES:

[1] Kim Young Jun (2008), "Training on ED-7275 Mobile Robotics 2008", ED corporation, vol. 110, no. 22, pp. 38-143.

[2] F.G. Pin, S.M. Killough (1994), "A new family of omnidirectional and holonomic wheeled platforms for mobile robots", IEEE Transactions on Robotics and Automation, vol. 2, no. 10, pp. 480–489.

[3] M.-J. Jung, H.-S. Kim, S. Kim, J.-H. Kim (2000), "Omni-directional mobile base OK-II", in: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3449–3454.

[4] [Microrobo Companies], "3wd-triangular robot", http://www.microrobo.com/3wd-triangular-100mm-omni-wheel-mobile-robotics-car-c003.html [accessed on 1 May 2013 at 05:00 PM].

[5] [Futurelec Companies], " atmel ATMega128 ", http://www.futurlec.com/ATMEGA_Controller.shtml [accessed on 1 May 2013 at 08:12 PM].

[6] [CS.Washington Web page], "AVR Studio 4 C Programming language", http://www.cs.washington.edu/education/courses/cse466/07wi/labs/l1/Debugging%20Tutorial.htm [accessed on 20 Apr 2013 at 09:00 PM].

APPENDIX A

Programming parts of robot:

1. LED

```
// ED-7275 Front LED Operation Example
// led.c
//---------------------------------------
#include<avr/io.h>
#include<util/delay.h>

// External Memory Address Define
#define LED_OUT_EN                    (*(volatile unsigned char *)0x2300)

// Operate LED on the front panel
#define RLED        0x10    // Red LED (PW) -> 0xEF ( 1110 1111 )
#define YLED        0x20    // Yellow LED (ST) -> 0xDF ( 1101 1111 )
#define GLED        0x40    // Green LED (AL) -> 0xBF ( 1011 1111 )

void LED_SW(unsigned char led)
{
        // Turn on the corresponding LED.
        // The upper 3bit is LED 0110, and the lower 4bit (0~3) is KEY PULLUP.


LED_OUT_EN = ~(led);
}

// test led
int main(void)
{
        // Set an external memory
   MCUCR = 0x80;                    // enable external memory and I/O
   XMCRA = 0x44;            // 0x1100-0x7FFF=1 wait, 0x8000-0xFFFF=0 wait
   XMCRB = 0x80;                    // enable bus keeper, use PC0-PC7 as address

        LED_SW(0); // LED OFF

        while(1)        // main loop
        {
                LED_SW(RLED);
                _delay_ms(1000);
                LED_SW(YLED);
                _delay_ms(1000);
                LED_SW(GLED);
                _delay_ms(1000);
                LED_SW(RLED);
```

```
                _delay_ms(1000);
                LED_SW(RLED|YLED);
                _delay_ms(1000);
                LED_SW(RLED|YLED|GLED);
                _delay_ms(1000);
                LED_SW(0); // LED OFF
                _delay_ms(1000);
                LED_SW(RLED|YLED|GLED);
                _delay_ms(1000);
                LED_SW(0); // LED OFF
                _delay_ms(1000);
}
}
```

Explain for led codes:-

```
MCUCR = 0x80;
XMCRA = 0x44;
XMCRB = 0x80;
```

The sources above are to set a register to use an external memory interface. The three registers above are described in detail in datasheet of ATMega128.

```
#define RLED        0x10    // Red LED (PW) -> 0xEF ( 1110 1111 )
#define YLED        0x20    // Yellow LED (ST) -> 0xDF ( 1101 1111 )
#define GLED        0x40    // Green LED (AL) -> 0xBF ( 1011 1111 )

void LED_SW(unsigned char led)
{
        LED_OUT_EN = ~(led);
}

…
LED_SW(RLED);
_delay_ms(1000);
LED_SW(YLED);
_delay_ms(1000);
LED_SW(GLED);
_delay_ms(1000);
```

The LED_SW function is designed to control LED easily by entering the RLED**,** YLED and GLED declared above as factors. Try for all the LED values to be turned on. In case of turning off all the LEDs, enter 0.

```
LED_SW(RLED|YLED|GLED);
_delay_ms(1000);
LED_SW(0); // ALL LED OFF
_delay_ ms(1000);
```

The _delay_ms( ); function is to generate delay in milliseconds unit when #include<util/delay.h> is declared and frequency is set to 8000000 hz **in** Project → Configuration Options in Menu**.** In 1000ms, it makes a waiting state for 1 second.

