

# **FIVE SENSOR LINE FOLLOWING WITH OBSTACLE AVOIDANCE ROBOT**

A Mini-Project report submitted to the VELS UNIVERSITY

In the partial fulfillment for the award of degree of

**BACHELOR OF SCIENCE**

**IN**

**COMPUTER SCIENCE**

**By**

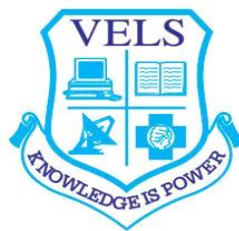
**SHIVRANJAN PRABHAKAR KOLVANKAR**

**REG NO : 09106124**

**UNDER THE GUIDANCE OF**

**Mrs. G. THAILAMBAL, M.C.A., M.Phil**

**Asst. Prof. Department of Computer Science**



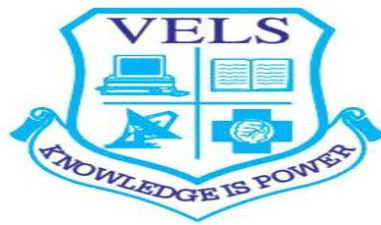
**VELS UNIVERSITY**

**SCHOOL OF COMPUTING SCIENCES**

**DEPARTMENT OF COMPUTER SCIENCE**

**APRIL 2012**

**VELS UNIVERSITY**  
**SCHOOL OF COMPUTING SCIENCES**  
**DEPARTMENT OF COMPUTER SCIENCE**



This is to certify that the project entitled **“FIVE SENSOR LINE FOLLOWING WITH OBSTACLE AVOIDANCE ROBOT”** being submitted to the **VELS UNIVERSITY** by **SHIVRANJAN P. KOLVANKAR** bearing **Reg No: 09106124**, for the partial fulfillment of award of degree in Bachelors of Science in Computer Science, is a bonafide work record work carried out by him under my guidance and supervision.

**INTERNAL GUIDE**

**HEAD OF THE DEPARTMENT**

Submitted for the viva-voice examination held on ..... at Vels University,  
Pallavaram

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# DECLARATION

I hereby declare that the project entitled “**FIVE SENSOR LINE FOLLOWER ROBOT WITH OBSTACLE AVOIDANCE**” is the bonafide work of mine and is not submitted to any other University/Institution before for any other degree.

SHIVRANJAN KOLVANKAR

## ACKNOWLEDGEMENT

I deeply wish to express my sincere thanks to **Dr. ISHARI K. GANESH, M.Com, B.L., Ph.D.**, Chancellor, **VEL'S UNIVERSITY** and **Mr. S.KAMALA KANNAN**, the Pro-Chancellor, **VEL'S UNIVERSITY** for providing me necessary facilities.

I wish to extend my heart and sincere thanks to **Dr. P.GOVINDRAJAN, M.Sc., M.Phil., Ph.D.**, The Registrar, **VEL'S UNIVERSITY**.

I wish to extend my deep sense of gratitude and sincere thanks to the Head of The Department, **Mr. S. Perumal, M.Sc., M.Phil.**, and also my project guide **Mrs. G. Thailambal, M.C.A., M.Phil.**, Asst. Prof. Department of Computer Science for their guidance in completing this project.

I wish to thank my staff members of the department of Computer Science for their co-operation during the completion of course of my study. It's not only my responsibility but a deep wish, to express my gratitude to my parents and the **ALMIGHTY** in making this project a success. I wish to remember forever help rendered by my friends for their constant encouragement during the study.

# INDEX

| S. No | Title                 | Page No |
|-------|-----------------------|---------|
|       | List of Tables        |         |
|       | List of Figures       |         |
|       | List of Abbreviations |         |
|       | Abstract              |         |
| 1     | Introduction          |         |
|       | 1.1 Objective         |         |
| 2     | System Analysis       |         |
|       | 2.1 Existing System   |         |
|       | 2.1.1 Drawbacks       |         |
|       | 2.2 Proposed System   |         |
|       | 2.2.1 Advantages      |         |

|   |                               |  |
|---|-------------------------------|--|
|   | 2.3 Feasibility Study         |  |
|   | 2.3.1 Economic Feasibility    |  |
|   | 2.3.2 Operational Feasibility |  |
|   | 2.3.3 Economic Feasibility    |  |
| 3 | System Specification          |  |
|   | 3.1 Hardware Specification    |  |
|   | 3.2 Software Specification    |  |
| 4 | Hardware Description          |  |
|   | 4.1 Microcontroller           |  |
|   | 4.2 Features                  |  |
| 5 | Project Description           |  |
|   | 5.1 Modules                   |  |
|   | 5.1.1 Line Follower Concept   |  |
|   | 5.2 Algorithms                |  |

|    |                                   |  |
|----|-----------------------------------|--|
|    | 5.3 Input Design                  |  |
|    | 5.4 Output Design                 |  |
| 6  | System Testing                    |  |
|    | 6.1 Unit Testing                  |  |
|    | 6.2 Integration Testing           |  |
|    | 6.3 Validation Testing            |  |
| 7  | System Implementation             |  |
|    | 7.1 Maintenance                   |  |
| 8  | Appendix                          |  |
|    | 8.1 Source Code                   |  |
|    | 8.2 Screen Shot                   |  |
| 9  | Conclusion and Future Enhancement |  |
|    | 9.1 Conclusion                    |  |
|    | 9.2 Feature Enhancement           |  |
| 10 | References                        |  |

# LIST OF TABLES



# LIST OF TABLES

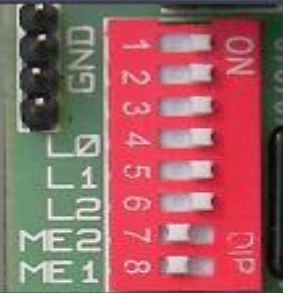
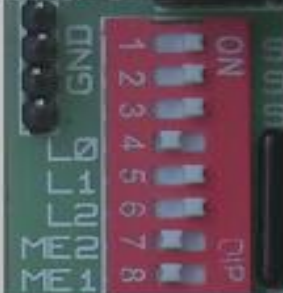
Table 1 Device Specification

| Device     | Program Memory   |                               | Data Memory     |                   | I/O | 10-Bit<br>A/D (ch) | CCP/ECCP<br>(PWM) | SPP | MSSP |                             | EAUSART | Comparators | Timers<br>8/16-Bit |
|------------|------------------|-------------------------------|-----------------|-------------------|-----|--------------------|-------------------|-----|------|-----------------------------|---------|-------------|--------------------|
|            | Flash<br>(bytes) | # Single-Word<br>Instructions | SRAM<br>(bytes) | EEPROM<br>(bytes) |     |                    |                   |     | SPI  | Master<br>I <sup>2</sup> C™ |         |             |                    |
| PIC18F2455 | 24K              | 12288                         | 2048            | 256               | 24  | 10                 | 2/0               | No  | Y    | Y                           | 1       | 2           | 1/3                |
| PIC18F2550 | 32K              | 16384                         | 2048            | 256               | 24  | 10                 | 2/0               | No  | Y    | Y                           | 1       | 2           | 1/3                |
| PIC18F4455 | 24K              | 12288                         | 2048            | 256               | 35  | 13                 | 1/1               | Yes | Y    | Y                           | 1       | 2           | 1/3                |
| PIC18F4550 | 32K              | 16384                         | 2048            | 256               | 35  | 13                 | 1/1               | Yes | Y    | Y                           | 1       | 2           | 1/3                |

Table 2 Operational Specifications

| Features                                 | PIC18F2455              | PIC18F2550              | PIC18F4455              | PIC18F4550              |
|--|-------------------------|-------------------------|-------------------------|-------------------------|
| Operating Frequency                      | DC – 48 MHz             | DC – 48 MHz             | DC – 48 MHz             | DC – 48 MHz             |
| Program Memory (Bytes)                   | 24576                   | 32768                   | 24576                   | 32768                   |
| Program Memory (Instructions)            | 12288                   | 16384                   | 12288                   | 16384                   |
| Data Memory (Bytes)                      | 2048                    | 2048                    | 2048                    | 2048                    |
| Data EEPROM Memory (Bytes)               | 256                     | 256                     | 256                     | 256                     |
| Interrupt Sources                        | 19                      | 19                      | 20                      | 20                      |
| I/O Ports                                | Ports A, B, C, (E)      | Ports A, B, C, (E)      | Ports A, B, C, D, E     | Ports A, B, C, D, E     |
| Timers                                   | 4                       | 4                       | 4                       | 4                       |
| Capture/Compare/PWM Modules              | 2                       | 2                       | 1                       | 1                       |
| Enhanced Capture/<br>Compare/PWM Modules | 0                       | 0                       | 1                       | 1                       |
| Serial Communications                    | MSSP,<br>Enhanced USART | MSSP,<br>Enhanced USART | MSSP,<br>Enhanced USART | MSSP,<br>Enhanced USART |
| Universal Serial Bus (USB)<br>Module     | 1                       | 1                       | 1                       | 1                       |
| Streaming Parallel Port (SPP)            | No                      | No                      | Yes                     | Yes                     |
| 10-Bit Analog-to-Digital Module          | 10 Input Channels       | 10 Input Channels       | 13 Input Channels       | 13 Input Channels       |
| Comparators                              | 2                       | 2                       | 2                       | 2                       |

Table 3 Mode Selection

| L2 | L1 | L0  | S2 | S1 | S0 | Logic to uC (binary) | Mode | Switch position   | Function           |
|----|----|-----|----|----|----|----------------------|------|---|--------------------|
| On | On | On  | On | On | On | 000                  | 0    |  | Line Following     |
| On | On | Off | On | On | On | 001                  | 1    |  | Obstacle avoidance |



# LIST OF FIGURES

# LIST OF FIGURES

Figure 1.1 PIC 18F Pin Diagram

## 40-Pin PDIP

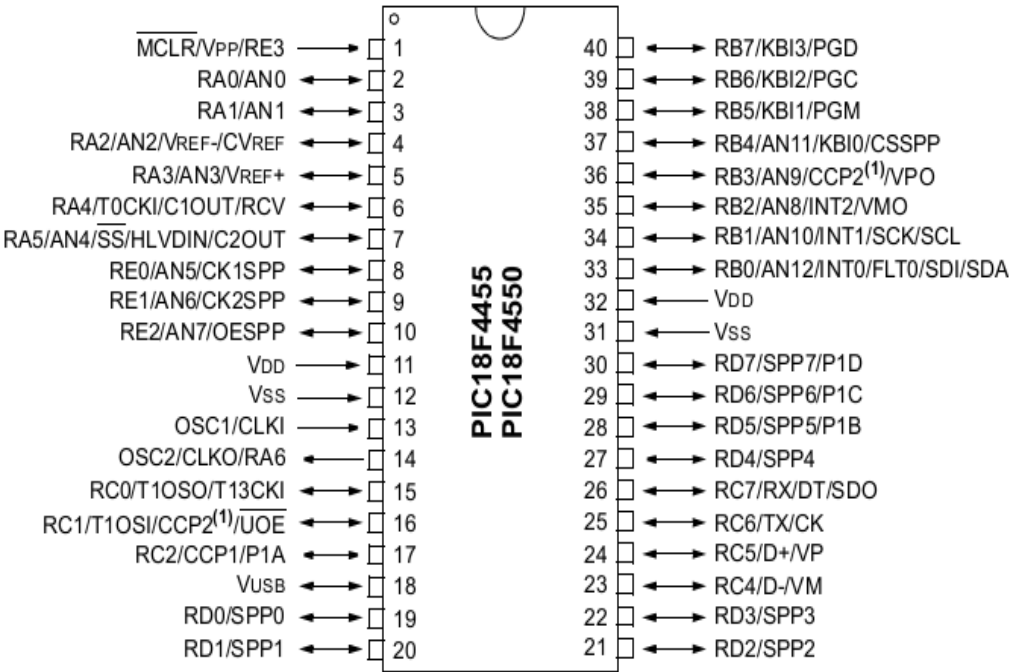


Figure 2 Standard Application

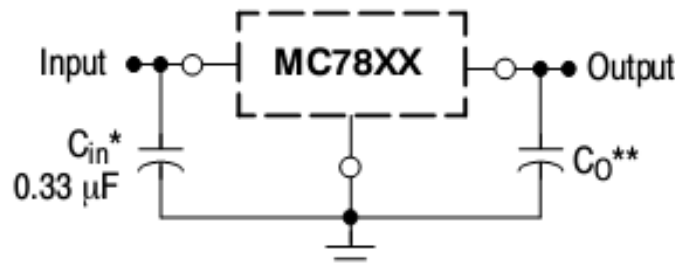
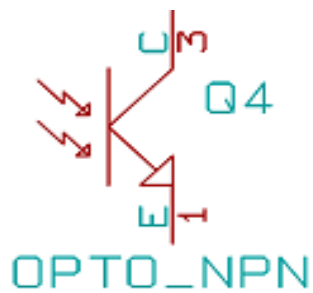


