

racinator FINAL REPORT

By

Team 11 RACINATOR

An assignment

submitted in partial fulfillment

of the requirements of

ELEC/COEN 390

Concordia University

April 4, 2010

RACINATOR AUTONOMOUS RACING ROBOT

# ABSTRACT

The following document is the response to your request for proposal (RFP), of an autonomous street racer. Based on the details of your RFP, our organization has prepared a design that can detect obstacle and follow black lines on a light background, from a start to finish as quickly as possible. Different implementation of hardware designs and control algorithms has been tested and improved to achieve costumer acceptance.

The design was achieved with seven infrared analog sensors placed strategically, and one digital sensor front mounted for object avoidance. This information is controlled by an eight-bit microcontroller for movement and direction controls. A pulse width modulated signals (PWM) controls the speed and forward/reverse for motion control. A line following algorithm is implemented to stay in the racetrack. The car works autonomously when placed on the floor at all time. The project history is included in this paper.

**Key Words:** Autonomous Robot Car, Infrared Analog/Digital Sensor

Table of Contents

[ABSTRACT 3](#_Toc259012118)

[Table of Figures 6](#_Toc259012119)

[Table of Tables 7](#_Toc259012120)

[1.0 INTRODUCTION 8](#_Toc259012121)

[2.0 PROBLEM STATEMENT 8](#_Toc259012122)

[2.1 Assumptions 9](#_Toc259012123)

[2.2 Requirements 10](#_Toc259012124)

[2.3 Constraints and Limitations 10](#_Toc259012125)

[2.4 Objective Tree 10](#_Toc259012126)

[3.0 BACKGROUND RESEARCH 12](#_Toc259012127)

[3.1 Survey of the Literature 12](#_Toc259012128)

[3.1.1 The Hopkins Beast 12](#_Toc259012129)

[3.1.2 The VAMP 13](#_Toc259012130)

[3.2 Technical Survey 13](#_Toc259012131)

[3.2.1 Hardware 14](#_Toc259012132)

[3.2.2 Software 17](#_Toc259012133)

[4.0 ALTERNATIVES 20](#_Toc259012134)

[4.1 Two Wheel Racing Robot 20](#_Toc259012135)

[4.2 8 Border Line Sensors RoBot 22](#_Toc259012136)

[4.3 Alternative Solutions and Evaluations 24](#_Toc259012137)

[5.0 PROPOSED METHODOLOGY 26](#_Toc259012138)

[5.1 Hardware 27](#_Toc259012139)

[5.1.1 Block Diagram 27](#_Toc259012140)

[5.1.2 Functional Description 29](#_Toc259012141)

[5.1.3 Turning 35](#_Toc259012142)

[5.1.4 Forward and Backward 35](#_Toc259012143)

[5.1.5 Avoid Object 36](#_Toc259012144)

[5.1.6 List of Components 37](#_Toc259012145)

[5.1.4 Description of Parts 38](#_Toc259012146)

[5.1.5 Assembly 40](#_Toc259012147)

[5.2 Software 45](#_Toc259012148)

[5.2.1 Description of the Algorithm 45](#_Toc259012149)

[5.2.2 Logic Flow Chart 46](#_Toc259012150)

[5.2.3 Function Description 47](#_Toc259012151)

[6.0 PROJECT MANAGEMENT 49](#_Toc259012152)

[6.1 Team Formation 49](#_Toc259012153)

[6.2 Working Agreement 50](#_Toc259012154)

[6.2.1 Group Organization 50](#_Toc259012155)

[6.2.2 Meeting Format 50](#_Toc259012156)

[6.2.3 Decision Making 51](#_Toc259012157)

[6.2.4 General Guidelines 51](#_Toc259012158)

[6.3 Work Breakdown Structure 52](#_Toc259012159)

[6.3.1 Initial Work Schedule 52](#_Toc259012160)

[6.3.2 Final Work Schedule 53](#_Toc259012161)

[6.4 Behavioural Checklist 54](#_Toc259012162)

[6.4.1 Phase1 – Defining the problem 54](#_Toc259012163)

[6.4.2 Phase2 – Formulating the solution 55](#_Toc259012164)

