March 13, 2014

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

**Re: ENSC 305/440 Design Specification for Auto Shopping Guide**

Dear Dr. Rawicz:

Attached is a document from Easy Way Inc. describing the design specification of Easy Way Auto Shopping Guide (ASG) System. We are designing and implementing an auto shopping guide that can automatically guide customers to the products that they want. The ASG will find the shortest path to the desired product and provide detailed price information to customers.

This design specification provides an overview of the system and detailed information about both hardware and software design. Future improvements are discussed but will not be contained at this stage of development.

Easy Way Inc. is founded by 5 creative and skillful senior engineering students: Joseph Lu, James Lin, Jacqueline Li, Tao Xiong and Enzo Guo. If you have any question about this proposal, please feel free to contract me by phone at (778) 385 – 0321or by email at zla18@sfu.ca.

Sincerely,

Joseph Lu

Joseph Lu
Chief Executive Officer
Smart Way Inc.

Enclosure: Design Specification for Auto Shopping Guide

EASY WAY

# DESIGN SPECIFICATION FOR AUTO SHOPPING GUIDE

**Project Team:**
Joseph Lu (CEO)
James Lin (COO)
Enzo Guo (CTO)
Tao Xiong (CFO)
Jacqueline Li (VP)

**Submitted to:**
Dr. Andrew Rawicz – ENSC 440
Mr. Steve Whitmore – ENSC 305
School of Engineering Science
Simon Fraser University

March 13, 2014

## Abstract

The document details the design specifications of the prototype of Easy Way's Auto Shopping Guide System. It provides a set of comprehensive descriptions for the design and development of the proof—of—concept prototype. The document will only discuss the design considerations pertaining to the functional requirements categorized as Rn—A and Rn—B that are described in the previous functional specifications including hardware and software requirements, as well as safety and sustainability.

The safety and sustainability of the system will also be included in this document. The section will be analyzed based on the "cradle—to—cradle" cycle for the proof—of—concept prototype. Major considerations for final production device will also be covered in this section to ensure a high quality marketable production.

In this document, fundamental test plan of the proof—of—concept prototype will be outlined in this document in order to ensure the prototype meet required functional specifications. Test procedures and possible outcomes will be demonstrated in this section.

## Contents

Abstract ........ ii
List of Figures ........ iii
List of Tables ........ iv
GLOSSARY ........ iv
1. Introduction ........ 1
1.1 Scope ........ 1
1.2 Intended Audience ........ 1
1.3 Background ........ 1
2. System Overview ........ 3
2.1 Hardware ........ 4
2.2 Software System ........ 5
3. ASG Hardware ........ 6
3.1 Robotic Car ........ 6
3.1.1 Micro-controller ........ 8
3.1.2 Infrared Receiver ........ 9
3.1.3 Motor Bridge. ........ 11
3.1.4 Collision Sensors ........ 11
3.1.5 LEDs ........ 12
3.2. Ultrasonic Ranging Sensor ........ 13
3.2.1 Input Voltage Regulation ........ 13
3.2.2 Ultrasonic Signal Transmitter/Receiver ........ 14
4. ASG Software ........ 15
4.1 User Interface and Database ........ 15
4.2 Compiler and flash tool ........ 20
5. USTAINABILITY AND SAFETY ........ 21
5.1 Safety and Reliability ........ 21
5.2 Sustainability ........ 22
6. System Test Plan ........ 23
6.1 Individual Module Testing ........ 23
6.1.1 Mobile System Testing ........ 23
6.1.2 Data Transmit Testing ........ 24

6.1.3 Robotic Car Testing ........ 25

6.2 Combined Modules Testing ........ 26

7. Conclusion ........ 27

8. Reference ........ 28

APPENDIX A – ROBOTIC CAR SCHEMATIC DIAGRAM ........ 30

APPENDIX B – ULTROSONIC SENSOR SCHEMATIC DIAGRAM ........ 31

APPENDIX C – TEST CODE FOR ASG ........ 32

## List of Figures

Figure 1 Shopping with ASG System ........ 2
Figure 2 System Overview: System is separated to be 2 major parts: mobile software system and Robotic hardware system ........ 3
Figure 3 ASG's Hardware Top-level Block Diagram ........ 4
Figure 4 ASG's Software Top-level Flow Diagram ........ 5
Figure 5 Top view of Robotic Car [2] ........ 7
Figure 6 Block Diagram of ATmega8 Micro-Controller [3] ........ 8
Figure 7 ATmega8 Circuit ........ 9
Figure 8 Block Diagram of SFH5110 [4] ........ 10
Figure 9 SFH5110 IR-receiver Circuit ........ 10
Figure 10 Motor Bridge Circuit ........ 11
Figure 11 Collision Sensors Circuit ........ 12
Figure 12 Motor LEDs Circuit ........ 12
Figure 13 Input Voltage Regulation Circuit [5] ........ 13
Figure 14 Ultrasonic Transmitter/Receiver Circuit [5] ........ 14
Figure 15 General Functionality Flowchart ........ 15
Figure 16 User Interface Design ........ 15
Figure 17 "Start Guiding" Prompt ........ 16
Figure 18 User Interface I/O Processing Flowchart ........ 16
Figure 19 Invalid Input Prompt ........ 17
Figure 20 Destination Coordinates in Plane Coordinate System ........ 19
Figure 21 Compiler Interface ........ 20

## List of Tables

Table 1 Information Stored in Database ........ 18
Table 2 Required Standards ........ 21
Table 3 Data Transmit Testing ........ 25

## GLOSSARY

| | |
|---|---|
| **ADC** | Analog-to-digital converter |
| **ASG** | Auto Shopping Guide |
| **AWI** | Approved new Work Item |
| **EEPROM** | Electrically erasable programmable read-only memory |
| **GUI** | Graphical user interface |
| **IC** | Integrated circuit |
| **IR-Transceiver** | Infrared transceiver |
| **ISO** | International Organization for Standardization |
| **IEC** | International Electro-technical Commission |
| **LED** | Light-emitting diode |
| **PC** | Personal computer |
| **PCB** | Printed circuit board |
| **SRAM** | Static random-access memory |
| **UI** | User interface |
| **USB** | Universal Serial Bus |

## 1. Introduction

The ASG, Auto Shopping Guide, is a system that allows users to be able to locate the items on their shopping lists. The system consists of both hardware and software. The hardware part that features a robotic car and several ultra—sonic sensors for range detection is mainly in charge of guiding users to their targeted location. The software part that is comprised of a user interface, database and robotic programming is primarily taking care of data manipulating, signal processing and route calculating. By inputting item names into the user interface, users are able to acquire the information of the items such as prices and locations in the form of coordinates. The fundamental functionality of the system is to help consumers, who are not familiar with layouts of local supermarkets, easily find what they desire to purchase. As a user friendly development team, we aim the design purpose of the system at people at all ages without any technical training.

### 1.1 Scope

This document presents the design requirements of the Auto Shopping Guide System. The set of requirements presented will fully describe the design specifications for the proof—of—concept system and partially describe the specifications for a production device. As this document is focusing on the proof—of—concept system, only requirements prioritized as Rn—A or Rn—B in the previous functional specification will be explicitly explained. Requirements categorized as Rn—C which is future development work will partially be discussed as well; nevertheless, they will not be physically implemented.

### 1.2 Intended Audience

The intended users of this document include all the members of Easy Way Inc. Developers shall refer to the specifications as over design guidelines to ensure all requirements and objectives are achieved. Testers shall refer to the specifications for implementing the test plan and confirming the correct and expected functional behavior of the Auto Shopping Guide System.

### 1.3 Background

The visual procedure of how ASG system helps users with their shopping experience is illustrated in the following figure.

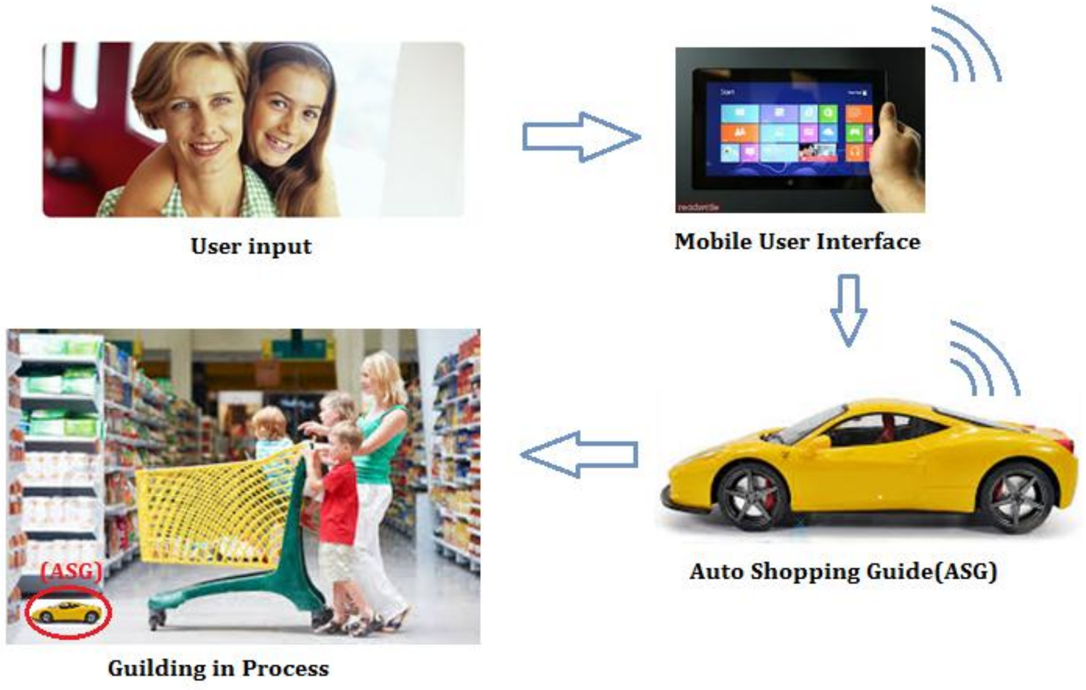


Figure 1 Shopping with ASG System

Original inputs, which are item names, are inserted in the user interface by users and information is loaded from the database. The outputs, which are coordinates, will then be implemented into robotic programming which is written in C language. After the implementation, the program will calculate the primary routes for the robotic car and convert the C file that contains the routes to a Hex file. These processes will be done via mobile devices. The final output Hex file will be transmitted through a USB IR—transceiver to command the hardware part of the system. The hardware part consists of a robotic car, which is assembled by soldering electronic components on the pre—designed PCB and several ultra—sonic ranging sensors. The output Hex file that is transmitted will be picked up by the IR—receiver and loaded into the Atmel AVR 8—bit microcontroller on the robotic car. The ultra—sonic sensors that are mounted on the robotic car are in charge of range detection. They are used to prevent the robotic car from possible collisions during moving and maintain a reasonable distance between the robotic car and the user. The entire operation of the system is optimized to serve users without any technical training.

## 2. System Overview

The Auto shopping guide is a revolutionary design of the current shopping method. It will help customers find what they need quickly. The top level system can be modeled as shown in the Figure 2.