Figure 3 Photo Transistor



**Photo Transistor**

Figure 4 PCB CIRCUIT DESIGN

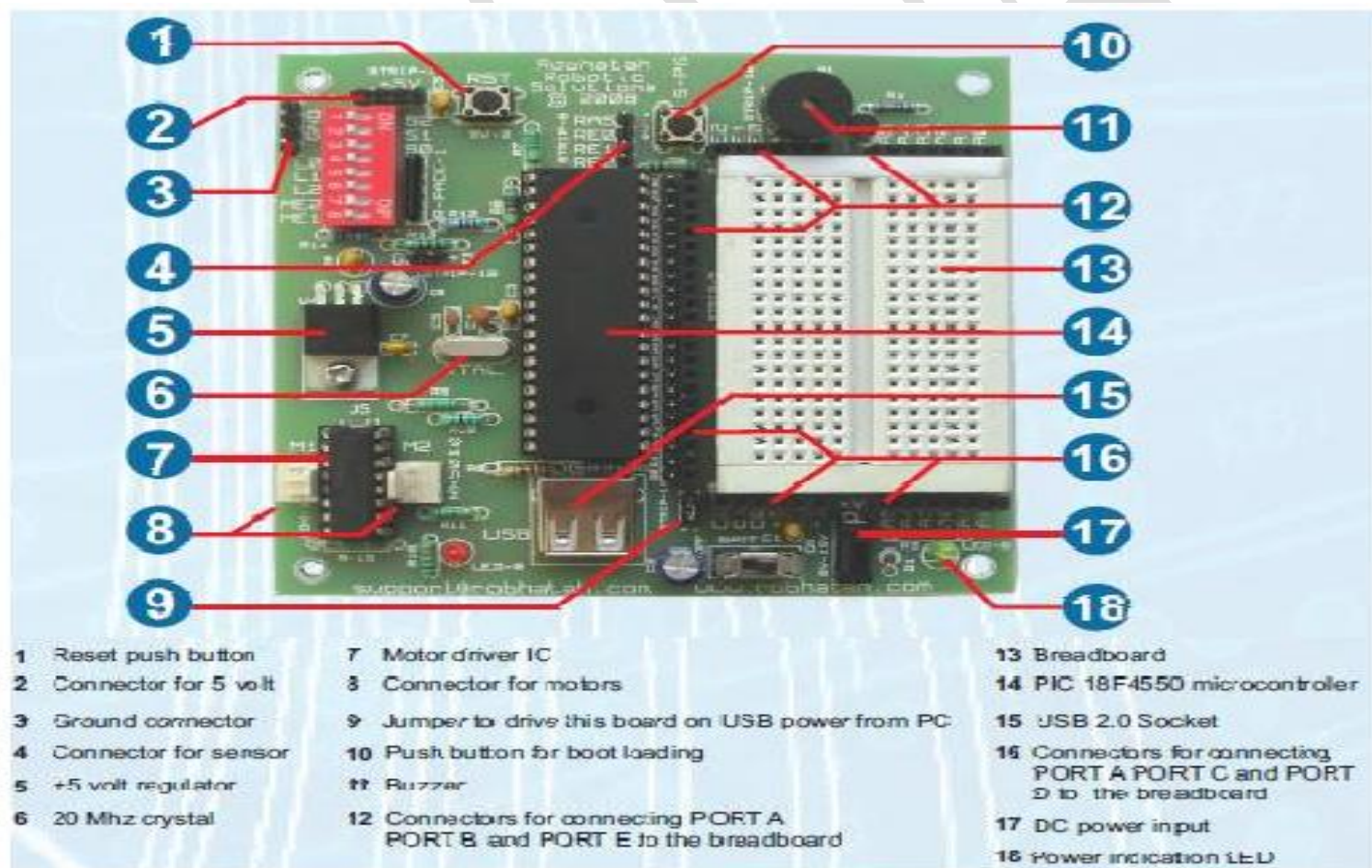
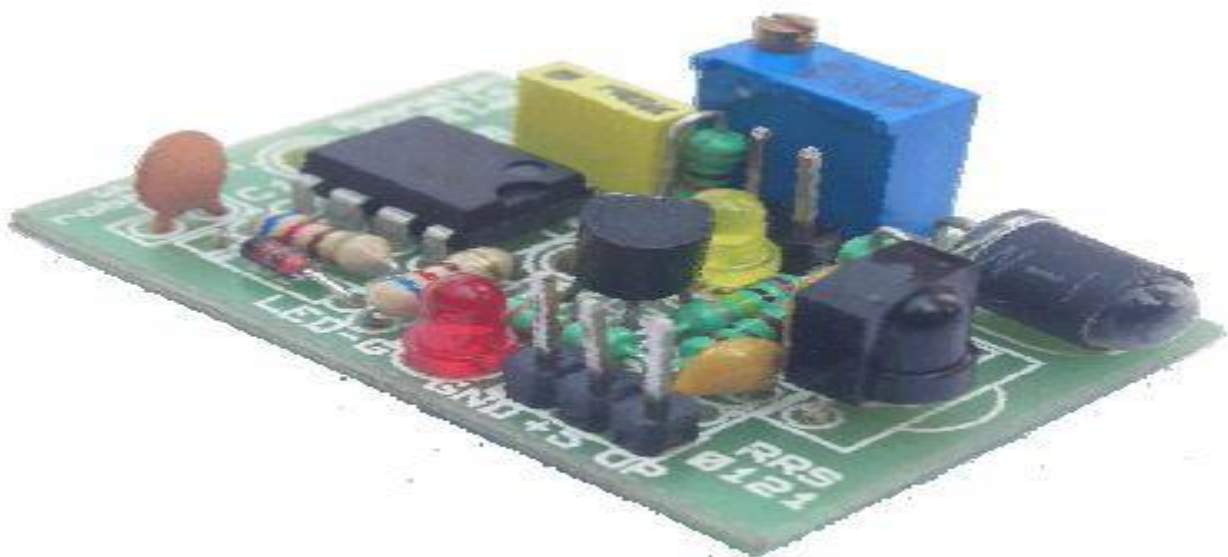


FIGURE 5 SENSOR MODULE





# LIST OF ABBREVIATIONS

## LIST OF ABBREVIATIONS

|                 |   |   |
|-----------------|---|---|
| ❖ PIC           | : | Peripheral Interface Controller                   |
| ❖ IC            | : | Integrated Circuits                               |
| ❖ DC            | : | Direct Current                                    |
| ❖ AC            | : | Analog Current                                    |
| ❖ IEEE          | : | Institute of Electrical and Electronics Engineers |
| ❖ IR            | : | Infrared Sensor                                   |
| ❖ TSOP          | : | Thin Small Outline Package                        |
| ❖ DIP           | : | Dual Inline Package                               |
| ❖ PWM           | : | Pulse Width Modulation                            |
| ❖ Li-Po Battery | : | Lithium-Polymer Battery                           |





# ABSTRACT

# ABSTRACT

A Robot is device which is capable of doing given task either as per programmed or by learning from environment. It has motion, it has manipulation power which is mostly offered to it by using hardware components and assembly language coding to make use of those components. An Autonomous robot is one which can carries certain task with no human intervention. One such basic autonomous wheeled robot is Line Follower Robot.

A line follower is a simple automatic wheeled robot which follows black color line with the help of IR and TSOP sensors. When the light falls on the black color it absorbs the maximum of light. And as maximum absorption takes place there is no or negligible reflection of light. Hence, input to IR is logical '0'. So, when robot falls out of given path i.e. out of black line. It adjusts own position such that it gets back on track. And this motion is controlled automatically. And accordingly motor speed which is used to drive the chassis. Further, some external hardware can be added in order to avoid obstacles as well as edges of the surface.

But, we must learn that although we are trying to implement the motion as of human being the manipulation power of human senses is very high in terms of both efficiency and reflexes. It is needed that more and more sensor modules as well as timer modules must be added to the platform so that it work and react to the environment variables faster. In this project the combination of two different microcontroller families has been carried out and thus the wheeled robot is named to be UNITY in the further documentation.



# INTRODUCTION

# INTRODUCTION

## 1.1 OBJECTIVE

Robotics is focused on developing expert system or machine which can perform with its own intelligence to that of similar to human intelligence. Whenever the need is better accuracy, high efficiency, greater productivity, always intelligent machine comes into play. In the development process of robots and their evolution,

- WHY TO AUTOMATE?
- HOW TO AUTOMATE?
- HOW TO REPRESENT THE KNOWLEDGE ABOUT THE WORLD?

The above mentioned are the most important three questions under consideration. As of now robots or any machine does not possess its own intelligence but in near future an intelligent robot would sit in front of us, will analyze the board and will make the move with the help of his hands and will try to consider our moves also, may not be a dream. Though, yet to overcome difficulties like speech recognition, emotions and expressions, common sense and perception of the situation are there, scientist still considers that A HUMANOID ROBOT is not mere a dream.

## WHY ROBOTICS?

Robotics, AI, machine vision, evolutionary robots, neural networks, game playing, these are some of the fields that are concerned with the development process of intelligent agents. Robots are though developed for human help but still their own intelligence is needed so that these agents can do task more than correctly. And, currently robotic research is in the middle of this evolution. *'Open sourcing of robotic hardware'* or *'generation robots'* is the key term in robotics. Robot ethics is similar to that of the social ethics and they mainly based on Assimov's laws of robotics.

In order to achieve better accuracy, high efficiency, greater productivity robots are useful. They play an important role in Space Exploration, Nuclear plants, Industrial manufacturing etc. Also they are useful in the science research when it is scaled on micro level. The main feature of robots is their stability under various conditions if they are programmed well to handle it. Hence, robots are going to be the part of human life cycle.

DA-VINCI a robot in healthcare can perform operations as per the instructions given to it by Doctor. ASIMO is most advanced Humanoid till date.

## HISTORY OF ROBOTICS:

- The word “Robot” is taken from Czech word meaning ‘forced labour’.
- It was introduced to public by Karel Capek in his play **RUR** in 1920.
- The term “Robotics” was first coined by Isaac Assimov in his story ‘**Liar**’.

## BASIC COMPONENTS OF ROBOTS :

- Power sources
- Actuation
- Structure
- Sensing
- Manipulators
- Locomotion
- Environmental Interactions and Navigation
- Human-Robot Interaction
- Machine vision
- Control



# SYSTEM ANALYSIS

# SYSTEM ANALYSIS

## 2.1 Existing System:

While when we search the existing system for line follower robots it is mostly pure electronics approach in which fundamental electronics circuits are used to build a line follower assembly. The existing system of this project is mostly a rigid and very complex in designing and debugging aspect.

### 2.1.1 Drawbacks :

1. No Machine can endure 100% efficiency whatever may the approach is.
2. With pure electronics approach there comes a possibility of unstable output or unclear path followed by robot.
3. When there is any circuit failure developer has to follow top down approach and have to debug entire system.

## 2.2 Proposed System:

The purpose of the development of the project is to develop a microcontroller based system which will remove the defects in existing one. It follows both electronics and programming approach to achieve the goal. The main objective of this project is to provide the better work efficiency, accuracy, reliability, feasibility. The error occurred could be reduced to nil and working conditions can be improved.

### 2.2.1 Advantages

1. As the programming approach is combined with electronics, the circuit design part is separated from working logic which helps the project development process to focus on one factor at a time.
2. Accuracy and reliability can be enhanced.