[6.4.3 Phase 3 – Developing models and prototypes 56](#_Toc259012165)

[6.4.4 Phase 4 – Presenting and implementing the design 58](#_Toc259012166)

[6.5 Feedback on Team Function 60](#_Toc259012167)

[7.0 TESTING AND RESULTIS 61](#_Toc259012168)

[7.1 Straight driving calibration 61](#_Toc259012169)

[7.2 Turning calibration 61](#_Toc259012170)

[7.3 Sensors reading 62](#_Toc259012171)

[7.4 Troubleshooting 63](#_Toc259012172)

[7.4.1 Hardware Trouble Shooting 63](#_Toc259012173)

[7.4.2 Software Trouble Shooting 65](#_Toc259012174)

[ANNEX 66](#_Toc259012175)

[A. REFERENCES 66](#_Toc259012176)

[B.DETAILS OF BUDGET 67](#_Toc259012177)

[C. APPENDIX 70](#_Toc259012178)

[Calculations 70](#_Toc259012179)

[CODE 72](#_Toc259012180)

# Table of Figures

Figure 1 Possible Path 9

Figure 2 Objective Tree 11

Figure 3 Hopkins Beast [2] 12

Figure 4 Lidar Diagram [5] 14

Figure 5 Dijkstra Algorithm [10] 18

Figure 6 A [12] 19

Figure 7 B [12] 19

Figure 8 A + B [12] 20

Figure 9 Two Wheel Robot Racer 21

Figure 10 Robot with 8 Border Line Tracker 23

Figure 11 Front View of Racinator 27

Figure 12 Racinator Operation Block Diagram 28

Figure 13 Communication Schematic 31

Figure 14 Control System Schematic 34

Figure 15 Full Electrical Schematic 35

Figure 16 Mechanical Schematic I 42

Figure 17 Mechanical Schematic II 43

Figure 18 Mechanical Schematic III 44

Figure 19 Mechanical Schematic IV 45

Figure 20 Logic Flow Chart 47

Figure 21 Initial Work Schedule 53

Figure 22 Final Work Schedule 54

Figure 23 Bill of Material Budget distribution 69

Figure 24 Labour vs. Parts Budget Distribution 70

Figure 25Calculations 72

Figure 26 CODE.H 73

Figure 27CASES.H 75

Figure 28 DRIVING.H 76

Figure 29 GLOBAL.H 77

Figure 30 INITI.H 77

Figure 31 INITPWM.H 78

Figure 32 LINE\_OBJ.H 82

Figure 33 TIMEDELAY.H 83

Figure 34 TURNING.H 85

# Table of Tables

Table 1Approximate Price for Two Wheel Racer Robot 21

Table 2 Approximate Price of the robot car with 8 side line sensors 23

Table 3 Con and Pros of two alternatives 24

Table 4 Motors Comparison 24

Table 5 Sensors Comparison 25

Table 6 Racinator specification 27

Table 7 Sensor stats at turning 35

Table 8 Forward and Backward operation 36

Table 9 List of Components 37

Table 10 HARDWARE Trouble Shooting 63

Table 11 SOFTWARE Trouble shooting 65

# 1.0 INTRODUCTION

The use of autonomous robots is globalized and becomes a strong tool to advance in technology nowadays. The main functions of autonomous robot such as navigation and localization are generally independent on the size of the robot therefore the small and low budgeted robot can be used and tested for the development and different area of society. [1] Therefore, the small scale racing robot is perfect to get experience in robotics. By doing so, COEN/ELEC390 team project will give practical skills in electrical and computer engineering: to become familiar with working together and to improve team work skills that required in the field of engineering. This document completely describes our racing robot, Racinator, in details.

# 2.0 PROBLEM STATEMENT

The problem of this project is to design a racing robot that can follow a black electrical tape which is going to be the line of the racing track, avoid any obstacles on the track, and arrive at the destination as fast as possible. The robot has to be able to finish a race within 200 seconds. The design racing robot has to autonomous driven by the programmed Atmega8 Microcontroller. Once the robot is activated, it should operate autonomously under its own power and complete the race without any further assistance. Furthermore, this robot has to be dismountable, sturdy and simple to use, and maintain. In addition, the cost of the design should not exceed two hundreds dollar of budget. In other words, our design has to be economical and efficient. The racing robot must be able to identify the different conditions of racing track in order to win the competition: such as sharp corners, different colors of lane, and obstacles on the track.