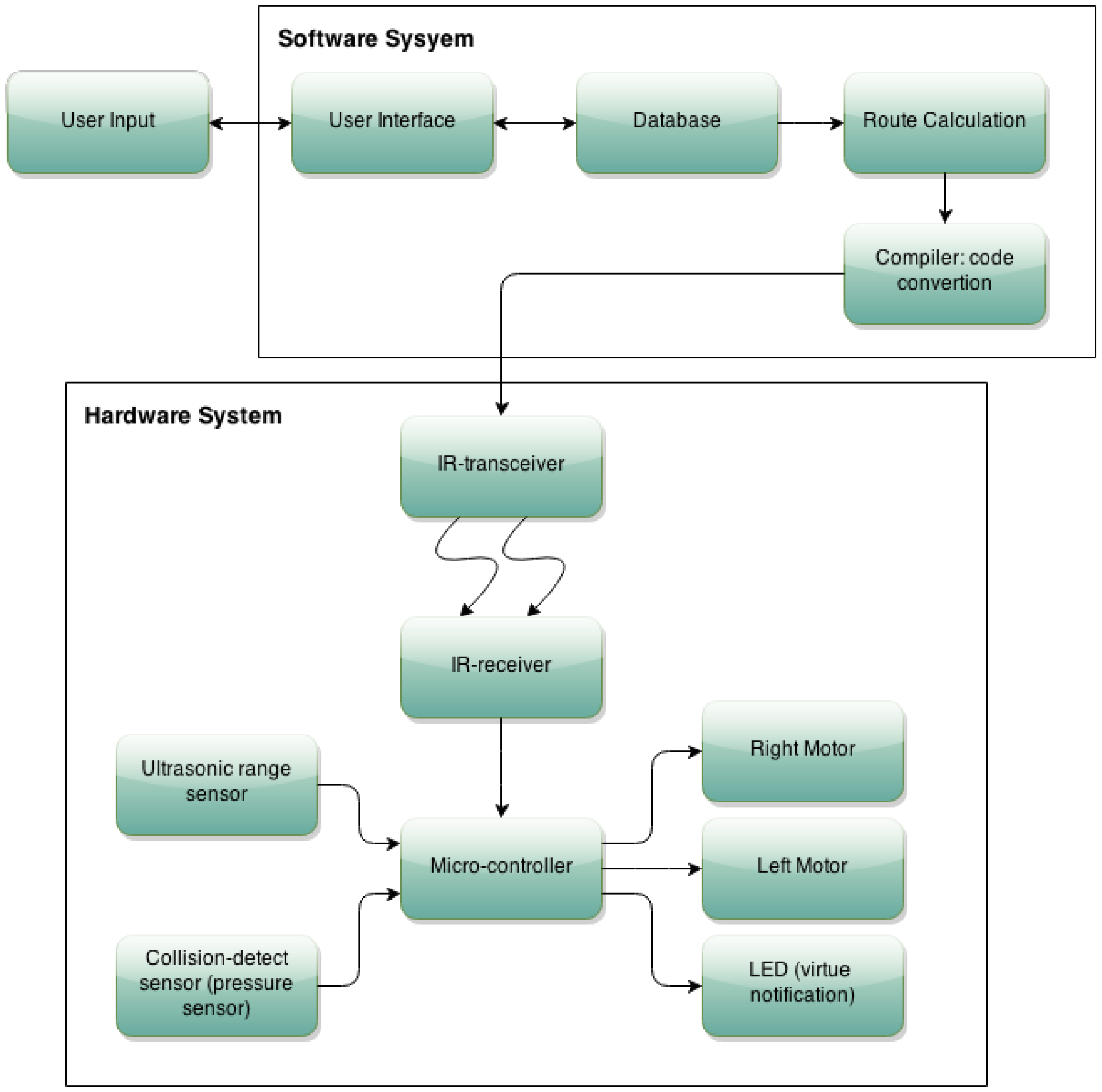


Figure 2 System Overview: System is separated to be 2 major parts: mobile software system and Robotic hardware system

The design consists of two major parts: mobile application (software system) and Robotic car (hardware system), and each part contain several individual modules. Software system contains user interface (UI), database server, route calculation and robotic programming compiler. Hardware system contains IR-transceiver/receiver, micro-controller, ultra—sonic ranging sensor, two independent motors and LEDs.

## 2.1 Hardware

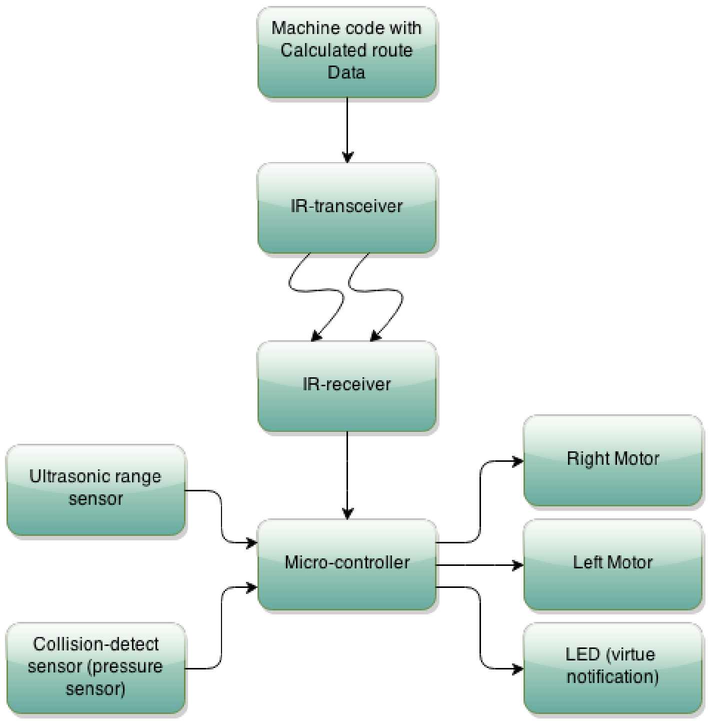

Machine code with Calculated route Data
IR-transceiver
IR-receiver
Ultrasonic range sensor
Micro-controller
Collision-detect sensor (pressure sensor)
Right Motor
Left Motor
LED (virtue notification)

Figure 3 ASG's Hardware Top-level Block Diagram

As the Figure 3 shows, the ASG's hardware system contains Micro-controller, USB-transceiver, IR-transceiver, motors (engine), ultra—sonic ranging sensor, status LED and 6 collisions-detect sensors. Once all the modules are powered, the system should automatically configure them to perform all the functionalities. When the IR-receiver receives the route data from the IR-transceiver (software system), the micro-controller will store the received route data in the micro-controller memory. Micro-controller will control the two independent motors. By changing the rotation speed of each motor, the robotic car will be able to move forward or backward, and make a turn. The micro-controller will also read the signal from each sensor. The front collision-detect sensor will prevent the car from hitting the front obstacle. The back ultrasonic sensor will detect the distance between the car and the customer behind to prevent the customer getting too far away from the ASG.  Status LEDs will be the visual alarm to users if the destination is reached, and it will also monitor the status of each component of the hardware system, which is convenient for issue trouble-shooting and testing purposes.

## 2.2 Software System

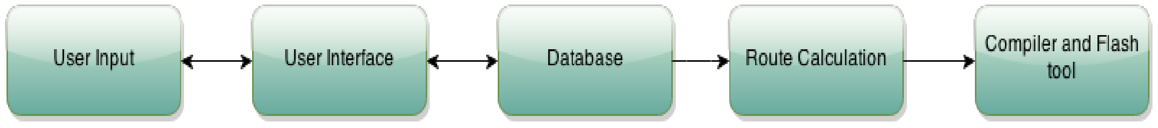


Figure 4 ASG's Software Top-level Flow Diagram

As the Figure 4 shows, the Software part of the ASG includes the graphical user interface, central database server and the flash tool. The user interface provides a user friendly framework for entering original inputs as well as visually displaying necessary information regarding the inputs. The user interface will allow customer to input item names with the virtual keyboard. Once input data is received by UI, the UI will match the item information from database, and show this item information to customers. Meanwhile the database server will deliver coordinates (item location data) to do the route calculation. Once the route calculation is done, route information will be implemented in C code. The C code contains all the moving methods of the robotic car. It controls the two independent motors on the robotic car and handle the collision-detect sensor and ultra—sonic range sensor. The C code will be compiled and converted to machine code by the compiler [1]. Finally, the flash tool will transmit the machine code to the robotic car with the IR-transceiver.

## 3. ASG Hardware

The ASG hardware contains two main parts: robotic car and ultra—sonic ranging sensor. The detailed information will be presented in the following sections.

### 3.1 Robotic Car

This section will present the detailed hardware design information of the robotic car. The goal of this circuit is to establish the infrared connection between robotic car and the computer and use the computer to program the movement of robotic car. When the programed movement is transmitted to the robotic car, the micro-controller of robotic car should be able to control the motors and detect collisions. The schematic diagram is adapted and modified from www.arexx.com and the schematic diagram also shows how the functions are achieved. At the time of writing, all the components (which are sold separately) have been soldered on the PCB board.

At the time of writing this document, the schematic diagram of this circuit has been created using AutoCAD and most figures of this section is taken from the schematic diagram. A copy of schematic can be found in Appendix A.

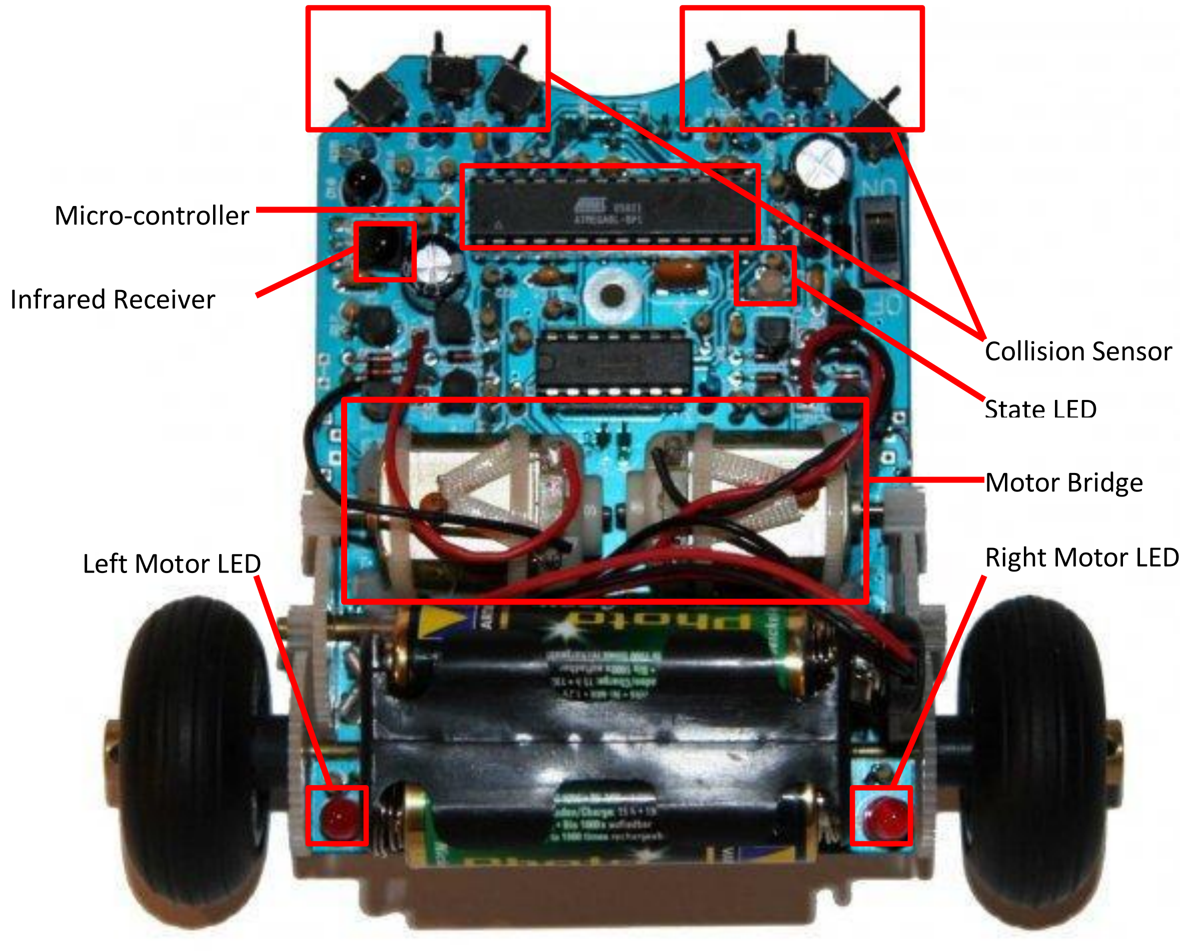


Figure 5 Top view of Robotic Car [2]