2. KEY

```
// ED-7275 Front KEY Operation Example
// key.c
//---------------------------------------
#include<avr/io.h>
#include "led.h"          // Import led.h.
// External Memory Address Define
#define KEY_IN_EN                    (*(volatile unsigned char *)0x2200)

// key
#define KEY0                 0x0E // 0000 1110
#define KEY1                 0x0D // 0000 1101
#define KEY2                 0x0B // 0000 1011
#define KEY3                 0x07 // 0000 0111

unsigned char key_in(void)
{
        // KEY uses the lower 4bit (0~3) only.
        // Disuse an upper bit through AND operation for 0x0F and return it.
        return (KEY_IN_EN & 0x0F);
}
// t` char key;
// Set an external memory
```

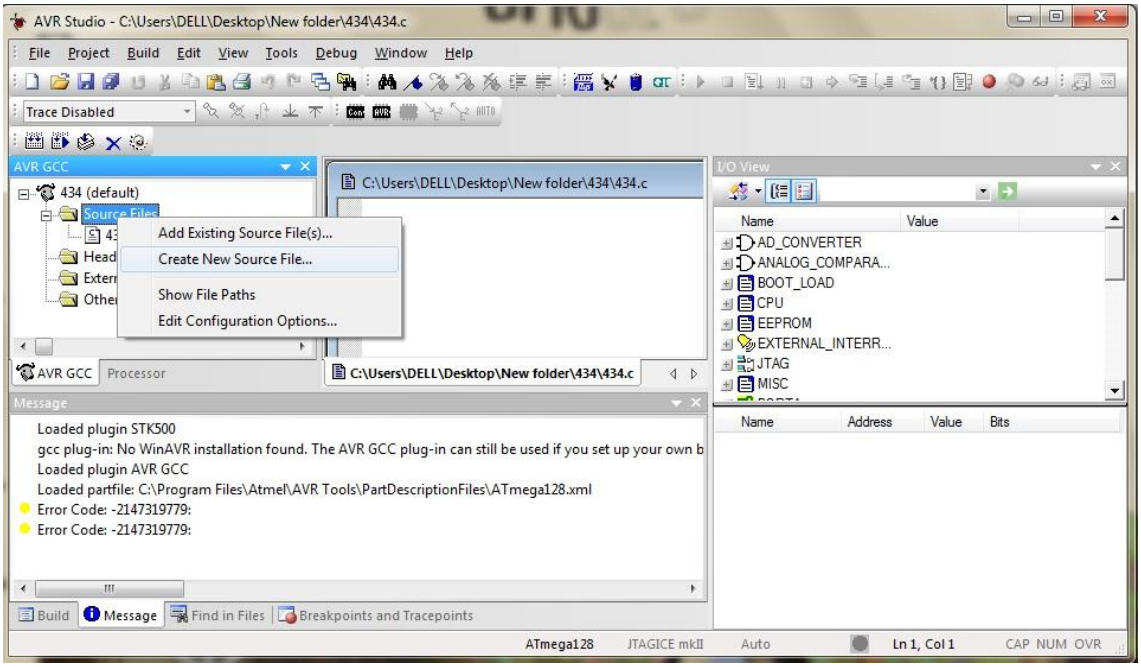
```
MCUCR = 0x80;                    // enable external memory and I/O
XMCRA = 0x44;                    // 0x1100-0x7FFF=1 wait, 0x8000-0xFFFF=0
wait
XMCRB = 0x80;                    // enable bus keeper, use PC0-PC7 as address

        LED_SW(0);  // Turn off first LEDs all. Function is called by led.h.

        // for an unlimited loop
        while(1)
        {
                key = key_in();
                switch(key)
                {
                case KEY0:                  // 0x0E -> 0000 1110
                        LED_SW(RLED);
                        break;
                case KEY1:                  // 0x0D -> 0000 1101
                        LED_SW(YLED);
                        break;
                case KEY2:                  // 0x0B -> 0000 1011
                        LED_SW(GLED);
                        break;
                case KEY3:                  // 0x07 -> 0000 0111
                        LED_SW(RLED | YLED | GLED);
                        break;
                default:
                        LED_SW(0);          // LED ALL OFF
                }
        }
}
```

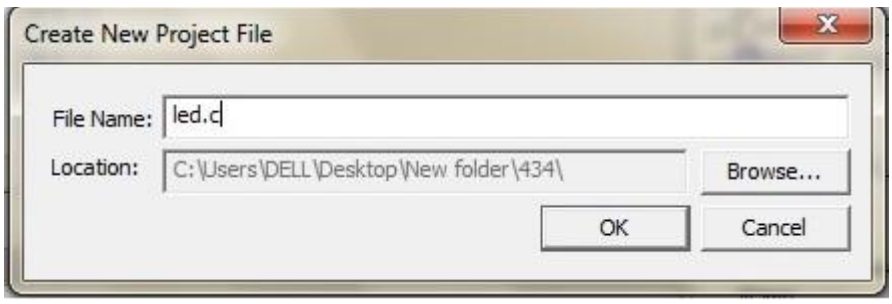
Explanation for key codes:-

#include "led.h" // Import led.h.

Make and include led.c and led.h so that the led.c source used in the 1.LED ex`ample can be also used in a KEY example.

In case of importing a user-created header file #include, write its name in " ". A source and header file can be added through a project tree in the left side. If you click the right button of the mouse at the source files at the project tree, you can select and create new source file… as shows below:





```
 key = key_in();
                switch(key)
                {
                case KEY0:                  // 0x0E -> 0000 1110
                        LED_SW(RLED);
                        break;
                case KEY1:                  // 0x0D -> 0000 1101
                        LED_SW(YLED);
                        break;
                case KEY2:                  // 0x0B -> 0000 1011
                        LED_SW(GLED);
                        break;
                case KEY3:                  // 0x07 -> 0000 0111
                        LED_SW(RLED | YLED | GLED);
                        break;
                default:
                        LED_SW(0);          // LED ALL OFF
                }
        }
}
```

It receives a key value in key_in() and carries out execution according to its corresponding key input through a switch statement.