## 2.1 Assumptions

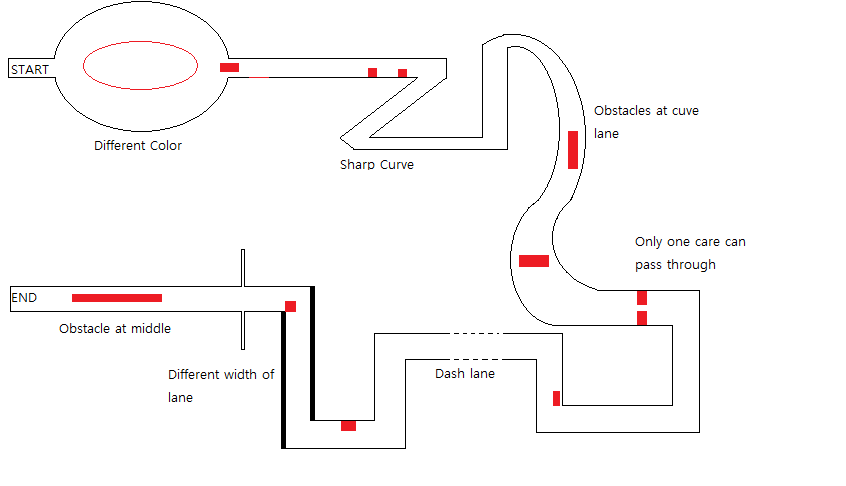
****

Figure Possible Path

The above figure helps us to imagine, assume and visualize various track situations and conditions. Many unexpected conditions could take place into the racing track. Our assumptions are the following:

* There will be obstacles on the middle of track
* There will be obstacles at the corner and on the curve lane
* There will be sharp and smooth curve turns
* There are different width of track lane
* The surface will be smooth

## 2.2 Requirements

* The robot should be efficient and economical.
* The robot must be built in March 29. 2010
* The robot should be dismountable, sturdy and simple to use and maintain
* The robot should be autonomously move; ability to detect obstacles and to follow lines
* The robot should be able to get the end of the racing track as fast as possible

## 2.3 Constraints and Limitations

* Completion time within 200 seconds
* Construction time, 1 term
* Limited budget 200$
* The size of robot
* Robot weight
* Limited access time to the lab
* The robot has to be autonomous.
* Limited hardware equipment and tools
* Components are easily mounted and dismounted
* Knowledge and experience with the Atmega8 Microcontroller
* The robot has to be efficient both in energy and cost

## 2.4 Objective Tree

In order to have a good design, we needed to analyze and identify its problem in details. This objective tree figure 2 was build before we came up with our design. It helped us to define the subcomponents of a design problem, specific tasks and activities that must be accomplished. In addition it was used to check whether our design met client’s needs. The initial main problem was broken down into five subcomponents; structure, user friendly, efficiency and mobility. And those five subcomponents are decomposed into more and more specific term.