As shown in figure 5, the robotic car contains 5 main sub-components and each sub-component provides necessary function for robotic car. These sub-components are as follows:

1. Micro-controller
2. Infrared Receiver
3. Motor bridge
4. Collision Sensor
5. LEDs

The detailed description of the function of each sub-component will be demonstrated in the following sub-sections.

### 3.1.1 Micro-controller

The micro-controller for the robotic car is ATmega8, which is an 8-bit micro-controller based on Harvard architecture and made by Atmel. The block diagram for the micro-controller is shown in figure 6. This micro-controller contains a microprocessor and other functions such as digital I/O ports, ADC and memories.

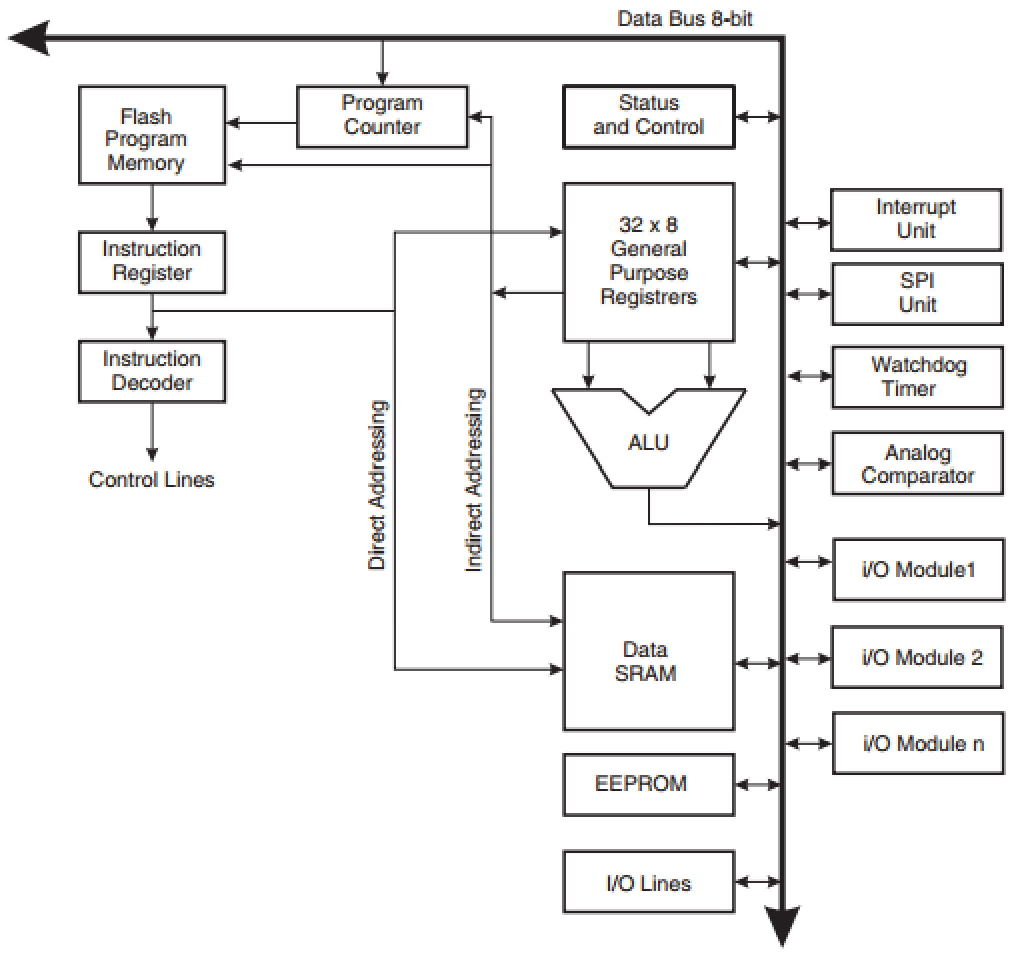


Figure 6 Block Diagram of ATmega8 Micro-Controller [3]

The ADC converts the voltage level to a digital signal and saves the value to RAM. The output value for this ADC has 1024 levels and the output value is calculated as the formula below [3]:

$$ADCout = round\left(\frac{V}{Vreference} * 1024\right)$$

ATmega8 micro-controller contains 3 types of memory: SRAM, flash-memory and EEPROM. The capacity of SDRAM is 1024 Byte and the stored data will be cleared after power off. ATmega8 contains an 8k Byte flash-memory with an endurance of 10,000 write/erase cycles to store the program; however, 1k Byte of the flash-memory is reserved for boot loader, therefore, the available flash-memory is 7k Byte. Additionally, ATmega8 has a 512 Byte EEPROM with an en endurance of at least 100,000 write/erase cycles.

The actual circuit of ATmega8 of robotic car is shown in figure 7.

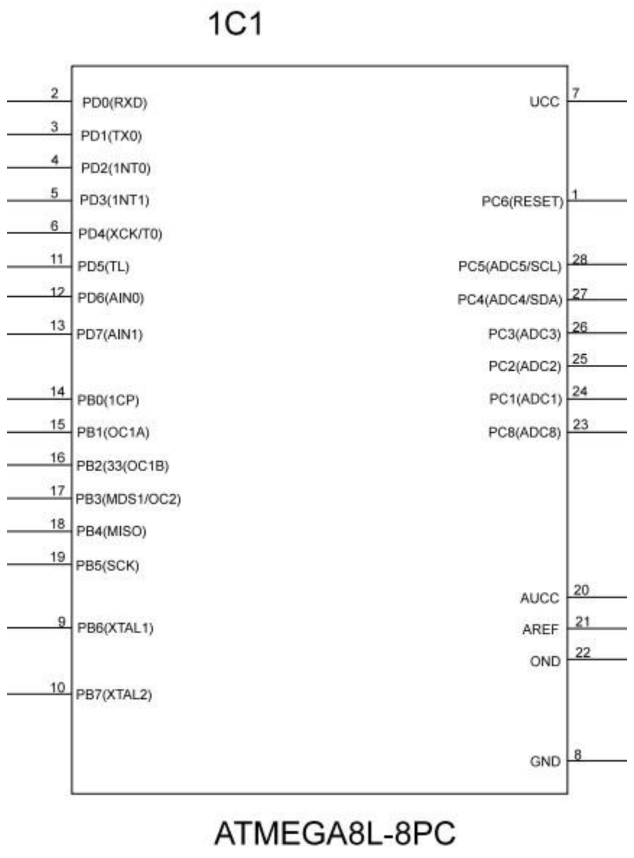


Figure 7 ATmega8 Circuit

### 3.1.2 Infrared Receiver

The infrared connection part contains a SFH5110 IR-receiver to detect, receive and amplify the light signal. The SFH5110 IR-receiver contains 3 pins: Vcc, Output and Ground. The block diagram for SFH5110 is shown below:

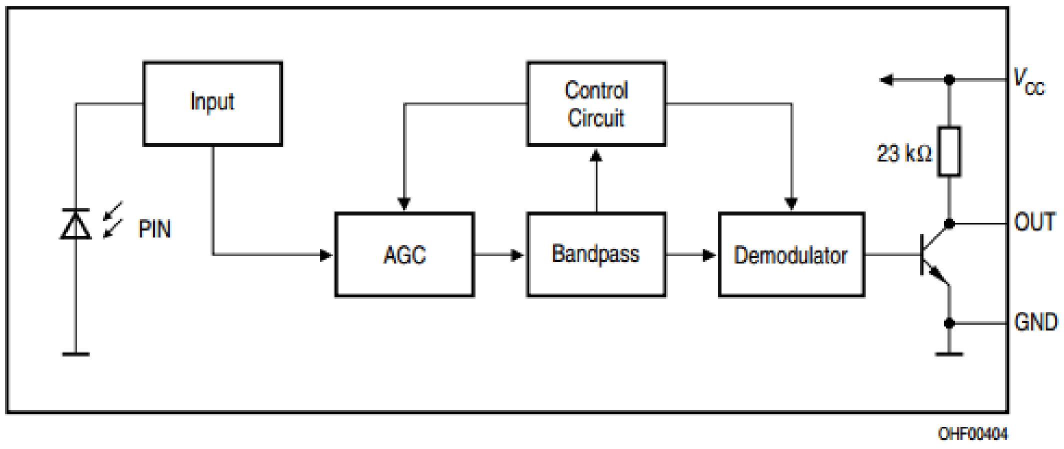


Figure 8 Block Diagram of SFH5110 [4]

The circuit of IR-receiver of robotic car is shown in figure 9; C2 and C8 are used to provide stable voltage for SFH5110 IR-receiver.

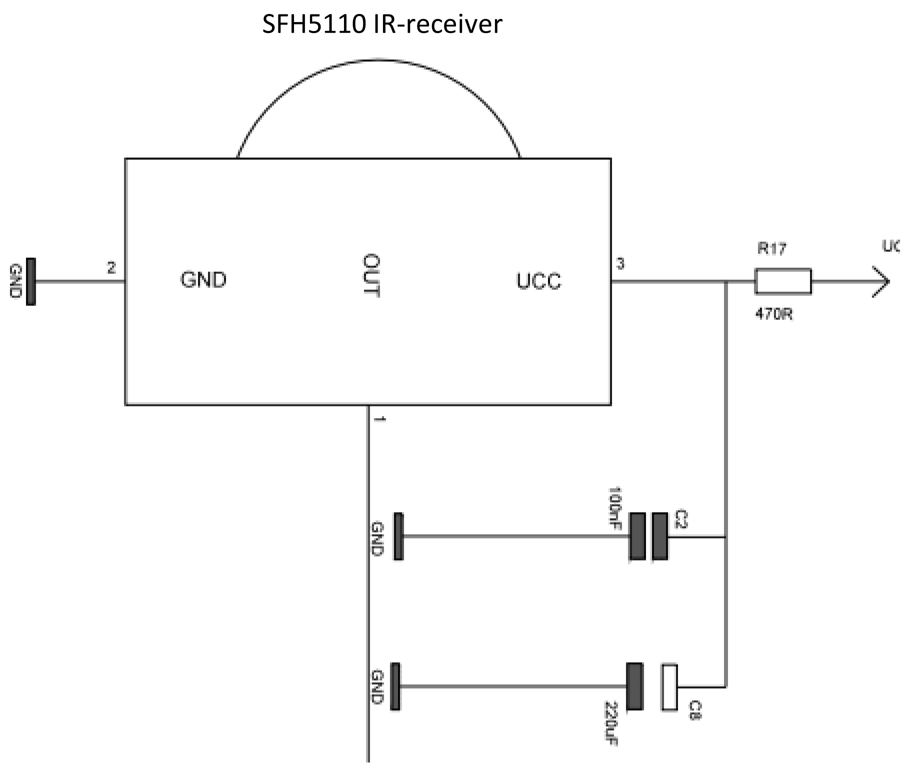


Figure 9 SFH5110 IR-receiver Circuit

On the sender side, a USB-Infrared-transmitter is used to send light signal.