3. IR sensor

```
// ED-7275 IR Operation Example
// ir.c
//--------------------------------------
#include<avr/io.h>
#include<util/delay.h>
#include "lcd.h"
```

```
// External Memory Address Define
#define IR_IN_EN              (*(volatile unsigned char *)0x2500)     // IR In

void ir_init(void)
{
        // IR Sensor Power ON
        DDRE |= (1<<3);
        PORTE |= (1<<3);              // IR power on
}

unsigned char ir_read(void)
{
        return (IR_IN_EN & 0x1F);
}



// test ir sensor
int main(void)
{
        unsigned char ir;

        // External memory setting
 MCUCR = 0x80;                        // enable external memory and I/O
XMCRA = 0x44;                         // 0x1100-0x7FFF=1 wait, 0x8000-0xFFFF=0
wait
 XMCRB = 0x80;                        // enable bus keeper, use PC0-PC7 as address

        lcd_init();
        lcd_string(LCD_LINE1, "IR SENSOR");

        ir_init();

        while(1)
        {
                lcd_setpos(LCD_LINE2);
```

```
                ir = ir_read();

                // ir sensor 0
                if((ir & 0x01) == 0)
                {
                        lcd_data('L');
                }else{
                        lcd_data('H');
                }
                lcd_data(' ');

                // ir sensor 1
                if(((ir>>1) & 0x01) == 0)
                {
                        lcd_data('L');
                }else{
                        lcd_data('H');
                }
                lcd_data(' ');

                // ir sensor 2
                if(((ir>>2) & 0x01) == 0)
                {
                        lcd_data('L');
                }else{
                        lcd_data('H');
                }
                lcd_data(' ');

                // ir sensor 3
                if(((ir>>3) & 0x01) == 0)
                {
                        lcd_data('L');
                }
else{
```

```
                lcd_data('H');
        }
        lcd_data(' ');
        // ir sensor 4
        if(((ir>>4) & 0x01) == 0)
        {
                lcd_data('L');
        }else{
                lcd_data('H');
        }

        _delay_ms(100);
    }
}
```

Explain for ir sensor codes:-

```
#define IR_IN_EN            (*(volatile unsigned char *)0x2500)     // IR In
```

Declare an expanded memory address value of the circuit IR sensor.

```
void ir_init(void)
{
    // IR Sensor Power ON
    DDRE |= (1<<3);
    PORTE |= (1<<3);            // IR power on
}
```

The ir_int() function makes the IR sensor run by setting the bit3 of PORTE as high and turning on the light-emitting sensor of IR sensor.

```
unsigned char ir_read(void)
```

```
{
        return (IR_IN_EN & 0x1F);
}
```

The ir_read() function reads and return the bit0~4 status values of IR_IN_EN when the light-emitting sensor turns on through the ir_init() function.

```
lcd_setpos(LCD_LINE2);
                ir = ir_read();

                // ir sensor 0
                if((ir & 0x01) == 0)
                {
                        lcd_data('L');
                }else{
                        lcd_data('H');
                }
```

The source above is an example of reading a sensor value, checking bit0~4, including ;L; on the lcd in order its corresponding bit 0, and including ;H; if its corresponding bit is 1.

Finally connection sequence between robot and AVR ISP:-

1.connect an USB of AVR ISP to PC.
2.Turn on the robot power.
3.connect the 6pin cable of USB AVR to the ISP terminal or the robot.
4.download the program to the robot using AVR Studio.
5.Check the robot operations.

الخلاصة

هذا المشروع يقدم تنفيذ نظام روبوت باستخدام مسيطر من نوع atmega128. الأنسان الألي تم تصميمه وتنفيذه باستخدام مسيطر دقيق با لإضافة الى شاشة عرض من نوع كريستال و محرك تيار مستمر و مستشعر لتحديد المسافة و كذلك مستشعر اشعة تحت الحمراء . و هذا التصميم سيكون قابل للتطبيق في كثير من المجالات العملية لما يحمله من مرونة في التصميم و كذلك يمكن تطويره مثال لذلك اضافة ذراع الية و جعله مساعد لنقل الأشياء من مكان الى اخر.

وزارة التعليم العالي والبحث العلمي
جامعة ديالى
كلية الهندسة
قسم هندسة الحاسوب و البرمجيات

برمجة و تشغيل الروبوت المتعدد الاتجاهات المتنقل

مشروع مقدم الى قسم هندسة الحاسبات والبرمجيات
في جامعة ديالى – كلية الهندسة كجزء من متطلبات نيل درجة
البكالوريوس في هندسة الحاسبات والبرمجيات

من قبل

عبير جاسم محمود

زينب خزعل شامل

بإشراف

م.م. حسين سلطان راضي

2016