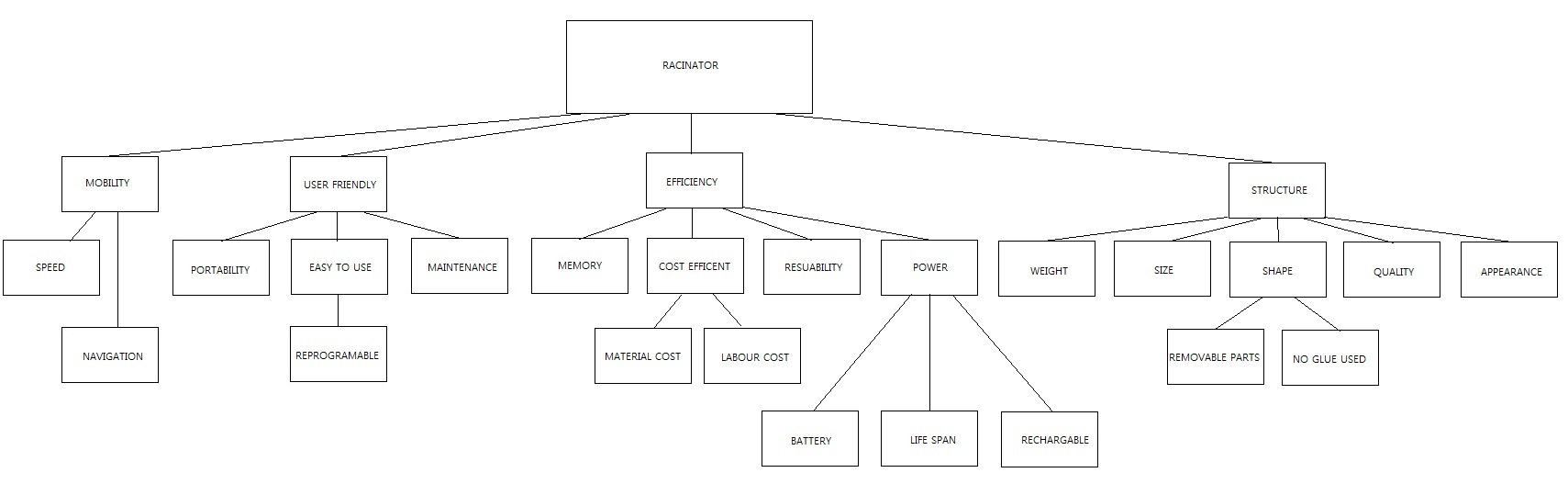


Figure 2 Objective Tree

# 3.0 BACKGROUND RESEARCH

## 3.1 Survey of the Literature

To come up with our design we had to do some research. Our understanding of the subject was nurtured by the following examples of robots we had examined. The Hopkins beast and the VAMP had inspired our design the most. Our survey of literature explains their basic functions.

### 3.1.1 The Hopkins Beast

## 1960beast.jpg

Figure 3 Hopkins Beast [2]

The beast was a simple robot built in the 1960s by Johns Hopkins. The robot goes around finding a white wall. Once the beast finds a white wall, it keeps wandering along it. The goal was as simple as to follow the white wall with the help of its sensors. It uses sonar to identify its surroundings. When the battery of the robot is going low it tries to find a black color surface, which represents the wall outlet; then, it will plug itself to recharge. The robot then starts over going along the white wall repeating itself [2].