## 2.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the organization. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ❖ ECONOMICAL FEASIBILITY
- ❖ TECHNICAL FEASIBILITY
- ❖ SOCIAL FEASIBILITY

### 2.3.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that it can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system is well within the budget and this was achieved because most of the electronic components used which are less in cost thus making significant adjustment to the productivity of the device to a certain extent. If more funds are provided the system can be enhanced in both productivity and efficiency of a device.

### 2.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### 2.3.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that



are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.





# SYSTEM SPECIFICATION

# SYSTEM SPECIFICATION

## 3.1 HARDWARE DESCRIPTION :

|                       |   |                      |
|-----------------------|---|----------------------|
| ○ Processor           | : | Pentium IV           |
| ○ Clock Specification | : | 1.7 GHZ              |
| ○ Memory              | : | 512 MB               |
| ○ Hard Disk Drive     | : | 20 GB                |
| ○ Operating System    | : | WIN XP , LINUX , MAC |
| ○ Microcontroller     | : | PIC 18F 4550         |
| ○ Motor Driver IC     | : | L293D                |

## 3.2 SOFTWARE DESCRIPTION :

- MPLAB IDE (Its freely available on Microchip's website)
- C18 C Compiler (A free student's version is available on the Microchip's website).
- Microchip MCHPFSUSB v1.3 (Its freely available on Microchip's website)

## MORE ABOUT MPLAB IDE :

MPLAB Integrated Development Environment (IDE) is a free, integrated toolset for the development of embedded applications employing Microchip's PIC® and dsPIC® microcontrollers. To create any project we have to create a corresponding workspace. A workspace links up all the associated files required for creating and debugging a project that has embedded software aspects. One can create assembly language programs for Microchip's PIC® and dsPIC® microcontrollers using MPLAB. To create C programs for the same task one has to use the C-18 tool suite along with MPLAB.

## OTHER COMPONENTS :

- Programming Board with PIC18F4550 and Motor Driver IC L293D
- Programming Board with ATMEGA 16 Ardiuno
- A DC power supply (ideally 9V-12V) (In this case the supplied batteries)
- USB cable



# HARDWARE DESCRIPTION

# HARDWARE DESCRIPTION

## 4.1 Microcontroller

The heart of the UNITY is a PIC 18F4550 microcontroller. This is an industrial grade microcontroller manufactured by Microchip technologies Inc. The PIC18F4550 microcontroller has been specifically designed for embedded C programming. The PIC 18F4550 microcontroller also has an integrated full speed USB 2.0 trans-receiver, which has been configured for high speed USB programming of the Revoboard.

## 4.2 Motor and Motor Driver

The UNITY comes with two geared dc motors of 500 rpm. The motors have helical gears for higher efficiency and lower noise. A single L293D motor driver IC drives the motors. The motor driver IC has a current rating of up to 600mA per channel. The purpose of the motor driver IC is to convert the five or 0-volt signal generated by the microcontroller to a level of 12 or 0 volt so that it can power the motor. Had the motors been directly connected to the microcontroller, the voltage and current produce by it will be very low to drive the motor.

## 4.3 IR sensor module

The UNITY comes with four IR sensors. These sensors can be configured as line sensors or obstacle sensors. The sensors have a tuning screw to vary the range of sensing. The sensors require a 5-volt supply voltage and can generate digital output of 5 or 0 volts when functioning properly. The sensor module consists of the following components. Each UNITY educational kit contains four pieces of sensor modules, which can be used to implement the different features, such as

- LINE FOLLOWING,
- OBSTACLE DETECTION

## 4.4 Timer

The sensor module comprises of a 555 timer IC that creates a square wave of 38 kHz and 50% duty cycle. This is essential for the modulating the IR rays emitted by the IR LED. Hence, the IR led is powered by the square wave generated by the 555 timer IC.

#### 4.5 IR transmitter

The transmitter section of the sensor board comprises of an IR LED. This LED in conjunction with the 555 timer IC generates a pulsating IR beam of 38 kHz. The sensor module can sense an obstacle or white surface by detecting if the pulsating IR rays emitted by the IR LED are reflected back into the IR receiver.

#### 4.6 Power Supply

The UNITY consist of an 8\* 1.5 V AA cell bundle. This pack can provide a supply voltage of 12 volts for the UNITY. This battery pack can be easily mounted on the underbody of the chassis. The 7805 voltage regulator on board takes the 12volt as input and generates a regulated 5-volt supply required for the electronic components onboard of microcontroller board. The board can also be powered by drawing power from the USB port during programming and testing of sensor modules. However, this is not suitable for driving the motors.



# PROJECT DESCRIPTION



# PROJECT DESCRIPTION

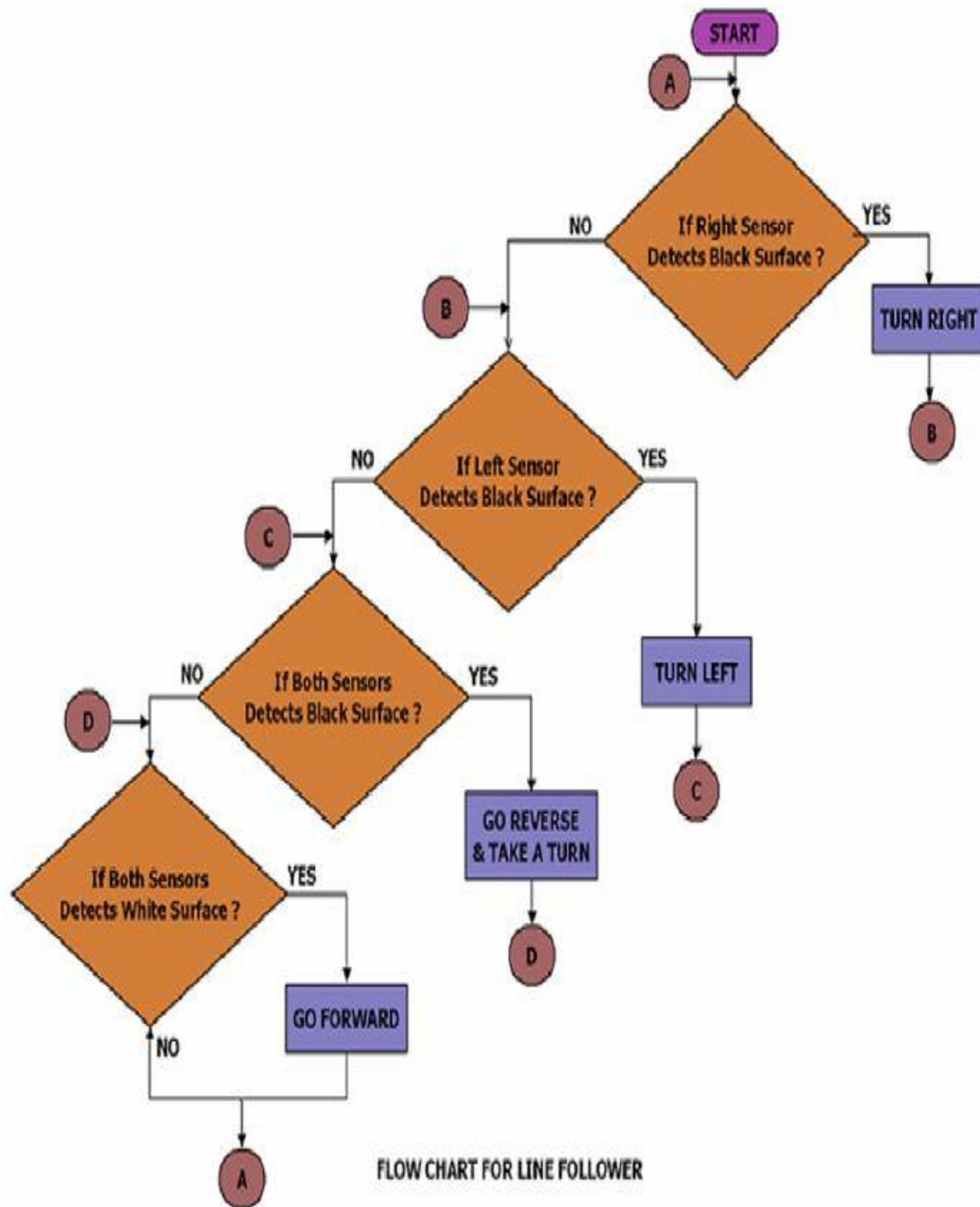
## 5.1 MODULES

### 5.1.1 LINE FOLLOWER CONCEPT :

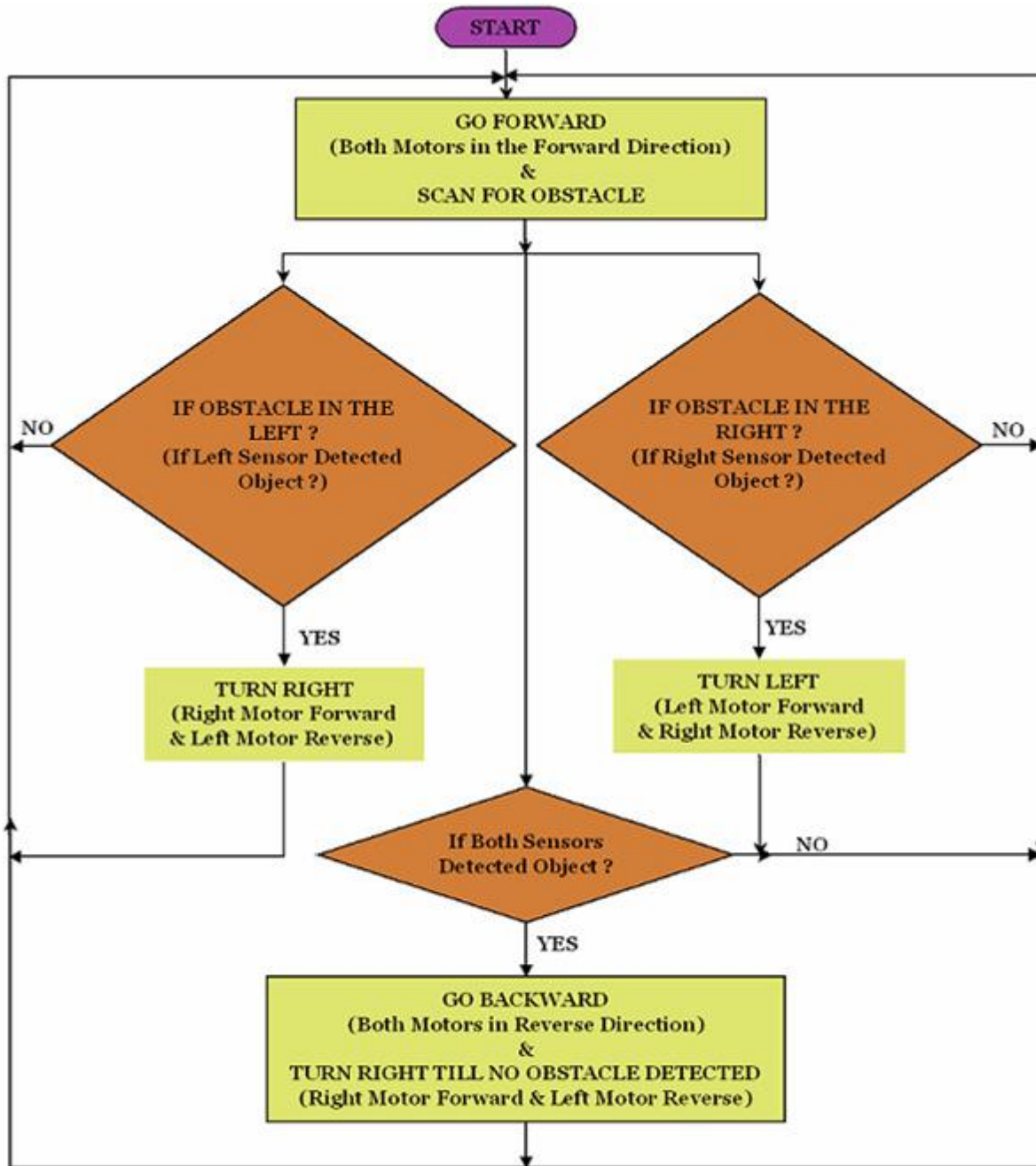
This platform which is developed using PIC 18 F and Arduino ATMEGA 16 microcontrollers be called as UNITY. Unity relies on two Infrared (IR) sensors, which are mounted on the bottom of the chassis to detect the position of the black line on the course. The width of the black track must be less than the distance between IR sensors, so that both the IR emitters will be facing towards the white surface. While the UNITY moves along the path, the IR Emitter emit light beam towards the surface of the course and the IR Detector will detect the reflected infrared light beam. Hence, both receiver outputs, which are connected to the input of PIC microcontroller, will be in logic zero and the controller drives both the motors in the forward direction. When any of the IR emitter comes above the black track, the black surface absorbs the IR rays and hence no light beams get reflected back to the receiver. In the absence of IR signal, the receiver output changes to logic high state and the microcontroller sends the control signals to the motors based on these signal and the UNITY aligns properly into the black track by turning accordingly. This scanning and aligning is done in a specific interval. Because of this, the UNITY keep on following the black track. When both sensors receive reflected IR signal then the UNITY moves forward. This is achieved by driving both the motors in the forward direction. When right sensor detects the absence of reflected IR beam, the UNITY takes a right turn by reducing the right motor speed and keeping the left motor speed constant. When left detector detects the absence of IR beam, the UNITY takes a left turn by reducing the left motor speed and keeping right motor speed constant.