### 3.1.3 Motor Bridge.

The two motors of robotic car are controlled by micro-controller separately. Robotic car can move forward and backward by applying the same voltage to both motors, and robotic car can make a turn by applying different voltages to the motors. In order to control the voltage for each motor, four BJTs are required for each motor to build the motor bridge. The circuit for motor bridge is presented in figure 10.

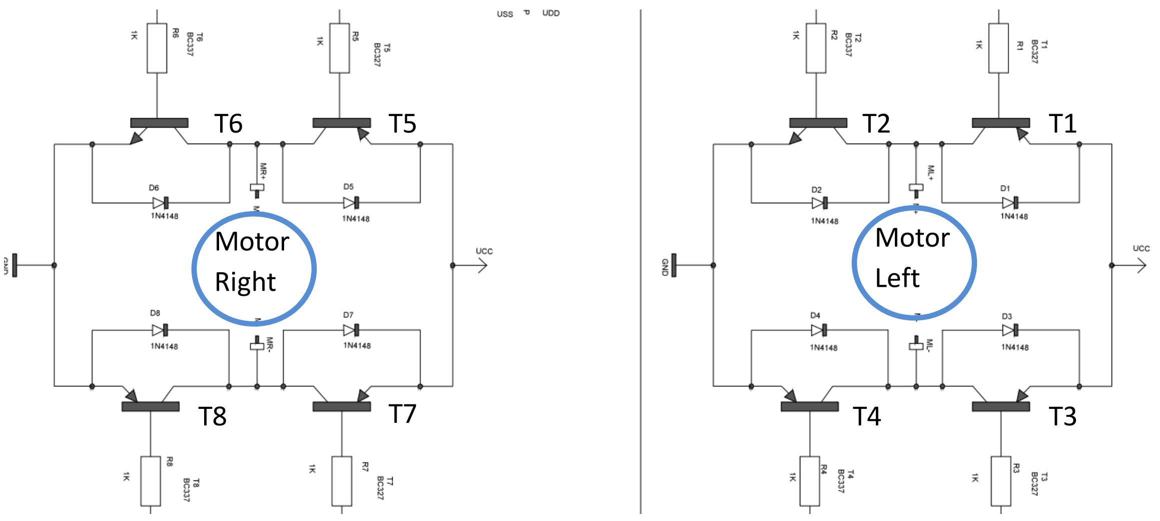


Figure 10 Motor Bridge Circuit

If T1, T4, T6 and T7 are turned on, both motors can make positive rotation, which means the robotic car will move forward; If T2, T3, T5 and T8 are turned on, both motors can make negative rotation and the robotic car will move backward. If robotic car need to make a turn, the microcontroller only turns on the BJTs for one motor.

If the rotation of motors is changed, the inducted voltage that is generated by motors may destroy the BJTs. In order to protect the BJTs of motor bridge, diodes are applied to the BJTs to eliminate the induced voltage.

### 3.1.4 Collision Sensors

This circuit can be found in figure 11. There are 2 major purposes for this circuit. The first one is protect the robotic car and other objects from collision. And the second purpose is adjusting the route of robotic car in order to help the robotic car to find the available path.

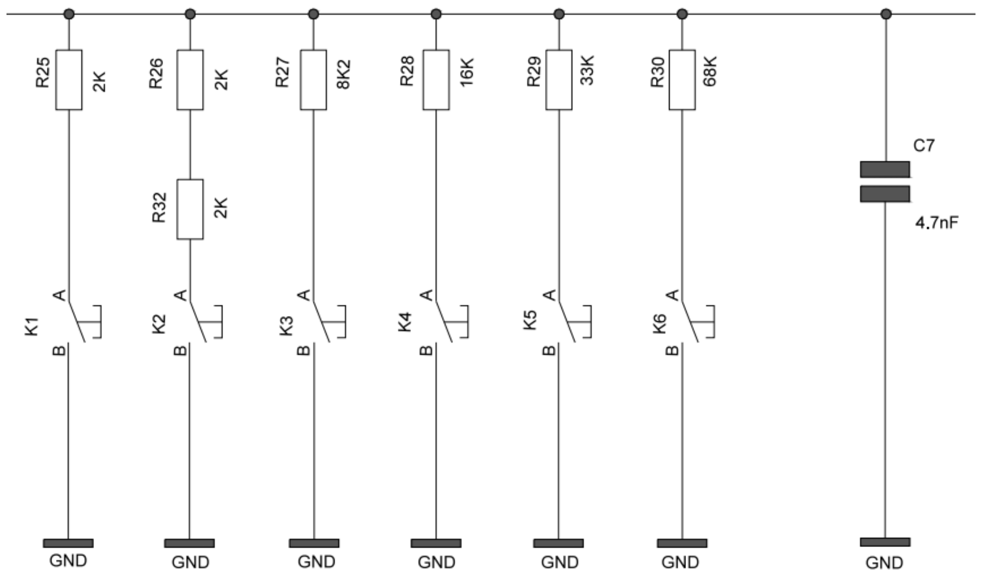


Figure 11 Collision Sensors Circuit

The basic components of collision sensors are switches. When these switches are not active, the capacitor C7 will be charged and the voltage level will be detected as logic high. If robotic car hits something and one of the switches is active, capacitor C7 will be discharged and the voltage level will be detected as logic low. When the logic low signal is detected by the micro-controller, the movement of robotic car will be interrupted.

### 3.1.5 LEDs

The following figure shows the circuit for motor LEDs. LED D15 and D16 are connected to light switches T11 and T12. These light switches can detect the movement of wheels. If the motor is working, the corresponding motor LED will be turned on to indicate the motor state.

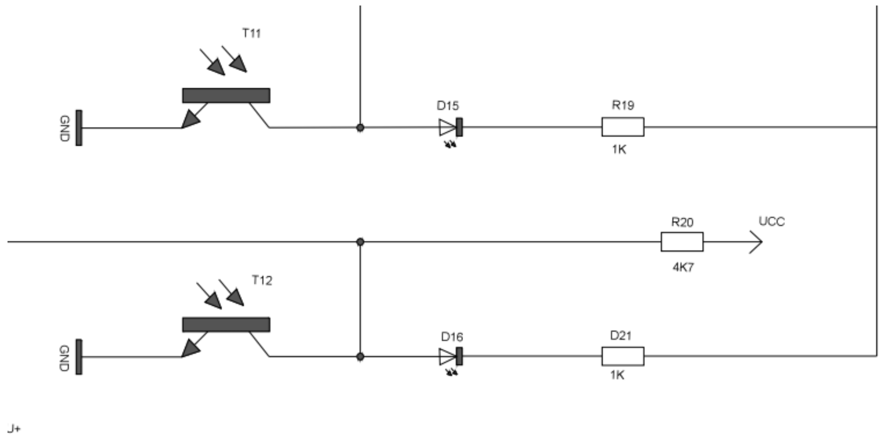


Figure 12 Motor LEDs Circuit

The State LED is controlled by the micro-controller. This LED will flash during the data transmission and it can be programmed to inform user messages. In the original design, robotic car is programmed to flash state LED when robotic car arrives the destination.

## 3.2. Ultra—sonic Ranging Sensor

The ultra—sonic ranging sensor is built on a separate PCB and can be added to the robotic car. The goal of adding ultrasonic sensor to the robotic is to prevent the car from hitting the front barrier and prevent the customer from getting to far away from ASG. The schematic diagram of ultrasonic ranging sensor is adapted from www.arexx.com and can be found in Appendix B. The next steps for the hardware design team include:

1. Transferring the schematic diagram in to PCB layout design
2. Solder all the components on PCB

The ultra—sonic ranging sensor circuit contains two main sub-components:

1. Input voltage regulation
2. Ultra—sonic signal transmitter/receiver

The detailed description of the function of each sub-component will be demonstrated in the following sub-sections.

### 3.2.1 Input Voltage Regulation

The circuit can be seen in figure xx. The circuit uses the voltage divider to reduce the input Vcc by half. After Vcc is divided, an op-amp is applied to make the output voltage stable.

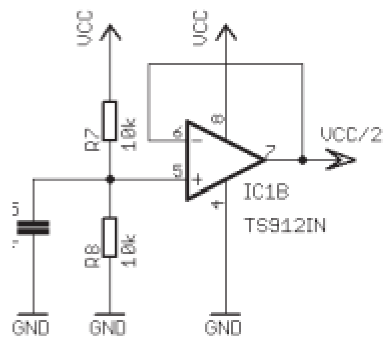


Figure 13 Input Voltage Regulation Circuit [5]

### 3.2.2 Ultra—sonic Signal Transmitter/Receiver

The circuit of transmitter/receiver contains a pair of open face piezoelectric transducers- 40LT12 and 40LR12. The information about ultrasonic signal sending time and receiving time is transmitted to the micro-controller. Based on the time difference, the distance can be calculated. The following figure shows the circuit of Ultrasonic signal transmitter/receiver.