### 3.1.2 The VAMP

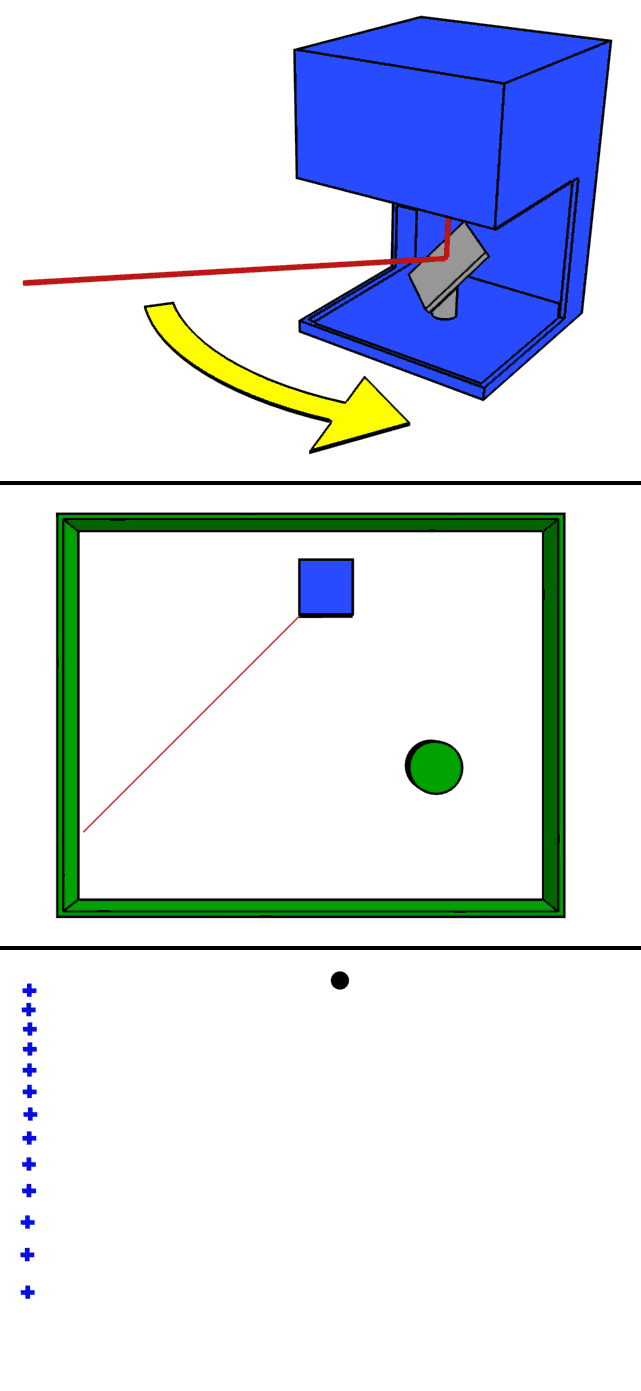
The vamp was one of the first autonomous cars. It was built by Mercedes-Benz in the 1990’s. The car guided by a computer drives kilometers into traffic with a safety driver in it. It takes decisions to change lanes which are to be confirmed by the safety driver. It does not have a human driver per se; the safety driver just makes sure that the output given was not going to cause casualties, because the car was operating in real traffic. The car drove from Munich to Copenhagen at speed up to 180 km/h. It takes sensors input of the environment and compute the path it has to take. It has to take complicated cases into consideration. The algorithm has to be complex; it did not use any GPS system. It guidance was visual only. To account for complicated visuals the car only considers objects that were essentials to its motion decision. That was a judicious decision because the capabilities of computers were limited at the time. To handle the vision four cameras with different focal points were used to allow for depth perspective. The experiment was one of the major accomplishments of motion robots [3].

## 3.2 Technical Survey

The tools we used to implement our design follows the techniques we are describing in this section. The software and the hardware parts are separated for clearer understanding of the structure of our design.

### 3.2.1 Hardware

The hardware part is the physical aspect of the project. To have an idea of how to use the different devices we used, we had to understand how they function. The LIDAR is a device that makes us grasp the idea of sensors. This device explains the basic concept of sensors. To come up with a circuit design we try to follow the beam robotics way. We explain below these two ideas that we use as guidance when we want to come up with a good hardware design.

LIDAR

The lidar (**Li**ght **D**etection **A**nd **R**anging) scans its surrounding with a laser to represent surrounding. It is similar to a radar. It uses light pulses to calculate the distances of objects. The delay between the sent signal and the reflected one is used to compute the distance. This method is often used today for mapping an environment with sensors [4].

Figure Lidar Diagram [5]

Beam Robotics

The beam robotics invented by Mark Tilden is a robot built without a microprocessor. It is composed of simple analog electronic components like amplifiers. The robot operates by comparative logics; to move, the robot compares the voltage stimuli it receives (from sensors) and move according to the pre-disposed logic circuits placed in it [6].

Components: physical parts

Our hardware design was not entirely inspired from the beam robotics, but the concept helps us understand how the physical part should be adequate with different type of the software algorithm. The Street Racer that we design can be implemented in different manners. For a low cost and short time table design, we have chosen right of the shelf items, as proposed in the following:

1. Logic Controller:

* Programmable Logic Controller (PLC)
* Programmable Flash Controller (PFC)

From the controllers proposed, the PLC would be the easiest to implement with ladder diagrams. The designed software available can limit you for this one. Those devices involve high cost. In a lower cost alternative, PFC offers flexibility into programming and wide range of inputs outputs.