## 5.2 ALGORITHMS

### LINE FOLLOWER ALGORITHM



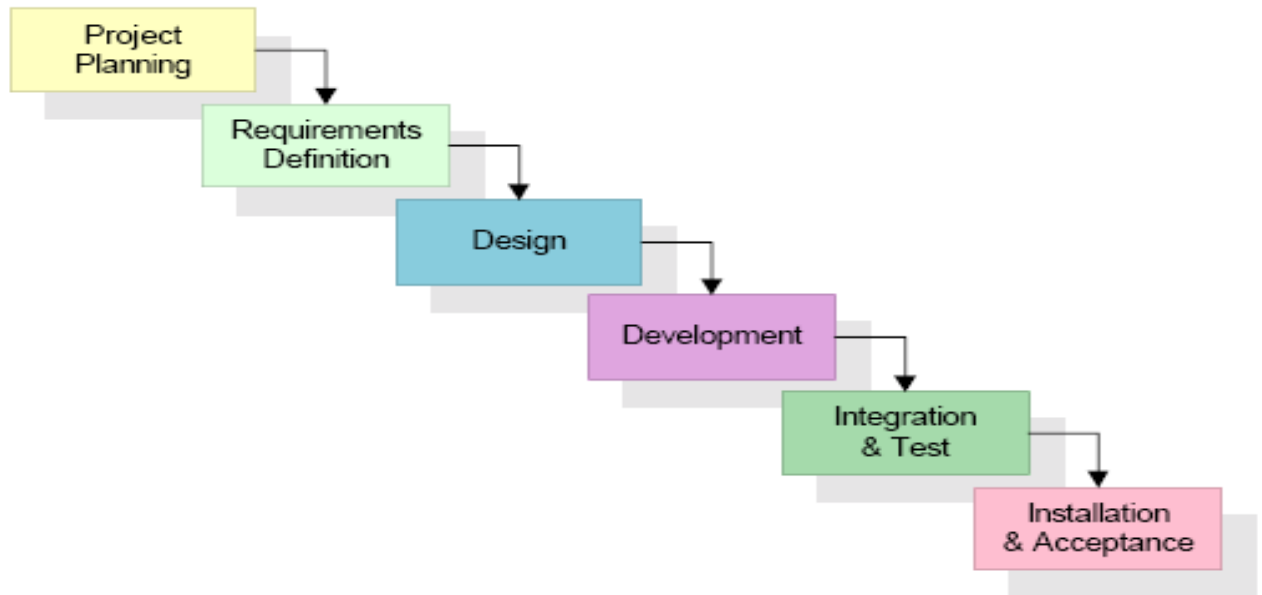
## OBSTACLE DETECTION AND AVOIDANCE ALGORITHM



Flow Chart for Obstacle Detection and Avoidance

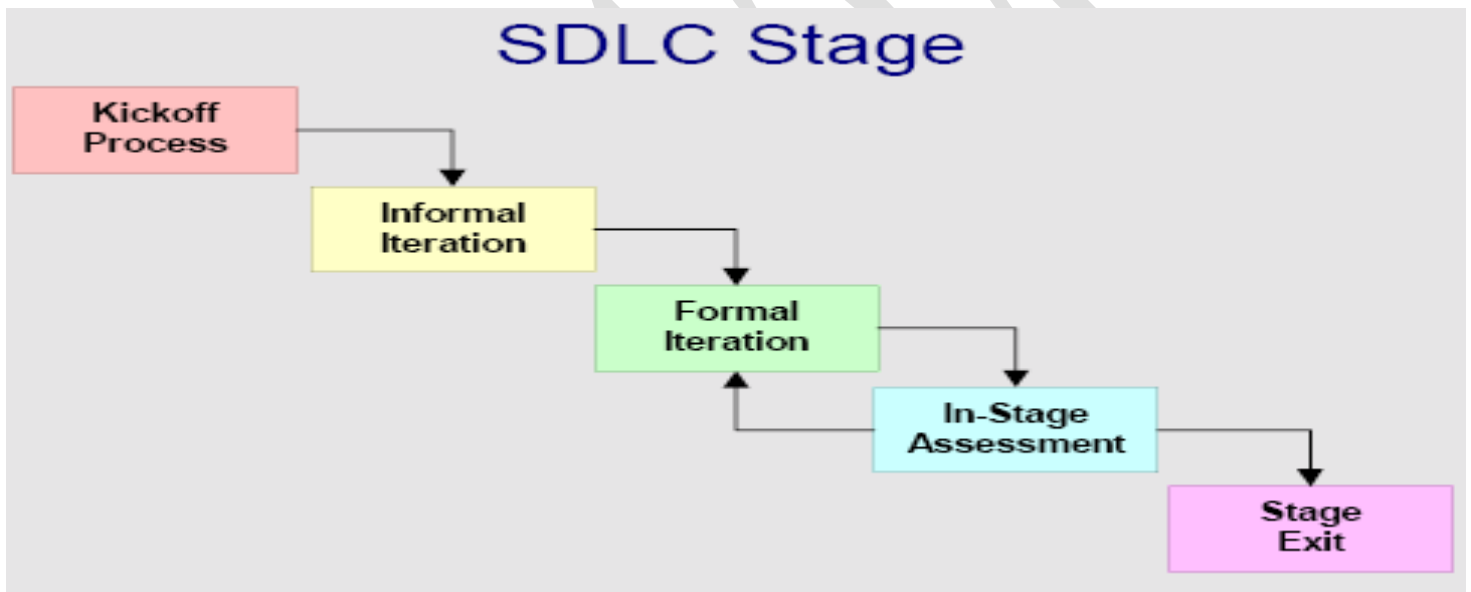
## 5.3 INPUT DESIGN

Input design is the process of converting environment variables and noise as inputs to a embedded platform designed with microcontroller. Input design is one of the most critical phase of the operation of microcontroller based system and is often the major problem of a system.



## 5.4 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; here it is nothing a line followed by a wheeled robot. Output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. In any system, the output design determines the input to be given to the application.





# SYSTEM TESTING

# SYSTEM TESTING AND MAINTENANCE

## 6.1 UNIT TESTING

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. As the main function of the device is to follow line it is tested for black and white color patches whether it stops when two of sensors are above black color. Similarly, it is tested against the environmental noise such as all the lights inside the room were kept on and motion of UNITY is tested whether it follows path or not. Then the environment variable testing is carried out. For example putting a human hand in front of the obstacle sensor.

In the line following unit each and every TSOP as well as IR sensors were tested against environmental noise so that accuracy can be increased. Finally the testing was carried based on algorithm used and time taken by moving robot was calculated and enhancements were made such that it will respond quickly.

## 6.2 INTEGRATION TESTING

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

When all the units are integrated then Arduno board was tested for LDR and same result was maintained for PIC 18F, since, the interfacing was carried out.

## 6.3 VALIDATION TESTING

The final step involves Validation testing, which determines whether the embedded platform function as the user expected. The end-user rather than the system developer conduct this test. User performs tests in its own environment and checks for performance. The compilation of the entire project is based on the full satisfaction of the end users. In the project, validation testing is made in various forms. In a path few obstacles were kept and were checked for whether it avoids them.



# SYSTEM IMPLEMENTATION



# IMPLEMENTATION

## 7.1 MAINTENANCE:

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or hardware platform. This is called the maintenance of the system. Nowadays there is the rapid change in the hardware world. Due to this rapid change, the system should be capable of adapting these changes. In our project the process can be added with no or few adjustments to entire assembly.

Maintenance plays a vital role. The system liable to accept any modification after its implementation. This system has been designed to favor all new changes. Doing this will not affect the system's performance or its accuracy.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, the application is implemented on PIC 18 F microcontroller based embedded platform as shown in figure 4. The power source is main concern and it should not be ON when the system is not in use. Usage of 1.5 V battery source makes it usable only for 1 hour without interruption. The power source should changed time to time basis.



# APPENDIX

# APPENDIX

## 8.1 SOURCE CODE

### 1. linefollowing.c

```
// Code for line following

//-----

// header file for Line Follower

#include<linefollow.h>

void main(void)

{

//setting PORTA as inputs except PA3

TRISA=0b11110111;

//setting PORTB as outputs

TRISB=0b00000000;

//setting PORTC as outputs

TRISC=0b00000000;

//setting PORTD as outputs

TRISD=0b00000000;
```

```
//setting PORTE as outputs
```

```
TRISE=0b11111111;
```

```
//Making the buzzer off
```

```
buzzer=0;
```

```
/*Initializing adc, pwm modules and setting PA0, PA1, PA2 AND PA3 pins as analog.*/
```

```
initialize();
```

```
// loop to perform line follower using 2 sensors
```

```
while(1)
```

```
{
```

```
// if line is between 2 sensors
```

```
if(rightlsensor==ws && leftlsensor==ws)
```

```
{
```

```
// then move forward
```

```
speedirr(512,cw);
```

```
speedirl(512,cw);
```

```
}
```

```
// if the bot has drifted to the left of the line
```

```
if(rightlsensor==bs && leftlsensor==ws)

{

    // Then move the bot to the right

    speedirr(512,aw);

    speedirl(512,cw);

}

// if the bot has drifted to the right of the line

if(rightlsensor==ws && leftlsensor==bs)

{

    // then move the bot to the left

    speedirr(512,cw);

    speedirl(512,aw);

}

/* if the bot has encountered a black line or surface orthogonal to its path */

if(rightlsensor==bs && leftlsensor==bs)

{

    speedirr(0,cw);

    // Then move stop
```

```
speedirl(0,cw);  
  
}  
  
}  
  
}  
  
//-----
```

## 2. linefollow.h

```
#ifndef __UNITY_H    //custom header file creation

#define __UNITY_H

//-----

#include <adc.h>      //header files required

#include <stdio.h>

#include <stdlib.h>

#include <p18f4550.h>

#include <delays.h>

#include <pwm.h>

#include <i2c.h>

#include <math.h>

#include <xlcd.h>

//-----

#pragma udata        //code required for bootloading

extern void _startup (void);

#pragma code _RESET_INTERRUPT_VECTOR = 0x000800

void _reset (void)
```

```
{

_asm goto _startup _endasm

}

#pragma code

#pragma code _HIGH_INTERRUPT_VECTOR = 0x000808

void _high_ISR (void)

{

;

}

#pragma code _LOW_INTERRUPT_VECTOR = 0x000818

void _low_ISR (void)

{

}

#pragma code

//-----

//constant defenitions
```



#define 10 PORTAbits.RA0

#define 11 PORTAbits.RA1

#define 12 PORTAbits.RA2

#define mode (PORTA&0b00000111)