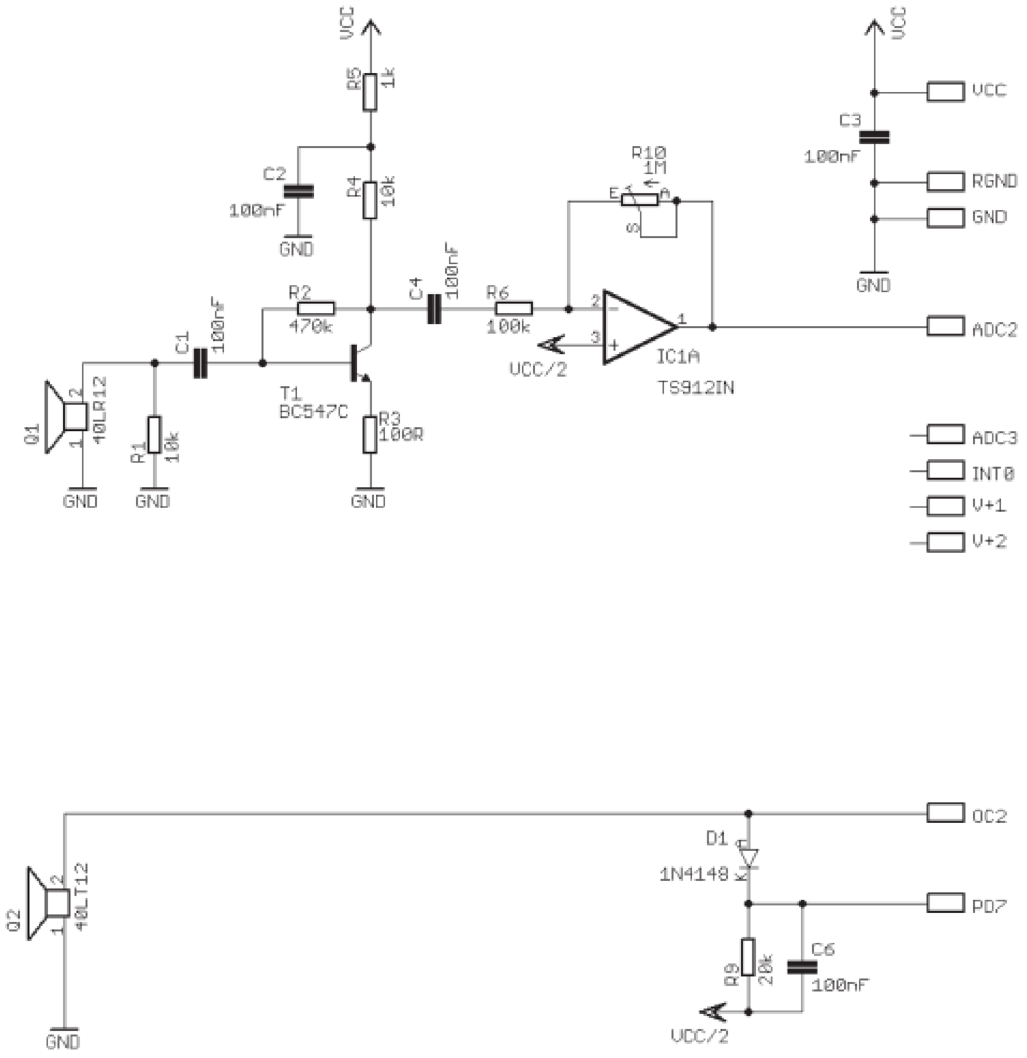


Figure 14 Ultrasonic Transmitter/Receiver Circuit [5]

In the original design, two ultrasonic sensors will be added to the robotic car to detect both front distance and rear distance. However, there is only one set of connection pin for sensors on the robotic car, which means if two sensors are connected at the same time, the output signal will be interfered. Pull-up or pull-down can be added to one of the sensors to solve the signal interference; the verification of design is still under process.

## 4. ASG Software

The ASG software contains two main parts: User Interface & Database and compiler & flash tool. The detailed information will be presented in the following sections.

### 4.1 User Interface and Database

The graphical user interface primarily provides a user friendly framework for entering original inputs as well as visually displaying necessary information with regard to these inputs. The general functionality flowchart involving the user interface is demonstrated via the following figure.

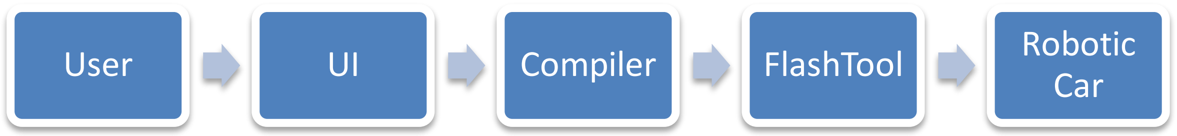


Figure 15 General Functionality Flowchart

The layout of the user interface is designed to be concise, perspicuous and comprehensive allowing users at all ages to quickly start with the operation without receiving additional technical trainings. The expected final design of the layout is shown in the following figure.

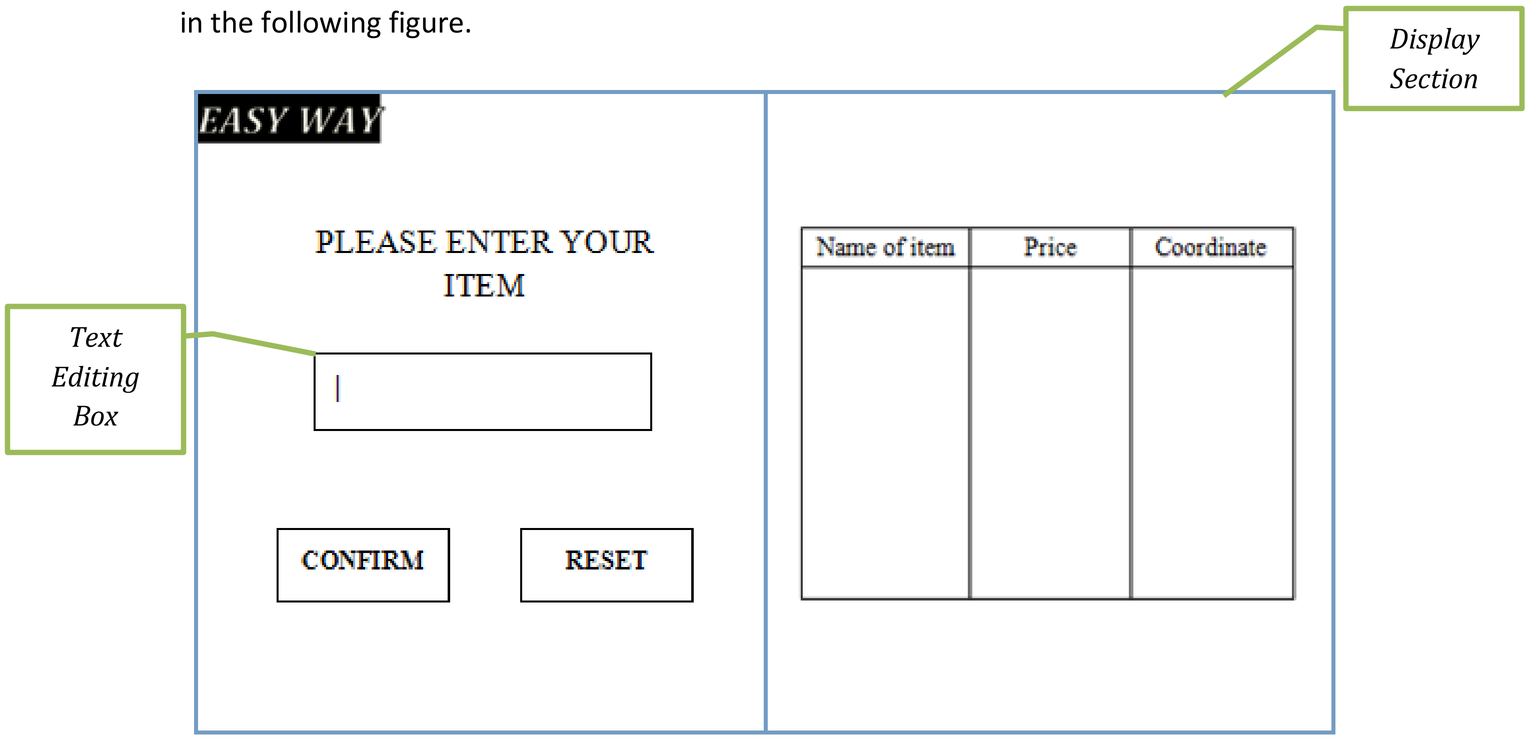


Figure 16 User Interface Design

The entire layout is divided into two sections. The section on the left is user operation section and the one on the right is the display section. The text editing box in the middle of the left section is where users enter their desired items. After manually entering their first item's name, users will press the "CONFIRM" button for the system to store the input and load the information from the database. The system will prompt message window "START GUIDING" to notify users about the running operation. The expected design of prompt is shown in the following figure.

Figure 17 "Start Guiding" Prompt

Then the loaded information including the name, price, and destination coordinates will be displayed inside the table of the right side section. Meanwhile, the system will calculate the output coordinates by using the destination coordinates minus the initial coordinates, and use the output coordinates as the input of the compiler for robotic programming module. The detailed design flowchart regarding I/O processing of the user interface is shown below in the figure.

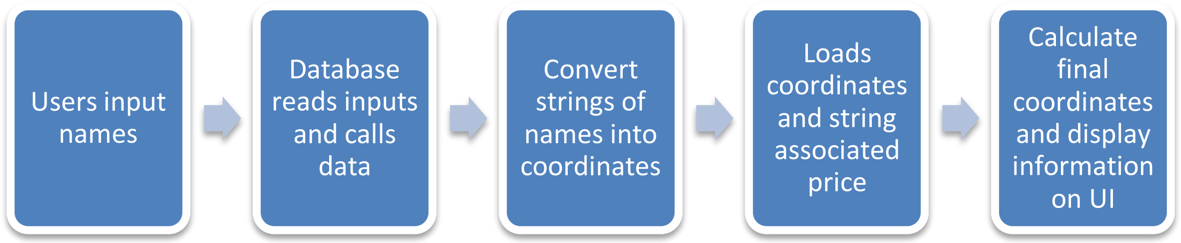

Users input names
Database reads inputs and calls data
Convert strings of names into coordinates
Loads coordinates and string associated price
Calculate final coordinates and display information on UI

Figure 18 User Interface I/O Processing Flowchart

Additionally, the previous items one user has inputted will have their information displayed throughout this user's entire shopping trip.

After a user complete his/her entire shopping trip, he/she could return back to the initial point (0, 0) by entering the item name as "0" and pressing "CONFIRM", as well, reset the user interface to its original state without any information displayed by clicking the "RESET" button for the next user to start.

If the item that user inputs is not available in the database or the name contains spelling mistakes, the system will prompt another message window indicating that the user will have to re—enter the item. The expected design of prompt is shown below in the figure.

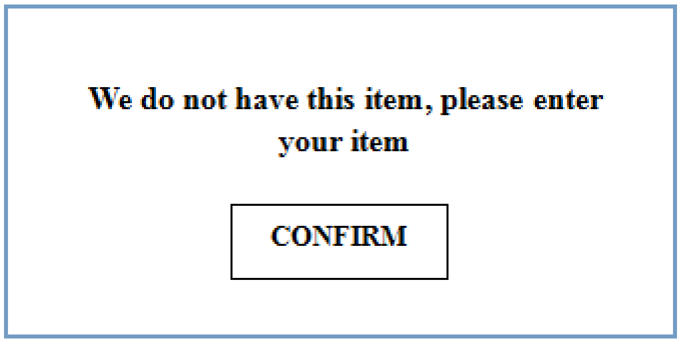


Figure 19 Invalid Input Prompt

In order to create a user interface with compatibility and usability, we are in the process of designing two user interfaces with the same layout and functionality but via two different approaches. One is using HTML to design a web based UI, using JavaScript for coordinates calculation and using PHP to design the database. The other one is using C# for the user interface and SQL server for data access.

The designed database will be embedded within the main structure of the user interface to store the necessary information of all the items. As it is shown in the previous design flowchart, after users inputting the names, the database will read the inputs and calls the stored data.

Only data that matches with the information inputted will be loaded for further calculation and outputting. And similarly, only when the information inputted by users matches the data stored in the database will trigger the loading process.