1. Front Sensor:

* Range Infrared Distance Sensor & Light Sensor
* Ultrasonic Range Finder (URF)

To detect ahead vehicles and to be able to shift lane, a wide range of sensors are available in a low cost manner. First a range sensor combined with a light sensor, both analog devices very easy to implement and control. Brute force programming will have to be implemented due to nature of the signal, for such a complex decision. In second, ultrasonic sensors, which use sonar technology to map the environment, provides the necessary information for the PID to make the appropriate decision. The programming involve is more complex, more time will have to allocated.

1. Line sensor:

* Infrared Reflective/ Light Sensor (IRS, ILS)
* Monochrome VGA Sensor

(There exist line sensors by them self, combined with rang and light easier to implement [7], [8])

In order to comply with the requirements to stay in between two lines, four analog sensors will be mounted to each wheel, to provide the necessary information for the line tracking PID. Infrared reflective analog sensor generates a 0 to 5 VDC signal representing the reflection intensity, with digital output to detect black tape, or with analog output to detect shades of grey. Easy to implement, due to factors, as light condition will vary depending of the location, this will have to be taken into consideration or always control the amount of lumen detected by the sensors. IRS It offers a wide range of possibility. Monochrome VGA would provide more than necessary information to stay in center line. The amount of unit would be less, still the cost is high. For simplicity and cost factor, the light sensor would be appropriate for this operation.

1. Motors:

* Motor with Servo
* Brushless DC Motor
* Geared DC Motor

The Servo is easy to use with the PWM. We did not have any servo motor in the part list available to us, so we did not have to take it in consideration. The Brushless DC motor has a very good power ratio. In another words, it is very efficient. Also, the maintenance is low. The Geared dc motor has a high maintenance cost due to brushing. It does not last long. For our project we need to test a lot so we ended up using the brushless DC motor.

1. Chopper circuit:

* Relay circuit
* BJT circuit
* MOSFET circuit

For motion and steering, a 2 quadrant chopper (Quadrant I & IV) will be used in order to operate with positive voltage/ positive current and positive voltage / negative current. Using relay circuit would introduce more possible point of failure, with more analog devices. It would be cost effective; as well as H-Bridge relay circuit will have to designed and tested. BJT circuit would be obsolete to use and expensive. A wide range of microchip H-Bridge is available at lost cost.

### 3.2.2 Software

The algorithms we are working on have the fingerprints of wonderful ideas that were introduced in the past. Some of them we learnt in previous courses we took, like the data structure graph. Others are new concepts to us like the Minkowsky sum. Surprisingly to us, we knew the tools already but did not know that they could be implemented in our project. For example, we knew the data structure graph but did not know it could be used innovatively for the visibility graphs concept, which we will explain soon. We mentioned here a few of the ideas we encounter; the one we think are essential to our programming methodology.

Visibility Graphs

To analyze the environment of the robot a coordinate system is used. After scanning the environment, each object is assigned a coordinate value (x and y); it can also be a three-dimensional assignment (x, y, and z). To find the path of the robot from point A to B we use the Dijkstra algorithm. The latter algorithm finds the shortest path between two vertices in a data structure graph. Empty spaces are assigned the vertices that are represented in the graph. Knowing the final destination (represented as the final vertex) we can compute the can compute the fastest path the robot should take. This is the way “Shakey the Robot"was built to behave (by SRI International in 1972) [9].

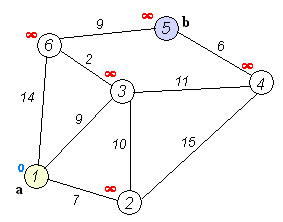


Figure 5 Dijkstra Algorithm [10]

Minkowski Sum

The Minkowsky sum is the addition of two sets A and B. If A and B represent triangles in figure 2 and 3 respectively the result will circle their shapes as shown in figure 4. This renowned algorithm is used in object detection, assuming that A and B are the objects to be avoided. The addition of the two sets *A* = { (1, 0), (0, 1), (0, −1)} and *B* = { (0, 0), (1, 1), (1, −1)} will give the Minkowski sum *A* + *B* = { (1, 0), (2, 1), (2, −1), (0, 1), (1, 2), (1, 0), (0, −1), (1, 0), (1, −2)}, which looks like a polygon. So the path that a robot will take will be the vertices of this polygon [11].