#define ws 0

#define bs 1

#define ob 0

#define nob 1

#define cw 0

#define aw 1

#define pit 1

#define nopit 0

#define wall 0

#define nowall 1

#define light 1

#define nolight 0

#define rightlsensor PORTAbits.RA5

#define leftlsensor PORTEbits.RE0

```
#define righsensor  PORTEbits.RE1

#define leftsensor  PORTEbits.RE2

#define buzzer PORTAbits.RA3

#define bld        PORTBbits.RB4

#define motorra PORTCbits.RC1

#define motorrb PORTDbits.RD0

#define motorla PORTCbits.RC2

#define motorlb PORTCbits.RC0

#define motor_r_fwd PORTCbits.RC1=1;PORTDbits.RD0=0

#define motor_r_bwd PORTCbits.RC1=0;PORTDbits.RD0=1

#define motor_l_fwd PORTCbits.RC2=1;PORTCbits.RC0=0

#define motor_l_bwd PORTCbits.RC2=0;PORTCbits.RC0=1

#define motor_r_stp PORTCbits.RC1=0;PORTDbits.RD0=0

#define motor_l_stp PORTCbits.RC2=0;PORTCbits.RC0=0

#define allanalog OpenADC(ADC_FOSC_2 &
ADC_12_TAD, ADC_CH4 & ADC_REF_VDD_VSS
& ADC_INT_OFF, ADC_13ANA);

#define allipdigital OpenADC(ADC_FOSC_2 & ADC_12_TAD,
```

```
ADC_CH4 & ADC_REF_VDD_VSS & ADC_INT_OFF, ADC_0ANA);
```

```
int rightldr, leftldr, rightthreshold, leftthreshold;
```

```
void initialize( void)
```

```
{
```

```
T2CON=0b000000110;
```

```
PR2=0b11111111;
```

```
CCPR1L = 0b00110011 ;
```

```
CCP1CON = 0b00111100 ;
```

```
CCPR2L = 0b00110011 ;
```

```
CCP2CON = 0b00111100 ;
```

```
OpenADC(ADC_FOSC_2 & ADC_12_TAD, ADC_CH4 &
```

```
ADC_REF_VDD_VSS & ADC_INT_OFF, ADC_0ANA);
```

```
}
```

```
void speedirr(int r,int dir) //function for speed and direction control of right motor
```

```
{
```

```
if(dir==cw)
```

```
{
```

```
SetDCPWM1(r);
```

```
PORTCbits.RC0=0;
```

```
}
```

```
if(dir==aw)
```

```
{
```

```
SetDCPWM1(1023-r);
```

```
PORTCbits.RC0=1;
```

```
}
```

```
}
```

```
void speedirl(int l,int dir) //function for speed and direction control of left motor
```

```
{
```

```
if(dir==cw)
```

```
{
```

```
SetDCPWM2(l);
```

```
PORTDbits.RD0=0;
```

```
}
```

```
if(dir==aw)
```

```
{
```

```
SetDCPWM2(1023-l);
```

```
PORTDbits.RD0=1;
```

```
}
```

```
}
```

```
void acquire_ldr_digital_values(){
```

```
    allanalog;
```

```
    Delay10TCYx( 5 );           //to get right ldr vaue
```

```
    ConvertADC();
```

```
    while( BusyADC() );
```

```
    rightldr=ReadADC();
```

```
    SetChanADC(ADC_CH8);        //to change analog channel
```

```
    Delay10TCYx( 5 );           //to get left ldr value
```

```
    ConvertADC();
```

```
    while( BusyADC() );
```

```
    leftldr=ReadADC();
```

```
    SetChanADC(ADC_CH12);
```

```
    Delay10TCYx( 5 );
```

```
ConvertADC();

while( BusyADC() );

rightthreshold=ReadADC();

SetChanADC(ADC_CH9);

Delay10TCYx( 5 );

ConvertADC();

while( BusyADC() );

leftthreshold=ReadADC();

SetChanADC(ADC_CH10);

if(rightldr<rightthreshold)

rightldr=1;

else

rightldr=0;

if(leftldr<leftthreshold)

leftldr=1;

else

leftldr=0;

allipdigital;
```

```
}

int convert2digital(int l)

{

switch (l)

{

case 0:SetChanADC(ADC_CH0);

    break;

case 1:SetChanADC(ADC_CH1);

    break;

case 2:SetChanADC(ADC_CH2);

    break;

case 3:SetChanADC(ADC_CH3);

    break;

case 4:SetChanADC(ADC_CH4);

    break;

case 5:SetChanADC(ADC_CH5);

    break;

case 6:SetChanADC(ADC_CH6);
```

```
break;

case 7:SetChanADC(ADC_CH7);

break;

case 8:SetChanADC(ADC_CH8);

break;

case 9:SetChanADC(ADC_CH9);

break;

case 10:SetChanADC(ADC_CH10);

break;

case 11:SetChanADC(ADC_CH11);

break;

case 12:SetChanADC(ADC_CH12);

break;

}
```

```
//to set analog channel
```

```
Delay10TCYx( 5 );           //delay to charge ADC's internal capacitor
```

```
ConvertADC();               //conversion
```



```
while( BusyADC() );           //check if conversion is over

return(ReadADC());           //return the 10 bit digital value

}

#endif
```

### 3. Ardiuno-PIC Interfacing program :

```
// Arduino code

// This code will try to pass any received serial data to the serial logger built into the
// Arduino IDE. It out puts the binary for 'a' which we will try to match with real
// serial data from the PIC chip's output.

int ledPin = 13;           // built in led
int incomingByte = 'a';

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);    // turn on the LED so we know when the program starts
  Serial.print("Good to go!\n");  // Let the serial monitor know when the program is
                                  // running as well...
  Serial.print(incomingByte, BIN); // Print 'a' in binary format. Hopefully our PIC's
                                  // serial data will match this.
}

void loop()
{

  if (Serial.available() > 0) {
    while (Serial.available() > 0) {
      incomingByte = Serial.read();
      Serial.print(incomingByte, BIN);
```

```

    }
    Serial.println();
}
}

// PIC code
#include <16F690.h>
#fuses INTRC_IO, NOWDT, NOBROWNOUT, PUT, NOMCLR
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C1, rcv=PIN_C2, bits=8) // Set LED 2 (and the
corresponding output) to the transmit pin

void main(void)
{
    while(1){
        output_c(0b00000001); // turn on LED 0 on the board
        delay_ms(1000);        // wait 1 second
        output_c(0b00000000); // turn the LED off
        delay_ms(1000);        // wait 1 second
        putc('a'); // Send the character 'a' to the serial out.
        // This should be '01100001' or '1100001'
        // when it gets to the serial monitor
    }
}

```

## 4. LIGHT DEPENDENT RESISTOR FUNCTIONING CODE :

//Light Dependent Resistir Code

int r,g,b;

void setup()

{

Serial.begin(9600);

}

void loop()

{

delayMicroseconds(100); //give 0 to flush all the outputs

int ldr=analogRead(3);

int perldr;

perldr=ldr\*100/1023;

int i;

i=perldr\*255/100;

analogWrite(3,i);

analogWrite(5,i);

analogWrite(6,i);

```
delay(10); //give 0 to flush all the outputs
```

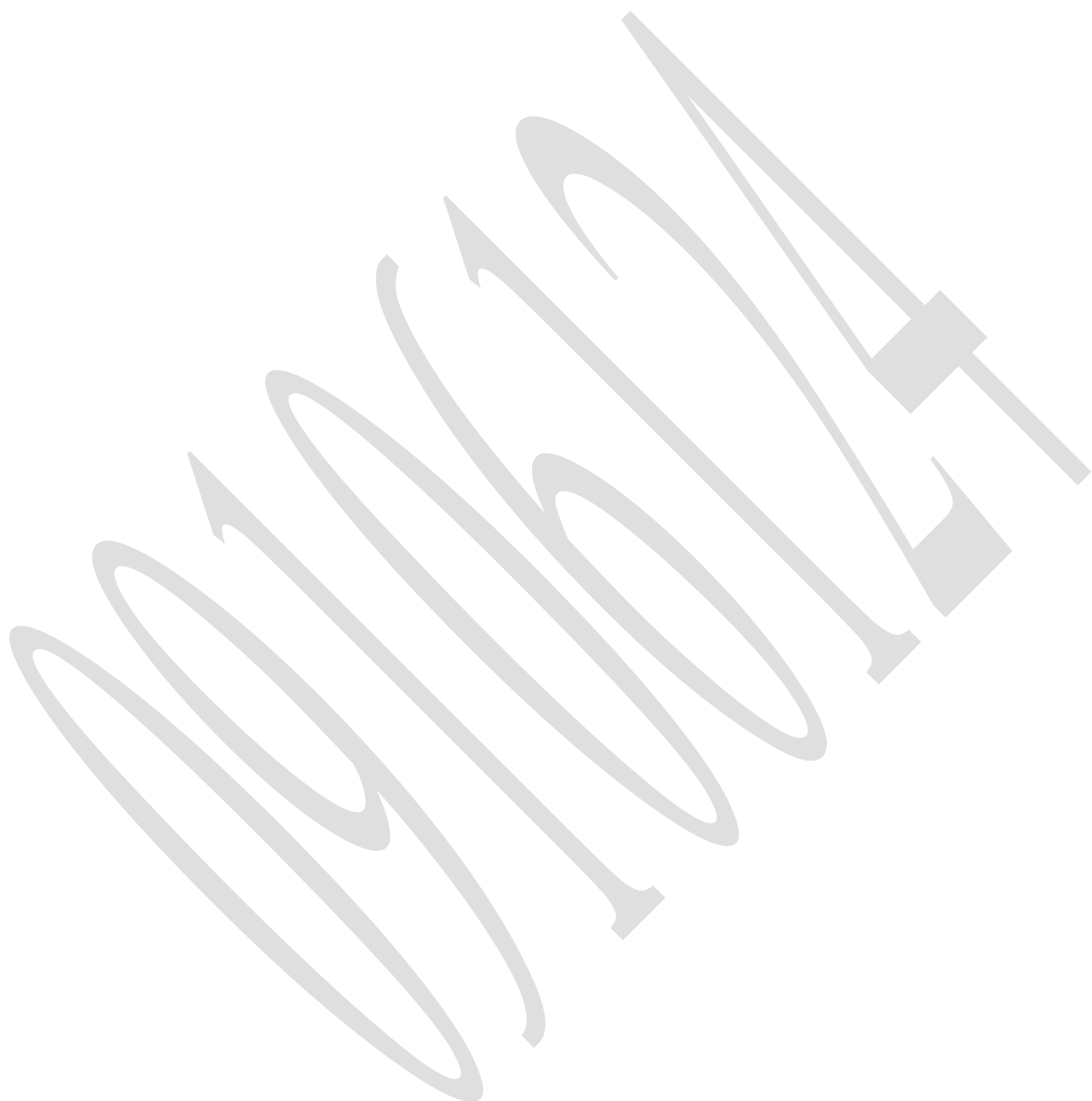
```
}
```



## 5. CODE FOR OBSTACLE AVOIDANCE :

```
// Code for obstacle avoidance
//-----
#include<revobot.h>
// Header file for Revoboard
void main(void)
{
// Setting PORTA as inputs except PA3
TRISA=0b11110111;
// Setting PORTB as outputs
TRISB=0b00000000;
// Setting PORTC as outputs
TRISC=0b00000000;
// setting PORTD as outputs
TRISD=0b00000000;
// setting PORTE as outputs
TRISE=0b11111111;
// making the buzzer off
buzzer =0;
// initializing adc, pwm modules and setting PA0,PA1,PA2 AND
PA3 pins as analog
initialize();
// loop to perform obstacle avoidance
while(1)
{
// if the bot has encountered an obstacle normally
if(rightsensor == ob && leftsensor == ob)
{
// first move the bot backwards
speedirr(1000,aw);
speedirl(1000,aw);
// for a small amount of time
```