An example of data stored in the database that will be used for our proof—of—concept prototype is demonstrated in the following table.

| Item Name | Price | Destination Coordinates |
|---|---|---|
| Apple | $1.99/lb. | (1, 1) |
| Milk | $2.99/gal | (1, 3) |
| Noodle | $1.99/kg | (3, 1) |
| Chili Sauce | $5.99/bag | (3, 3) |
| Cabbage | $0.99/kg | (2, 0) |
| Chocolate | $9.99/box | (3, 0) |
| Vacuum Cleaner | $199.99 | (0, 3) |
| Xbox One | $399.49 | (4, 1) |
| Mac Pro | $1399.00 | (4, 2) |
| Dog Food | $6.99/bag | (4, 4) |
| 0 | N/A | (0, 0) |

**Table 1 Information Stored in Database**

The coordinates in the table are destination coordinates which will be used to calculate output coordinates. The information for each item will be displayed individually on the user interface.

As the above information is stored in the database, and firstly the user inputs the first item "Apple" within the user interfaces. The database will read the input and call the matching information including the destination coordinates, (1, 1). The system will set the first initial coordinates as (0, 0) by default. Then the output coordinates will be (1, 1) – (0, 0) = (1, 1). That will be used as the input of the WinAVR compiler to calculate the route of the robotic car. Following by the second item "Milk", destination coordinates (1, 3) will be called and the output coordinates will now be (1, 3) – (1, 1) = (0, 2). The rest may be deduced by analogy. The destination coordinates of the user's entire shopping trip could be numerically illustrated below in the figure.

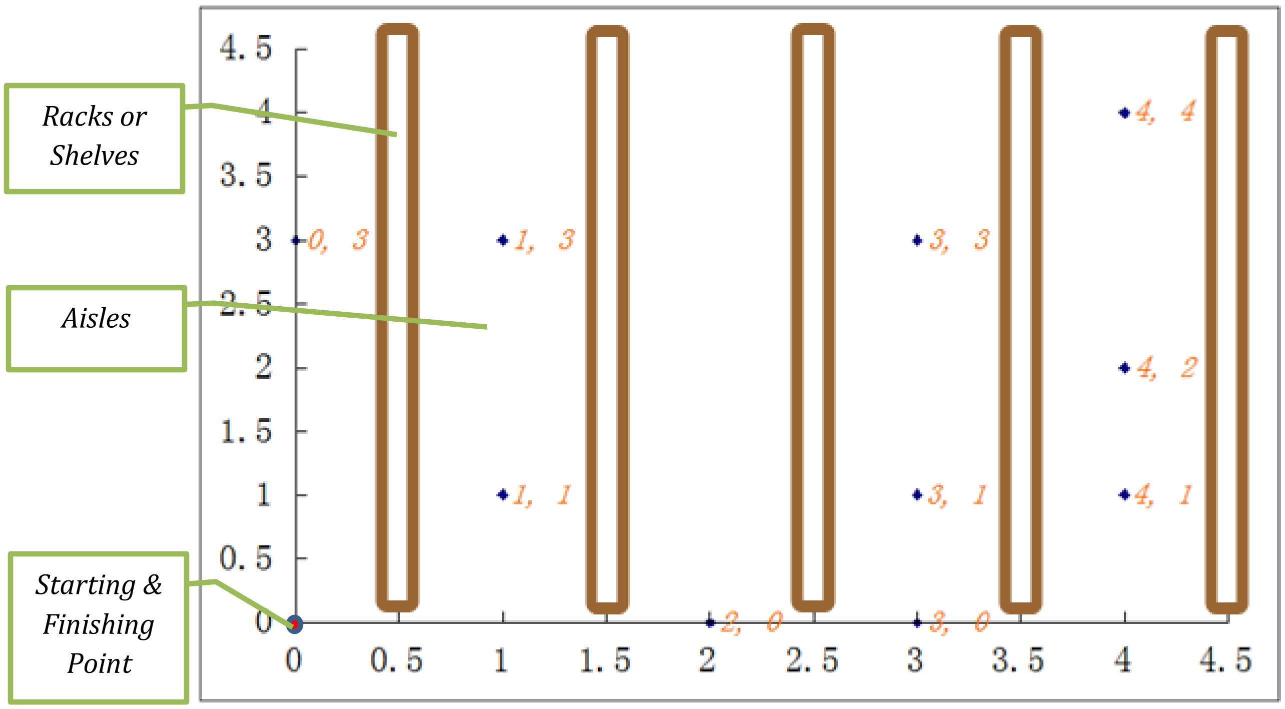


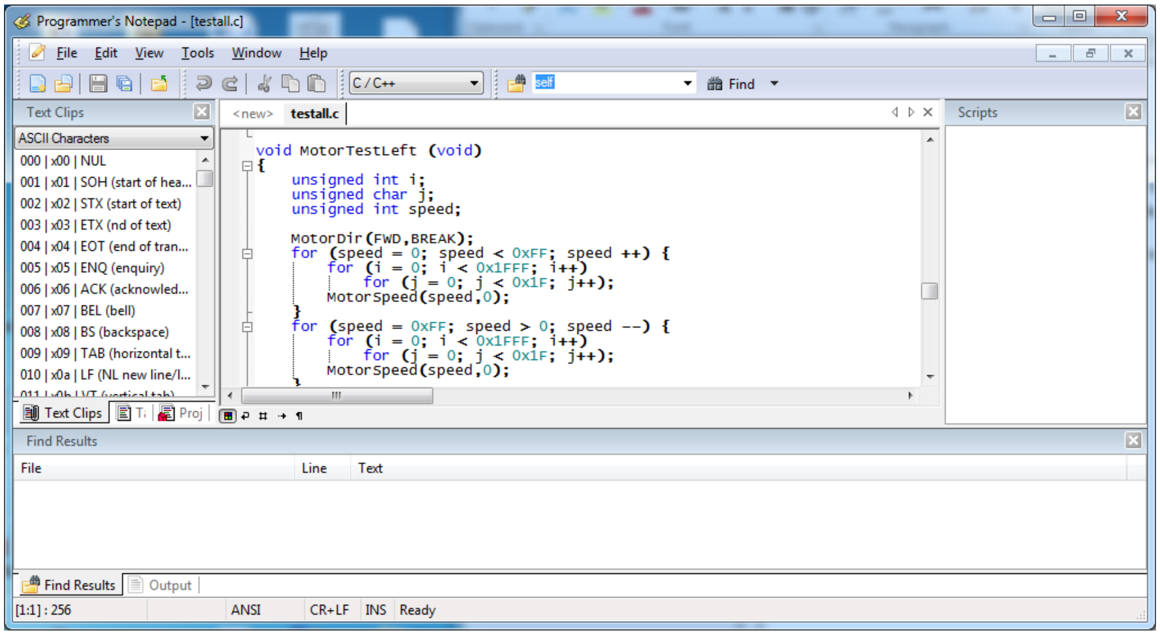
Figure 20 Destination Coordinates in Plane Coordinate System

The figure presents that the destination coordinates will only appear in aisles and will not overlap the brown bars, which are racks and shelves.

As it is stated previously, the first initial coordinates is set to (0, 0) in the robotic programming by default. In actual layout of the supermarket, the first initial point will be the front entrance only. Any other starting points will generate a complete different set of initial coordinates and confuse the system. With the awareness of this issue, we hope to resolve it and improve system usability in the future.

## 4.2 Compiler and flash tool

Our compiler is called WinAVR. It is an open source software development tools for Atmel AVR series of microcontrollers. WinAVR includes the GNU GCC compiler for C and C++. We will use the compiler convert the C code to machine code.



```
000 | x00 | NUL
001 | x01 | SOH (start of hea...
002 | x02 | STX (start of text)
003 | x03 | ETX (nd of text)
004 | x04 | EOT (end of tran...
005 | x05 | ENQ (enquiry)
006 | x06 | ACK (acknowled...
007 | x07 | BEL (bell)
008 | x08 | BS (backspace)
009 | x09 | TAB (horizontal t...
010 | x0a | LF (NL new line/l...
```

```
void MotorTestLeft (void)
{
    unsigned int i;
    unsigned char j;
    unsigned int speed;

    MotorDir(FWD,BREAK);
    for (speed = 0; speed < 0xFF; speed ++) {
        for (i = 0; i < 0x1FFF; i++)
            for (j = 0; j < 0x1F; j++);
        MotorSpeed(speed,0);
    }
    for (speed = 0xFF; speed > 0; speed --) {
        for (i = 0; i < 0x1FFF; i++)
            for (j = 0; j < 0x1F; j++);
        MotorSpeed(speed,0);
```

```
[1:1] : 256    ANSI    CR+LF    INS    Ready
```

Figure 21 Compiler Interface

The figure shows the compiler user interface. The c code can be edit with the build-in editor. The C code showed in the figure is partial code of motor test [6]. As we can see in the figure, the function control the motor rotation direction and rotation speed. We can change the speed by editing the integer 'speed'. For more details about the source code, please see the Appendix C.

The flash tool will transfer the machine code through IR- transceiver to the IR-receiver on the robotic car. The IR-receiver shall be placed in the transmit range 50cm with IR-transceiver. The transition will be stopped if the IR-receiver is out of range. For future design, we will expand the range of transition.

# 5. USTAINABILITY AND SAFETY

## 5.1 Safety and Reliability

Any electronic devices shall strictly follow corresponding safety regulations. According to our previous functional specification, the proof—of—concept prototype and final production device has been designed to conform to the standards listed in the following table:

| | | |
|---|---|---|
| [R16—B] | ISO 4141—1:2005 | The standard specifies the test methods and requirements of basic performance multi—core connecting cables for road vehicles, suitable for a temperature range of -40 °C to +85 °C. [7] |
| [R17—B] | ISO/IEC 10536—3:1996 | The standard specifies the nature and characteristics of the fields to be provided for power and bidirectional communications between devices and contactless integrated circuit cards. [8] |
| [R18—B] | ISO/AWI 11898—1 | The standard specifies the data link layer and physical signaling of the controller area network. [9] |

**Table 2 Required Standards**

In terms of our proof—of—concept prototype only, all the electronic components will be firmly soldered on our designed PCB for minimizing the chance of component failure due to tumbling and collisions. The power supply of the hardware that is a set of four AAA batteries will be placed in a multi—cell box. As the voltage provided by this power supply is approximately 6V, and the multi—cell box will be steadily mounted on the PCB, we could ensure that the prototype will not cause any harm to human bodies due to electric leakage. The prototype is also designed to be highly modular as each module is in charge of a different functionality. Hence, if one of the module fails its functionality due to damage, it will not bring significant impact on other modules. Another advantage of this modular design is that it simplifies the debugging procedures once any failures occur. By examining the specific functionality failure and identifying the failing module, the costs of the entire debugging and fixing process are dramatically reduced.

## 5.2 Sustainability

Easy Way Inc. believes that the way any products are made could be a huge impact on the quality of life just as the way they function. Hence, our design team has included the sustainability of our Auto Shopping Guide System in the priority list throughout the development and implementation phase. Our proof—of—concept prototype has been designed to be modular in terms of hardware device. On top of its functionalities, the prototype has been developed with minimized possible waste and toxic materials. For the robotic car module, all the electronic components were obtained from team members' previous projects or laboratory work except the microcontroller. For the microcontroller, we chose the Atmel AVR 8—bit microcontroller which is programmable through open source software development tools. And hence, it could be re—used for any other related projects. The motors and rubber wheels were taken from broken toy cars. The entire module is powered on four standard AAA battery cells that are rechargeable. Every piece of the robotic car module is re-usable and could be easily unsoldered and stored for future usage. The sensor module is considered as an additional function of the robotic car module. It could be simply installed and uninstalled without influencing the primary functionalities of the robotic car module. Aside from recyclability and reusability, the electronic components are also highly replaceable.

## 6. System Test Plan

In order to ensure the system will execute properly, the system test plan has to be setup. This session demonstrates that the system testing will not only cover individual modules testing, but also combined modules testing. Both testing methodologies will carry out from different levels with an emphasis on the requirements of the proof-of-concept version. All possible situations will be discussed in this session.

### 6.1 Individual Module Testing

Individual module testing consists of three main aspects: mobile system testing, data transmit testing and robotic car testing.

#### 6.1.1 Mobile System Testing

The mobile system contains the user interface and the database server as Figure 2 and Figure 16 shown. There are three main purposes of mobile system testing. The first test objective is to ensure that the database server is able to properly store and be updated all items' information. The second test objective is that all user interface functions behave as expected. The third test objective is to check if the route is calculated correctly in the database server.

##### *6.1.1.1 Database Server Testing*

Database is an organized collection of data. It stores all items details such as, items' names, price and location. When tester launches the database, the database should clearly show the details. Also, database shall allow tester add items and update the data on database server.

##### *6.1.1.2 User Interface Functions Testing*

The user interface is an effective operation and control of the machine between user and the machine. The tester shall confirm that all interface functions work as expected.

1. Text-editing box: Tester shall type an item in the box as Figure 16 shown.
2. Confirm button: One of the main functions of this button is checking if there is the item from the database. If the item cannot be found from database, error message shall be popped up as Figure 19 shown. Otherwise, another function of the confirm button will be run that is listing the item on the right side of the user interface which is user operation section as Figure 16 shown. A command route calculation is delivered to database server is another important function of this button. As a result, tester will need open the code to check if the calculation function is working or not.

3. Reset button: The major function of "Reset" is reset or clear the user interface so that user can start another shopping trip. Also, the user operation section should be empty when click on the Reset button.
4. User operation section: It shows the items that user already input and confirm on the text-editing box.

#### *6.1.1.3 Route Calculation Function Testing*

Route Calculation function plays an important role in the whole system. It calculates and gives a result how to reach the item from one point. Tester shall check the calculation result from the code if the result is same as expected.

### 6.1.2 Data Transmit Testing

Data transmit is established based on the wireless communication between Mobile system and the receiver of robotic car. The major purpose of Data Transmit testing is to ensure the result of route calculation can be delivered from user interface and received by robotic car successfully within certain range. The Data Transmit Testing Table shows the possible situations during the communication between Mobile system and Robotic car. In order to execute the following test cases properly, the prerequisite has to be completed first in each test case. The prerequisite is there is a valuable route file in Mobile system.

| Test Cases | Steps | Expected Result |
|---|---|---|
| Test Case 1. Data Transmit within 10CM | 1. Place mobile system and robotic car in 10 CM<br>2. Allow the route data transmitted from Mobile system to robotic car<br>3. Wait until the route data transmitted completed. | 2. When the data is being transmitted, the LED light shall be lighted up constantly in orange.<br>3. The LED light shall be lighted up constantly in green instead of orange. |
| Test Case 2. Data Transmit out of range 10CM | 1. Place Robotic car far away from mobile system out of range 10CM<br>2. Allow mobile system transmit the data to Robotic car | 2. There is no any signal or feedback shown from Robotic car or user interface.<br>The supposed result is the user interface shows message "detecting the target". If the user interface is unable to detect the robotic car in 10 seconds, the data transmit request shall be paused. |

| | | |
|---|---|---|
| | | And the user interface will be back to previous status. |
| Test Case 3. Data Transmit get interrupted | 1. Place mobile system and robotic car in 10 CM<br>2. Allow the route data transmitted from Mobile system to robotic car.<br>3. During the route data is being transmitted, put the robotic car away Mobile system | 2. The LED is lighted up in orange during the data transmitted.<br>3. The data transmitted process shall get paused. And user interface will be back to previous status. However, the supposed result is the process gets paused. And a message "data transmitted is interrupted" is popped up on user interface. And there are two options for user. A. continue B. cancel. If user A. choose continue option, the rest of the data shall be transmitted when the robotic car is placed to mobile system in 50CM. Otherwise, if user chooses B. cancel, the user interface will be back to previous status. |

Table 3 Data Transmit Testing

### 6.1.3 Robotic Car Testing

Significant testing shall be performed on the Robotic Car testing of the project. Tester will conduct unit tests on the following functional blocks, with the pass criteria specified below:

- Left and Right Engines: tester shall load the motor test file to the robotic car. And tester shall hold the robotic car and observant both sides wheels when running the engine testing. First, the left motor is started in forward direction from zero speed to maximum speed and it will run in back direction with maximum speed to zero. Also, the right motor shall be test the same way. After the single engines checked, tester shall test both engines together. Both motors are started in forward direction with speed, then in back direction from maximum speed to zero speed.

- Sensors: tester shall load the sensor test file to the robotic car. The file will order the car to keep running. When it is running, put the hand in front of the robotic car, both engines shall stop running until the hand is removed.
- LED display-elements: there three LED lights were assembled on the robotic car. Two of them are located next to the left and right engines. So when the engine is working, the corresponding LED shall be lighted. The other LED shall signal that the robotic car is receiving data or has completed data receiving.

## 6.2 Combined Modules Testing

The combined modules testing will ensure the whole system works properly.

1. Tester will input and confirm an item on the user interface.
2. Then transmit the route calculation data to the robotic car through IR interface. The LED signal shall be lighted in green once the robotic car has received the data completely.
3. Tester shall place the robotic car at the starting point (0, 0) as Figure 20 shown
4. When the robotic car is guiding user to one point, tester shall put some barriers in front of the robotic car.
5. Tester shall input and confirm the destination “0” in the text-editing box to complete the shopping trip.
6. Tester shall click on “Reset” button to clear the user interface

## 7. Conclusion

This document has outlined all the main design considerations that ASG system intends to follow throughout its development and test phases. These specifications will be adhered to as much as possible during the development and implementation to meet the functional specifications. Some of the hardware and software remain to be completed and might require changes and adjustments in order to complete the proof—of—concept prototype. The integration of the system also remains to be completed and tested while most of the hardware assembly and part of the software development are done. Through the provided test plans, the required functionalities of ASG system will be ensured. The primary function of ASG system, that is being able to guide users to targetted locations is clearly discussed in this design specification.

## 8. Reference

[1] *WinAVR*. (n.d). Retrieved from WinAVR compiler: http://www.webring.org/l/rd?ring=avr;id=59;url=http%3A%2F%2Fwinavr%2Esourceforge%2Enet%2F

[2] *ASURO Robot Kit.* (2004). Retrieved from Assembly and Operation MANUAL: http://www.arexx.com/downloads/asuro/asuro_manual_en.pdf

[3] *Atmel.* (2013, February ) Retrieved from ATmega8L Data Sheet: http://www.atmel.ca/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf

[4] *IR-Receiver for Remote Control System* (2006, September 6). Retrieved from SFH5110 Data Sheet (PDF): http://docseurope.electrocomponents.com/webdocs/08b4/0900766b808b4438.pdf

[5] *AREXX* (n.d). Retrieved from ASURO Ultrasonic ULT-10: http://www.produktinfo.conrad.com/datenblaetter/175000-199999/191360-as-01- de-ASURO_ULTRASCHALL_SATZ_FUER_ARX_03.pdf

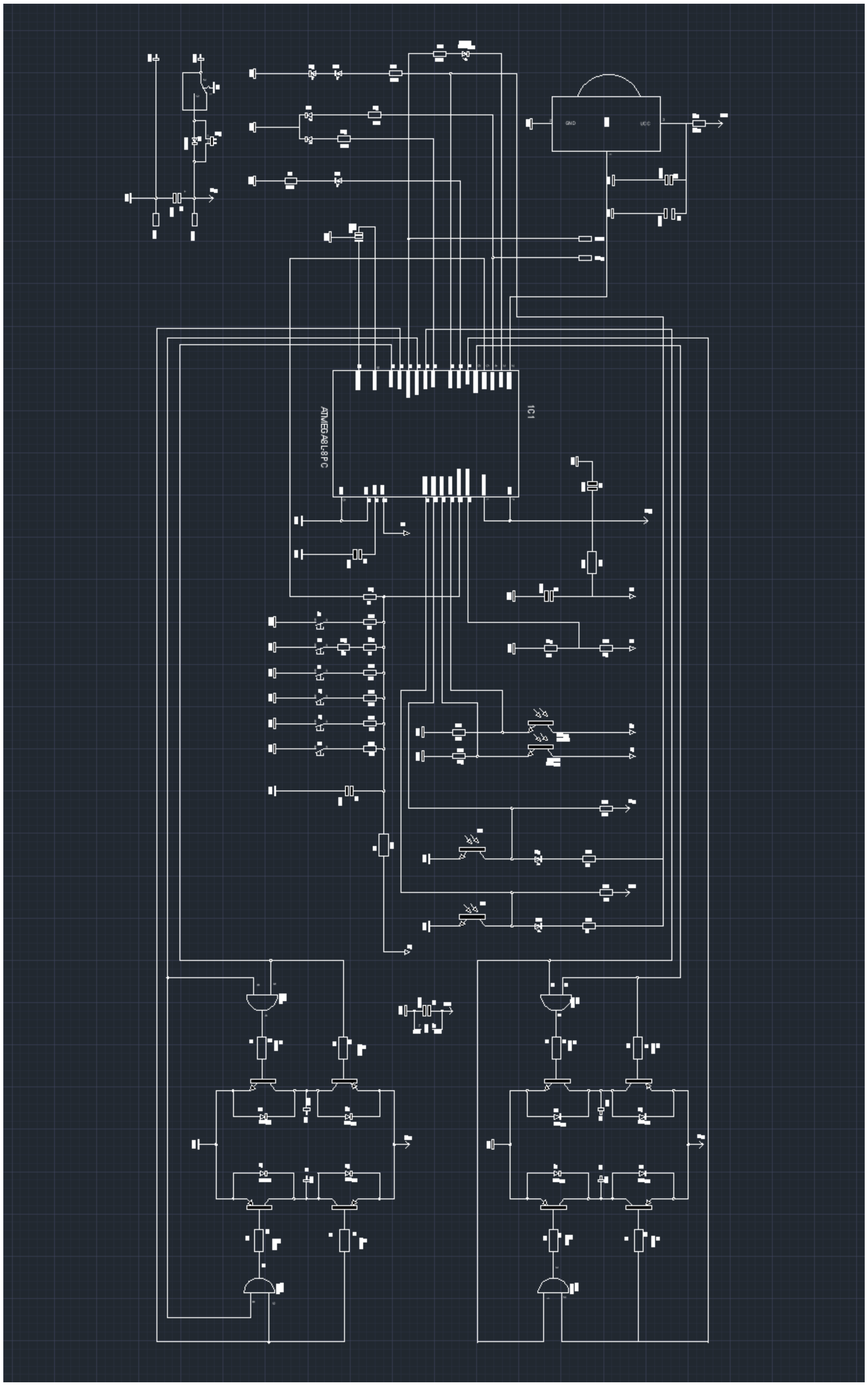
[6] *AREXX.* (n.d). Retrieved from ASURO download: shttp://www.arexx.com/arexx.php?cmd=goto&cparam=p_asuro_downloads -