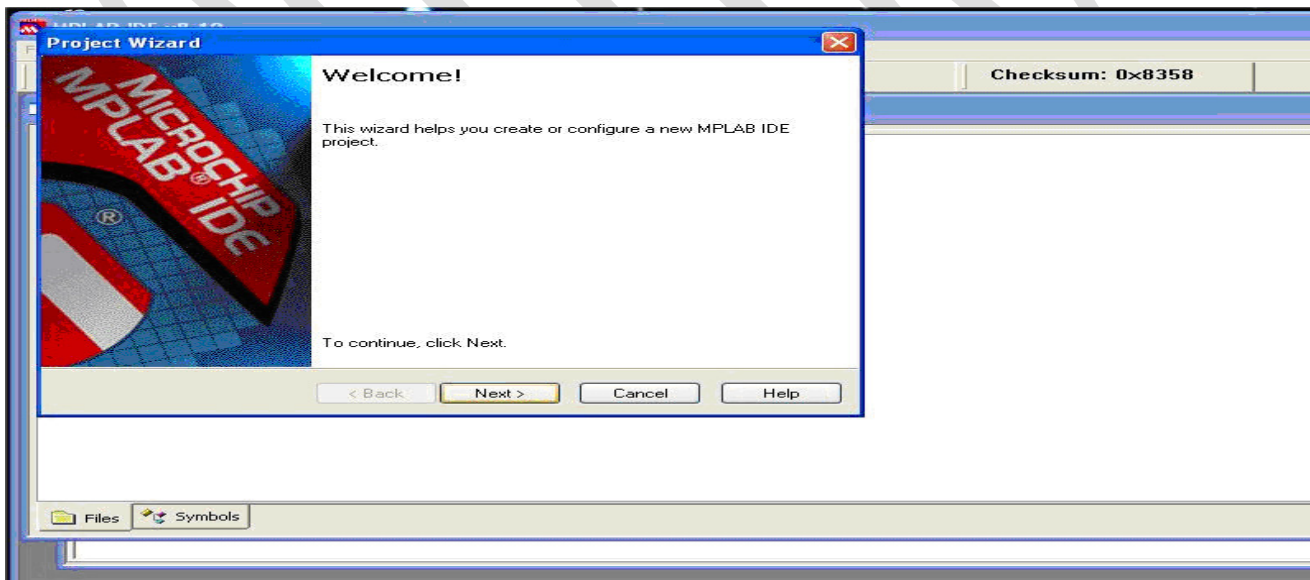
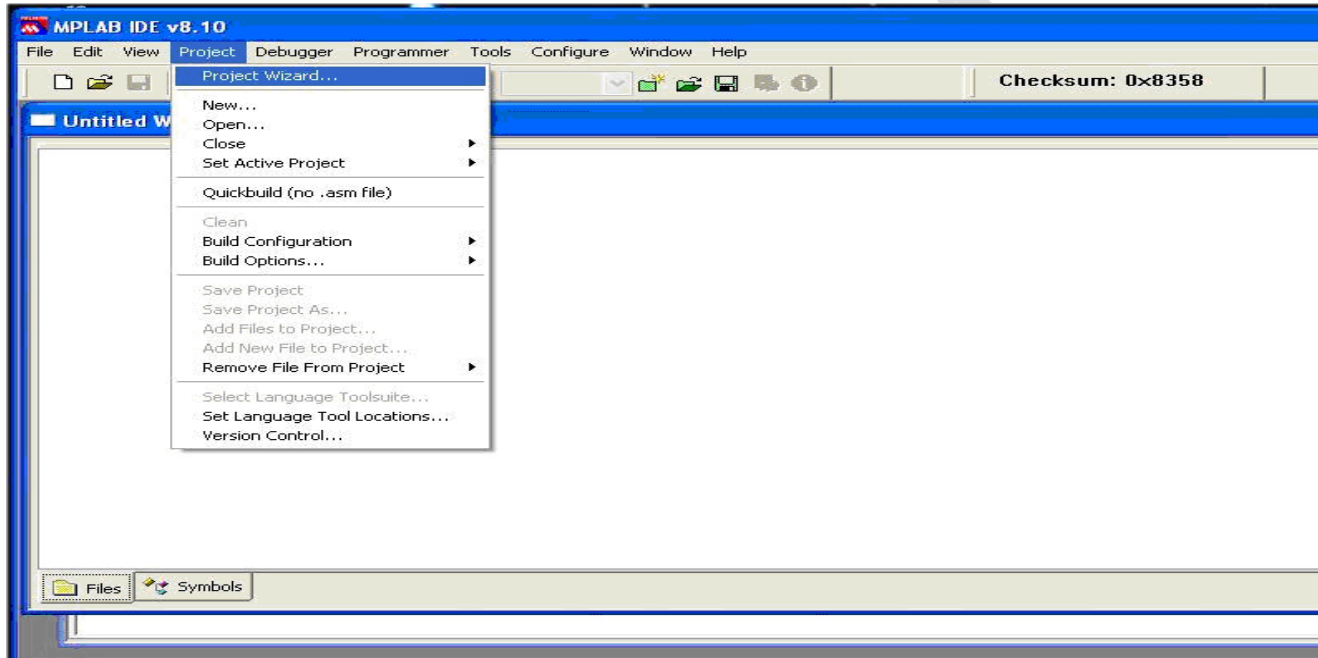
```
Delay10KTCYx(2);  
// Until the left sensor avoids the obstacle  
while(leftsensor==0)  
{  
    // turn to the the right to check for a path  
    speedirr(1000,aw);  
    speedirl(1000,cw);  
}  
// if the left sensor detects an obstacle  
if(rightsensor==nob && leftsensor==ob)  
{  
    // then turn to the right to avoid it  
    speedirr(1000,aw);  
    speedirl(1000,cw);  
}  
// if the right sensor detects an obstacle  
if(rightsensor==ob && leftsensor==nob)  
{  
    // then turn to the left to avoid it  
    speedirr(1000,cw);  
    speedirl(1000,aw);  
}  
// if both sensors dont detect any obstacle  
if(rightsensor==nob && leftsensor==nob)  
{  
    // then move forward  
    speedirr(1000,cw);  
    speedirl(1000,cw);  
}  
}  
}
```



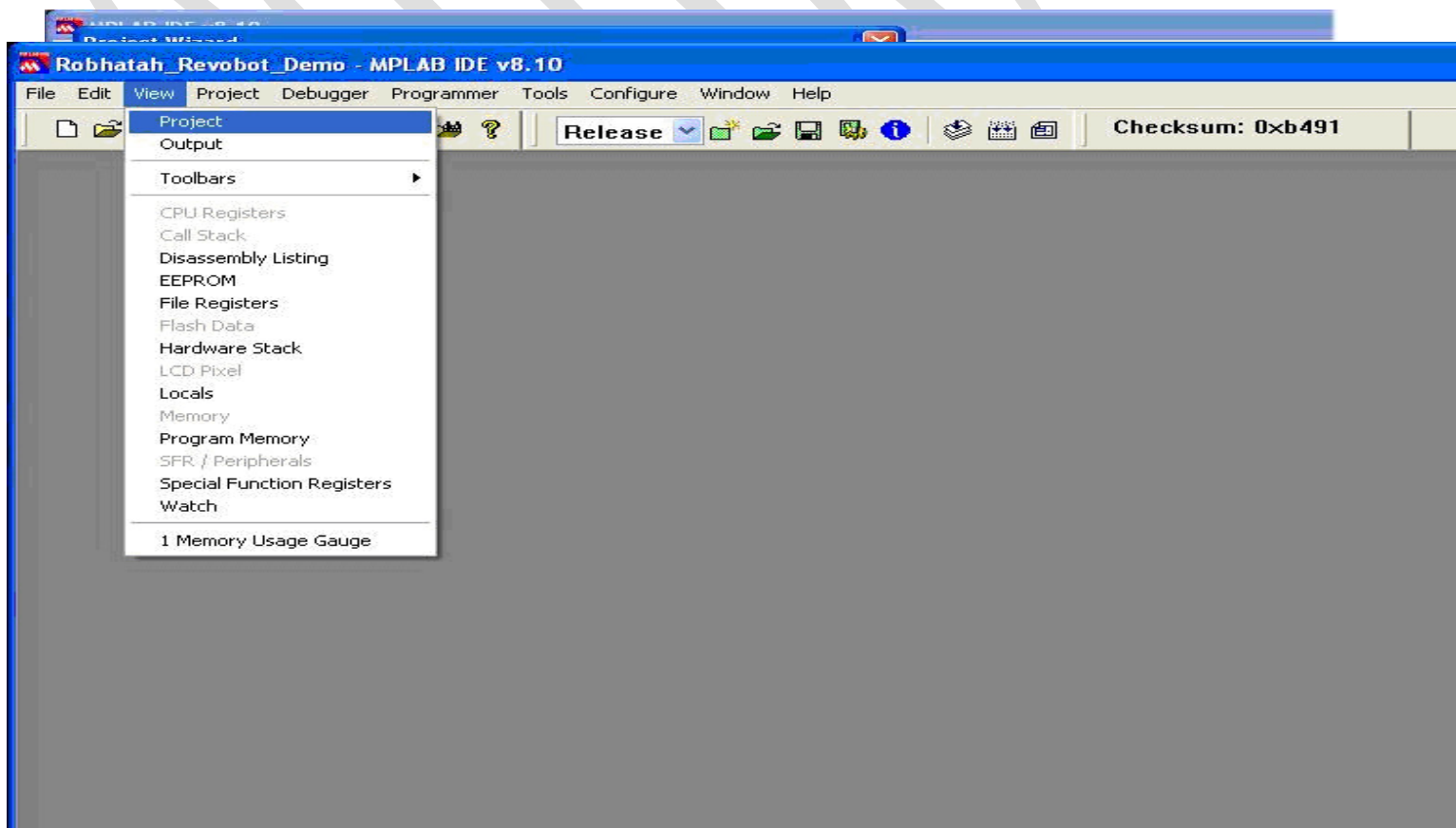
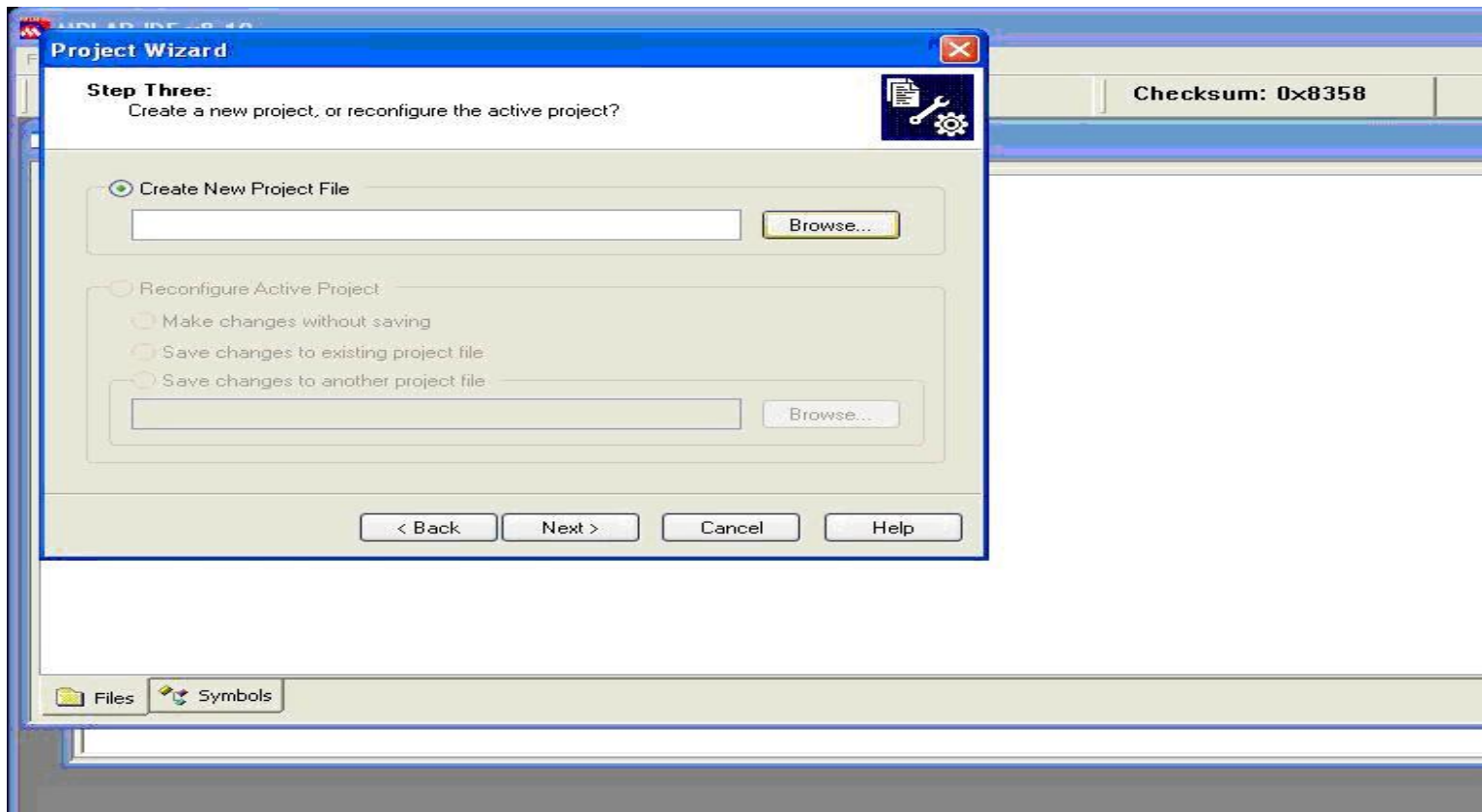