[7] *Electrical and electronic equipment*. (2006, January). Retrieved from BS ISO 4141-1:2005 Road vehicles -- Multi-core connecting cables -- Part 1: Test methods and requirements for basic performance sheathed cables: http://shop.bsigroup.com/ProductDetail/?pid=000000000030123174

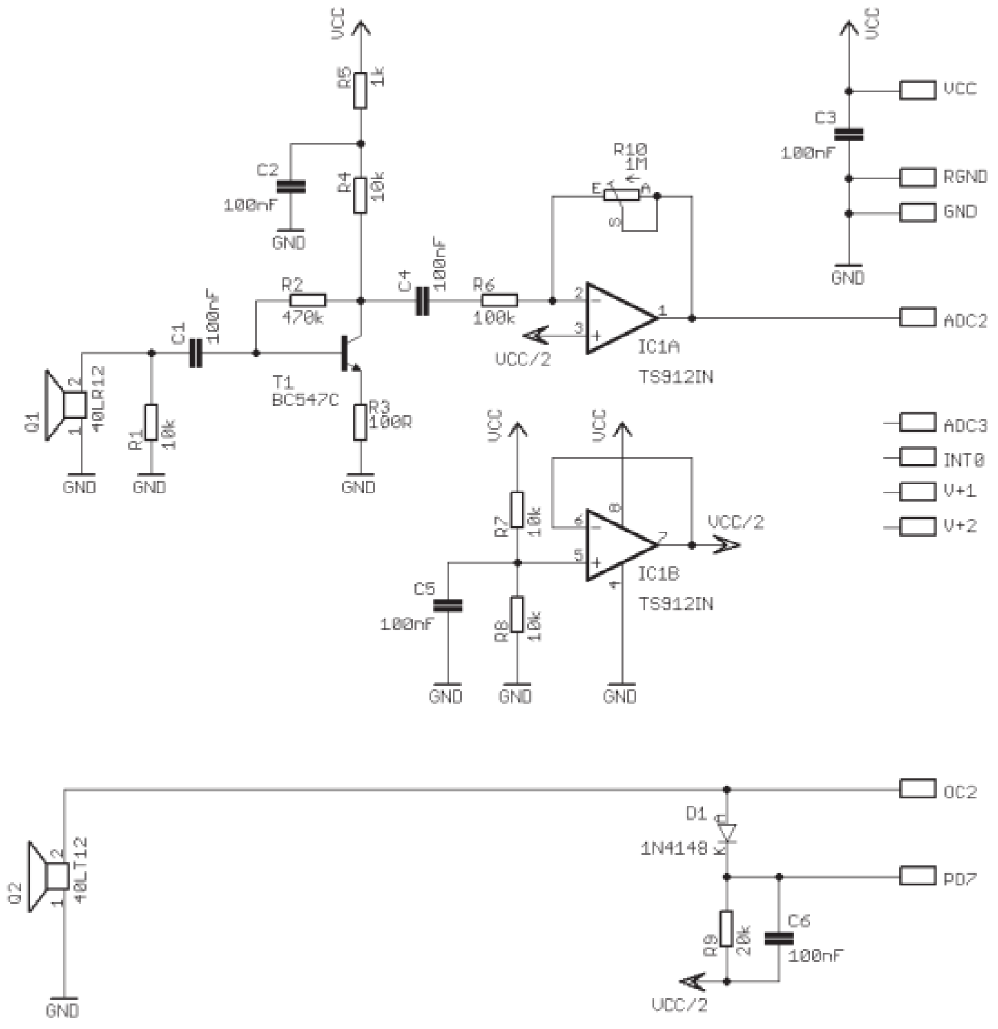
[8] *ISO Standards.* (n.d.). Retrieved from ISO/IEC 10536-3:1996 Identification cards – Contactless integrated circuit(s) cards -- Part 3: Electronic signals and reset procedures:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24184

[9] *ISO Standards.* (n.d.). Retrieved from ISO/TC 22/SC 3 Electrical and electronic equipment:http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/iso_technical_committee.htm?commid=46752

## APPENDIX A – ROBOTIC CAR SCHEMATIC DIAGRAM



## APPENDIX B – ULTROSONIC SENSOR SCHEMATIC DIAGRAM



## APPENDIX C – TEST CODE FOR ASG

```
#include "asuro.h"

/* ---------------------------------------- */
/* -------------- Serial Test -------------- */
/*   ASURO sends recieved data+1 directly    */
/* when TIMEOUT occures ASURO is sending 'T' */
/* ---------------------------------------- */
void SerialTest(void)
{
        unsigned char data;
        unsigned char i;
        for (i = 0; i < 0xFE; i++) {
                StatusLED(GREEN);
                SerRead(&data,1,0xFFFE);
                StatusLED(RED);
                if (data != 'T') data += 1;
                SerWrite(&data,1);
        }
}
/* END Serial Test ------------------------- */

/* ---------------------- */
/* ----- Switch Test ---- */
/* K1 -> Status LED Green */
/* K2 -> Status LED RED   */
/* K3 -> Line LED         */
/* K4 -> Break LED Left   */
/* K5 -> Break LED Rigth  */
/* K6 -> Motor Left       */
/* ---------------------- */
void SwitchTest(void)
{
        unsigned char sw,tmp;
        MotorDir(FWD,BREAK);

        sw = PollSwitch();
        StatusLED(OFF);
        FrontLED(OFF);
        BackLED(OFF,OFF);
        MotorSpeed(0,0);
        tmp = 0;

        if (sw & 0x01)
                MotorSpeed(200,0);
        if (sw & 0x02) {
                BackLED(OFF,ON);
                tmp = ON;
        }
        if (sw & 0x04)
                BackLED(ON,tmp);
        if (sw & 0x08)
                FrontLED(ON);
```

```
            if (sw & 0x10)
                        RED_LED_ON;
            if (sw & 0x20)
                        GREEN_LED_ON;
}
/* END Switch Test ------------------------------ */

/* -------------------------------------------- */
/* ------------ Line Sensor Test ---------------- */
/* Left Sensor -> Left Break LED ON when Light on
   Left Phototransistor bright enough            */
/* Right Sensor -> Right Break LED ON when Light on
   Right Phototransistor bright enough           */

void LineTest(void)
{
            unsigned int data[2];
            unsigned char tmp[2] = {OFF,OFF};

            LineData(data);
            if (data[0] > 400)
                        tmp[0] = ON;
            if (data[1] > 400)
                        tmp[1] = ON;
            BackLED(tmp[0],tmp[1]);
}
/* END Line Sensor Test ------------------------- */

/* -------------------------------------------- */
/* ---------- Odometrie Sensor Test ------------- */
/* Left Sensor -> Status LED GREEN ON when Light on
   Left Phototransistor bright enough            */
/* Right Sensor -> Status LED RED ON when Light on
   Right Phototransistor bright enough           */
void OdometrieTest(void)
{
            unsigned int data[2];

            OdometrieData(data);
            StatusLED(OFF);
            if (data[0] < 512)
                        StatusLED(GREEN);
            if (data[1] < 512)
                        StatusLED(RED);
}
/* END Odometrie Sensor Test ------------------- */

/* -------------------------------- */
/* ---------- Motor Test ----------- */
/* Motor Left forward to max. speed  */
/* Motor Left rewind  to max. speed  */
/* Motor Right forward to max. speed */
/* Motor Right rewind  to max. speed */
/* Both Motors forward to max. speed */
/* Both Motors rewind to max. speed  */
void MotorTestRight (void)
{
        unsigned int i;
        unsigned char j;
        unsigned int speed;

        MotorDir(BREAK,FWD);
        for (speed = 0; speed < 0xFF; speed ++) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(0,speed);
        }
        for (speed = 0xFF; speed > 0; speed --) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(0,speed);
        }

        MotorDir(BREAK,RWD);
        for (speed = 0; speed < 0xFF; speed ++) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(0,speed);
        }
        for (speed = 0xFF; speed > 0; speed --) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(0,speed);
        }
}

void MotorTestLeft (void)
{
        unsigned int i;
        unsigned char j;
        unsigned int speed;

        MotorDir(FWD,BREAK);
        for (speed = 0; speed < 0xFF; speed ++) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(speed,0);
        }
        for (speed = 0xFF; speed > 0; speed --) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
                MotorSpeed(speed,0);
        }

        MotorDir(RWD,BREAK);
        for (speed = 0; speed < 0xFF; speed ++) {
                for (i = 0; i < 0x1FFF; i++)
                        for (j = 0; j < 0x1F; j++);
```

```
			MotorSpeed(speed,0);
		}
		for (speed = 0xFF; speed > 0; speed --) {
			for (i = 0; i < 0x1FFF; i++)
				for (j = 0; j < 0x1F; j++);
			MotorSpeed(speed,0);
		}
}

void MotorTestBoth (void)
{
		unsigned int i;
		unsigned char j;
		unsigned int speed;

		MotorDir(FWD,FWD);
		for (speed = 0; speed < 0xFF; speed ++) {
			for (i = 0; i < 0x1FFF; i++)
				for (j = 0; j < 0x1F; j++);
			MotorSpeed(speed,speed);
		}
		for (speed = 0xFF; speed > 0; speed --) {
			for (i = 0; i < 0x1FFF; i++)
				for (j = 0; j < 0x1F; j++);
			MotorSpeed(speed,speed);
		}

		MotorDir(RWD,RWD);
		for (speed = 0; speed < 0xFF; speed ++) {
			for (i = 0; i < 0x1FFF; i++)
				for (j = 0; j < 0x1F; j++);
			MotorSpeed(speed,speed);
		}
		for (speed = 0xFF; speed > 0; speed --) {
			for (i = 0; i < 0x1FFF; i++)
				for (j = 0; j < 0x1F; j++);
			MotorSpeed(speed,speed);
		}
		MotorSpeed(0,0);
}

void MotorTest(void)
{
		MotorTestLeft();
		MotorTestRight();
		MotorTestBoth();
}
/* ---------------------------------------------- */

/* ------------- LED Test ----------- */
/* Switch On and Off all visible LEDs */
/* Status LED GREEN,RED              */
/* Line LED                                                  */
/* Break LEDs                                        */
```

```
void LEDTest (void)
{
        unsigned int i;
        StatusLED(GREEN); BackLED(OFF,OFF); FrontLED(OFF);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        StatusLED(RED);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        StatusLED(OFF); FrontLED(ON);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        BackLED(ON,OFF); FrontLED(OFF);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        BackLED(OFF,ON);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        BackLED(ON,ON); StatusLED(YELLOW); FrontLED(ON);
        for (i = 0; i < 842; i++) Sleep(0xFF);
        BackLED(OFF,OFF); StatusLED(OFF); FrontLED(OFF);
}
/* END LED Test --------------------- */
```