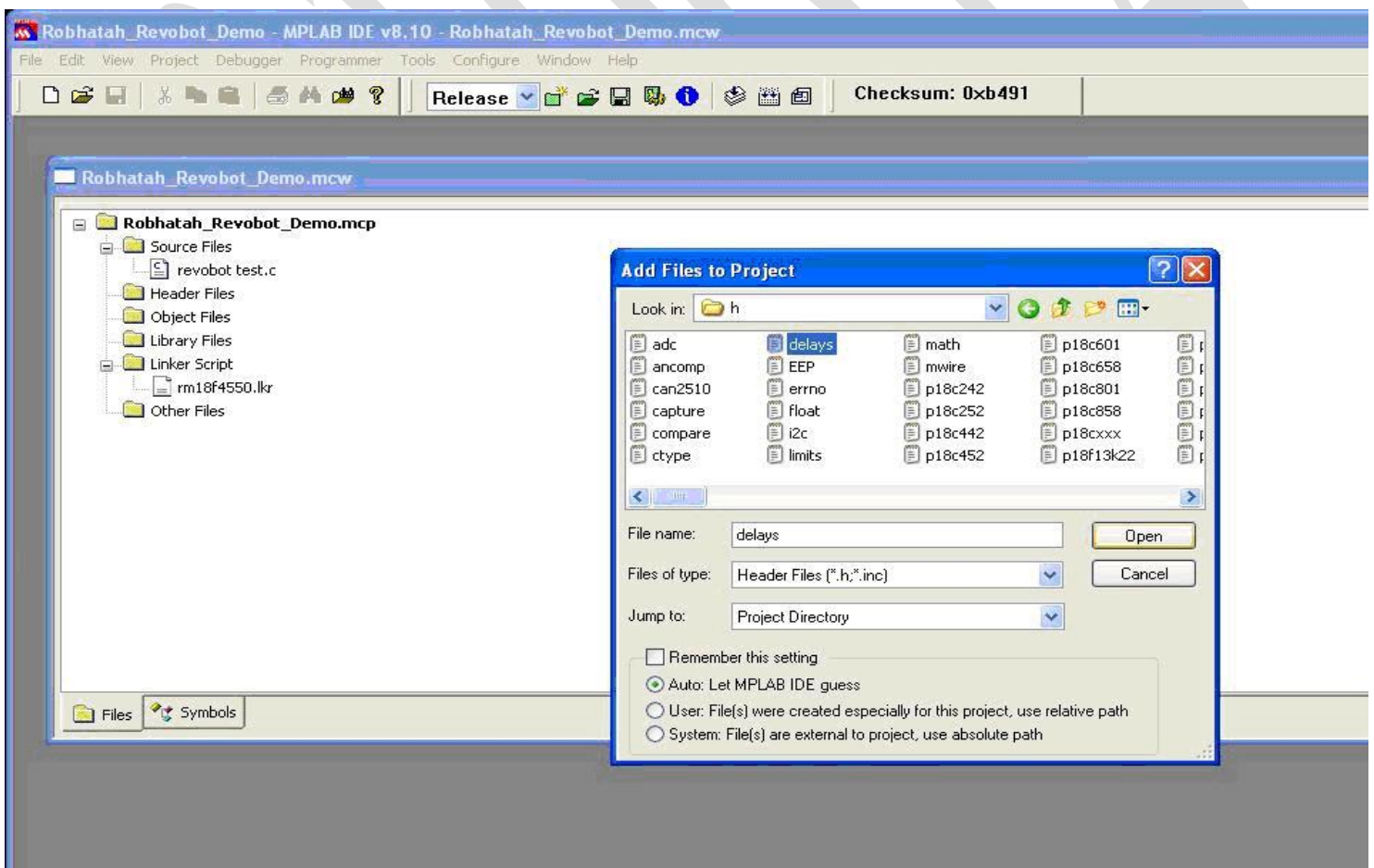
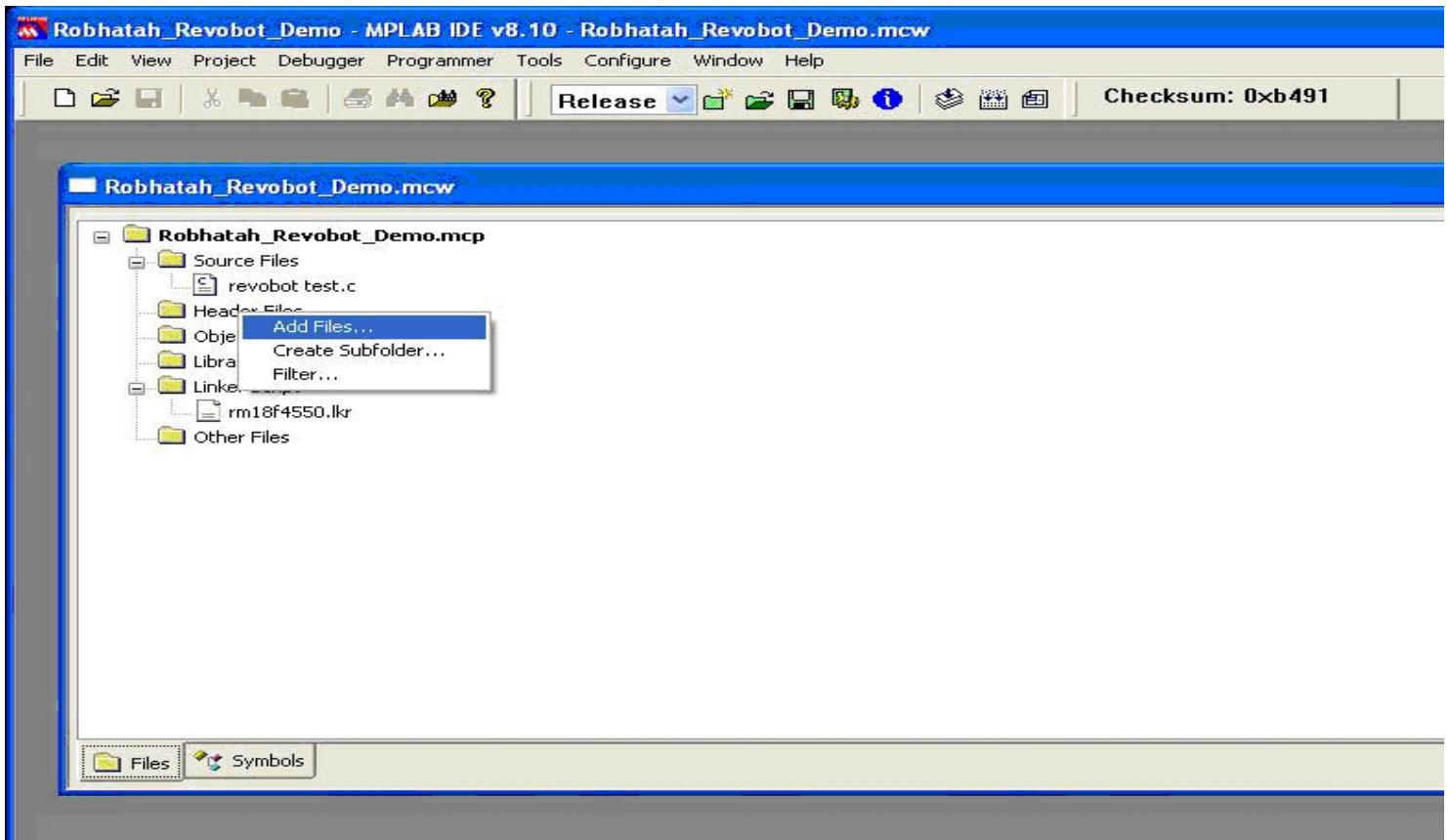
## 8.2 SCREENSHOTS

### 1. Create a New Project




## 2. Select Microcontroller Family





### 3. Building a Project and corresponding other files :



The screenshot shows the MPLAB IDE v8.00 interface. The title bar reads "demo071108 - MPLAB IDE v8.00 - [Output]". The menu bar includes File, Edit, View, Project, Debugger, Programmer, Tools, Configure, Window, and Help. The toolbar contains various icons for file operations and development. A red box highlights the "Build All" button, which is represented by a stack of documents with a downward arrow. Below the toolbar, the "Checksum: 0x4315" is displayed. The "Build" tab is active, showing the output window with the following text:

```
Debug build of project 'C:\revobot\revobot.mcp' started.  
Preprocessor symbol '__DEBUG' is defined.  
Wed Nov 19 03:17:56 2008
```

---

```
Clean: Deleting intermediary and output files.  
Clean: Deleted file "C:\revobot\revobot.mcs".  
Clean: Done.  
Executing: "C:\MCC18\bin\mcc18.exe" -p=18F4550 "revobot.c" -fo="revobot.o" -D__DE  
MPLAB C18 v3.21 (demo)  
Copyright 2000-2008 Microchip Technology Inc.  
Days remaining until demo becomes feature limited: 60  
Executing: "C:\MCC18\bin\mplink.exe" /I"C:\MCC18\lib" "..\MCC18\lkr\rm18f4550.lkr" "re  
MPLINK 4.21, Linker  
Copyright (c) 2008 Microchip Technology Inc.  
Errors : 0
```

---

```
MP2HEX 4.21, COFF to HEX File Converter  
Copyright (c) 2008 Microchip Technology Inc.  
Errors : 0
```

---

```
Loaded C:\revobot\revobot.cof.
```

---

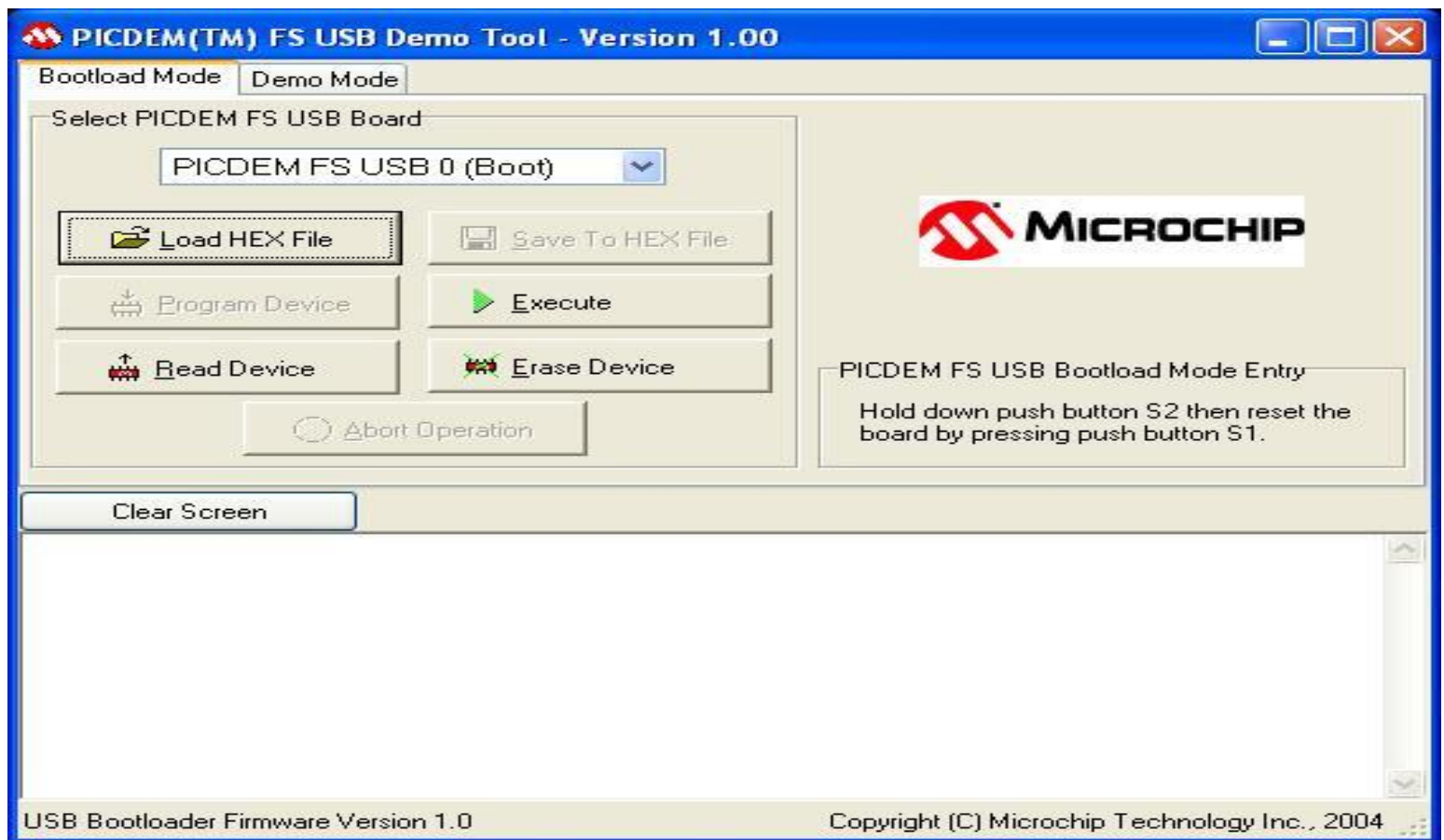
```
Debug build of project 'C:\revobot\revobot.mcp' succeeded.  
Preprocessor symbol '__DEBUG' is defined.  
Wed Nov 19 03:17:57 2008
```

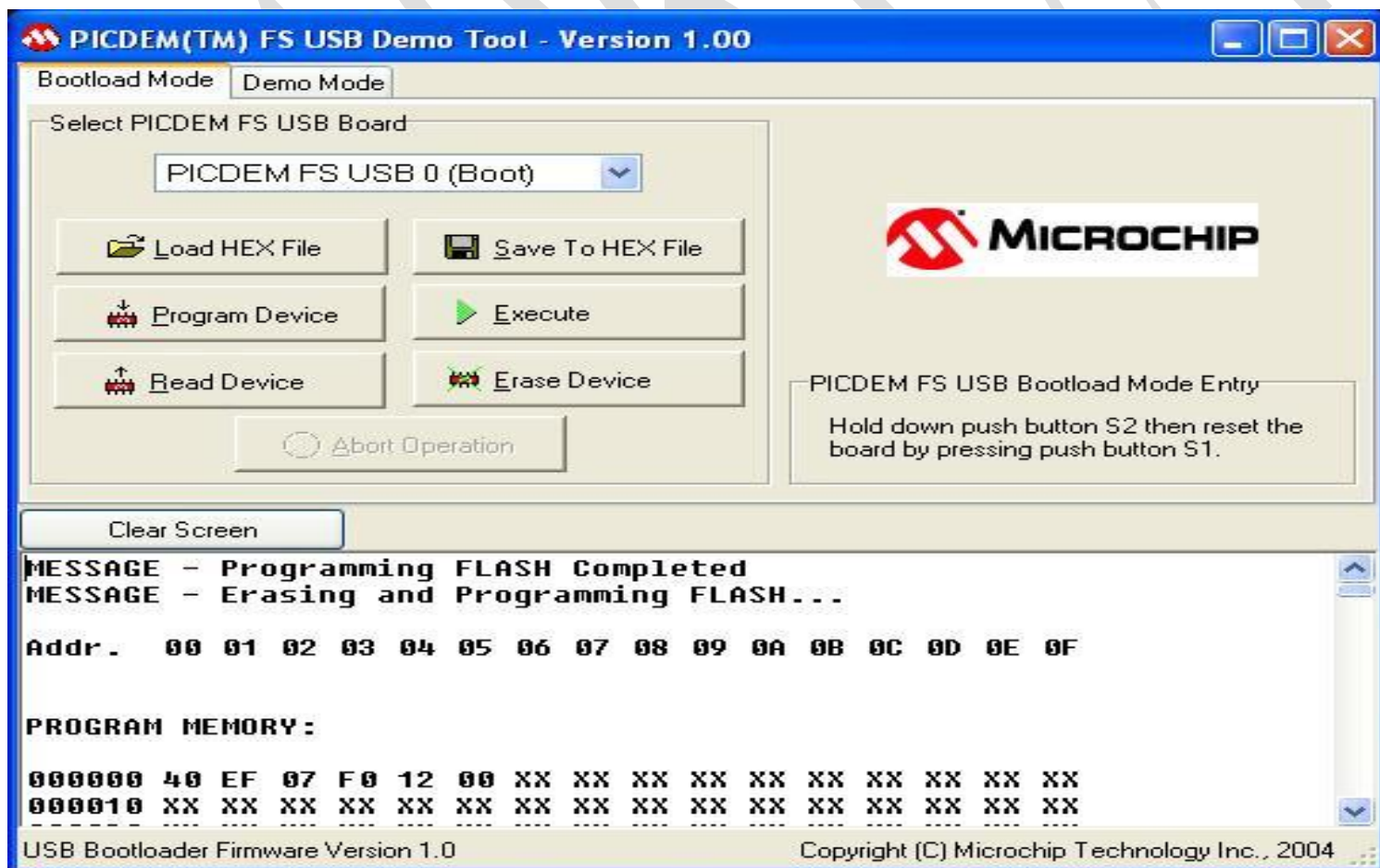
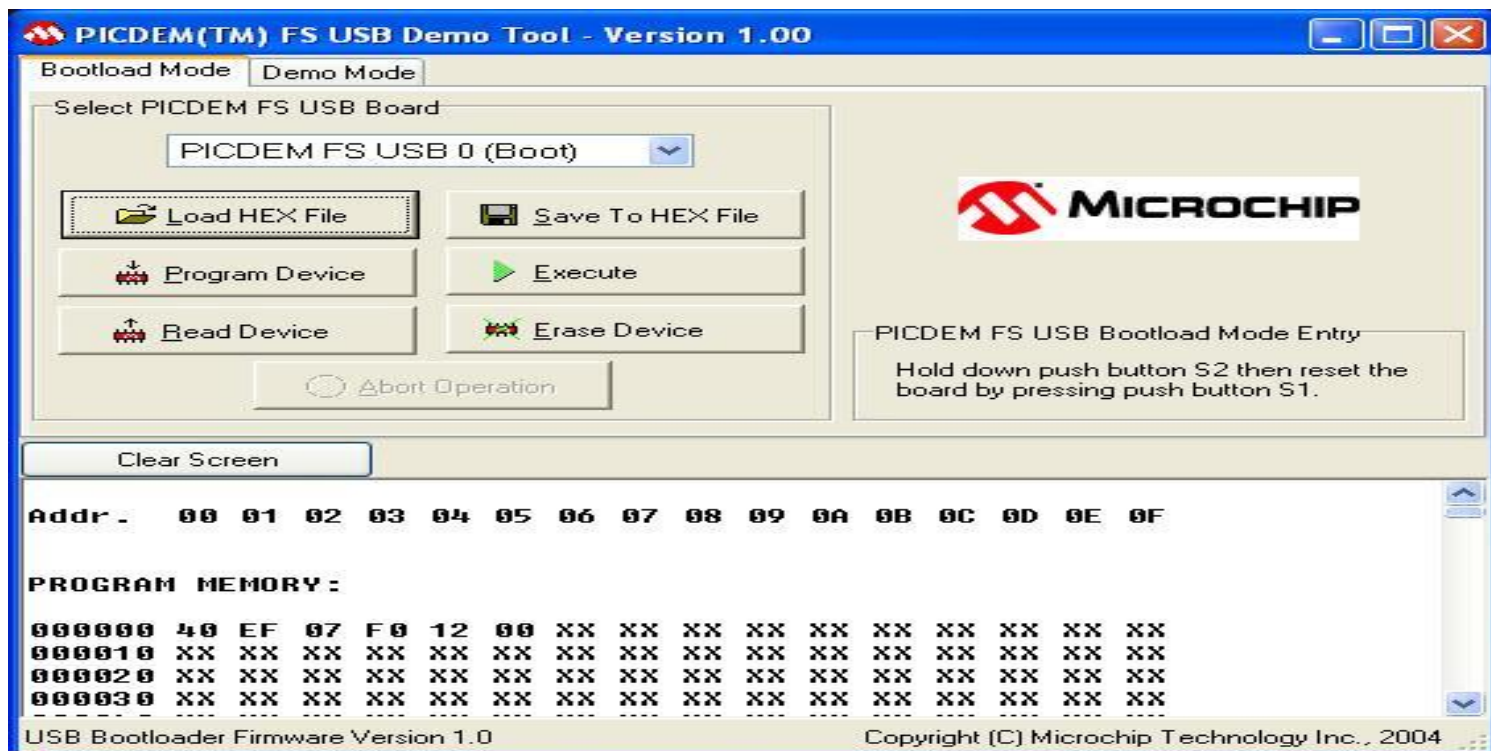
---

**BUILD SUCCEEDED**



#### 4 Burning HEX program into the device:







# CONCLUSION AND FUTURE ENHANCEMENT

## 9.1 CONCLUSION :

A Line Follower Robot which avoids obstacle along its path is a nice work to be carried out as a beginner. While working on it a pure electronics approach along with system level programming using C Language is key learning curve. Using this project we can easily elaborate the working of sensors, motor drivers. Also, the interfacing of Arduino and PIC is possible and for which a separate driver program has to be written. Light Intensity detection algorithm was a key in the project. An Autonomous Line Follower which follows line and avoids obstacles without no or very less human intervention is successful one.

## 9.2 FUTURE ENHANCEMENTS :

Using this project we can even develop a small toy which resembles to a car. In this project Sun Light Tracking and Charging module can be added with a financial support from an organization and even helping agent for household application can be developed as a future enhancement. In future Li-Po batteries also can be added to the device so that it can be operated for longer time without intervention. A grid of such Li-Po battery can be useful. Even already added module LDR circuit can be also enhanced to high level with the addition of Solar-Plates available in the market which are not added to this project due to cost factor. Use of higher order microcontrollers in the same family such as PIC 30 or 64-bit PIC microcontroller will make this device to work faster and will render highest of the efficiency. All this changes can be done with or without making changes in current circuitry or an external bread board can also be added to it.





# REFERENCES

# REFERENCES

## Books :

1. PIC 18F Datasheet.
2. UNITY Student Guide For Specification purpose.
3. *Embedded Robotics : Mobile Robot design* by Thomas Branul.
4. *Mobile Robotics : A Practical Implementation* by Ulrich

## Sites Referred :

The sites referred here are written in their standard web address because number of references taken from these sites are large in numbers. Also, these are open source sites so they work in forum culture so giving exact address is difficult task.

1. [www.arduino.cc](http://www.arduino.cc)
2. [www.sourceforge.net](http://www.sourceforge.net)
3. [www.microchip.com](http://www.microchip.com)
4. [www.avrfreaks.net](http://www.avrfreaks.net)
5. [www.robhatah.com](http://www.robhatah.com)