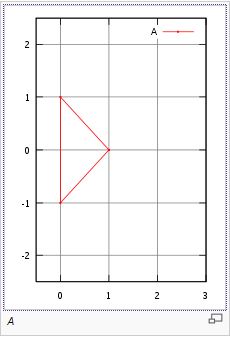


Figure A [12]

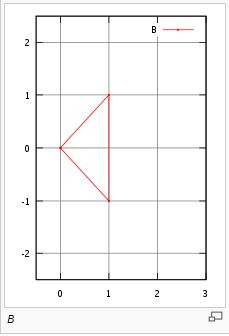


Figure B [12]

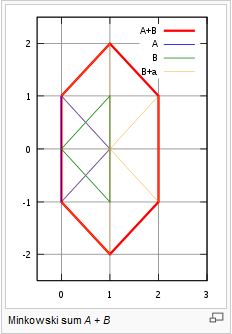


Figure A + B [12]

# 4.0 ALTERNATIVES

In order to solve and get better solution to the problem statement mentioned in the first page, each one of our team members brings up with different alternatives to this specific problem: build an autonomous racing robot. Every one of us has different ideas to share with. At the end we will compare and analyze each alternative to design a good racing robot.

## 4.1 Two Wheel Racing Robot

Our imaginary designed racing robot will look like the following figure:

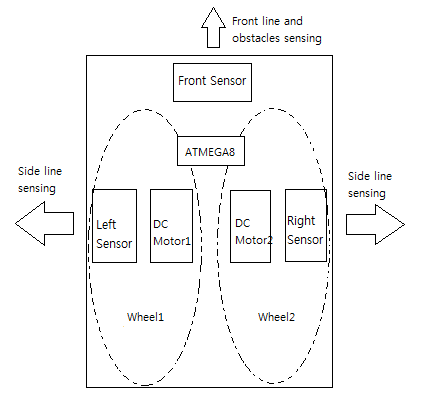


Figure Two Wheel Robot Racer

This simple designed racing robot is powered by 9V lithium battery. It has three sensors and two wheels. One sensor will be placed in front of the robot to identify any obstacles and two remaining sensors will be placed on the left and on the right sides of the robot to detect the line of the track. The wheels are big and strong enough to receive the weight of the robot and to make a left and a right turn. In addition, two DC motor will be used and will be directly connected to each wheel to provide power and speed to the robot. The main components for this robot are the following:

* 3 Sensors
* 9 V Lithium Battery
* 2 Wheels
* 2 Dc Motors
* Microcontroller Atmega8
* Voltage Regulator

Table Approximate Price for Two Wheel Racer Robot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Quantity | Cost($) | Total Cost($) | Balance($) |
| Sensor | 3 | 15 | 45 | 155 |
| Battery | 1 | 4 | 4 | 151 |
| Wheel | 2 | 10 | 20 | 131 |
| Motor | 2 | 13 | 26 | 105 |
| Atmega8 | 1 | 3 | 3 | 102 |
| Chassis | 1 | 5 | 5 | 97 |
| Voltage Regulator | 1 | 0.5 | 0.5 | 96.5 |
| **Total** |  |  | **103.5** |  |

This designed robot will be able to detect the object in front and make a perfect left and right turn. It will need approximately 103.5$ to build hence the price stays in the bounds of the budget.

## 4.2 8 Border Line Sensors RoBot

A robot race car that we shall implement in this project will need to satisfy the racing descriptions and constraints. For the robot to stay in between two lines and avoid objects, the race car will need a set of sensors to detect such things. In our view the robot will have an advantage in knowing where it is on the track by sticking itself to a border line of the track. In this respect, having the car follow the border line to reach the end of the track, we will need sensors to pick up the black border line as well as sensors to keep away from the black boarder line. With these sensors implemented, the car may follow a boarder of the track without falling away from the boarder or going too much outside of the track. The car will need 8 sensors on the sides of the car, 4 on one side while 4 on the other to maintain the position of the car following the line.

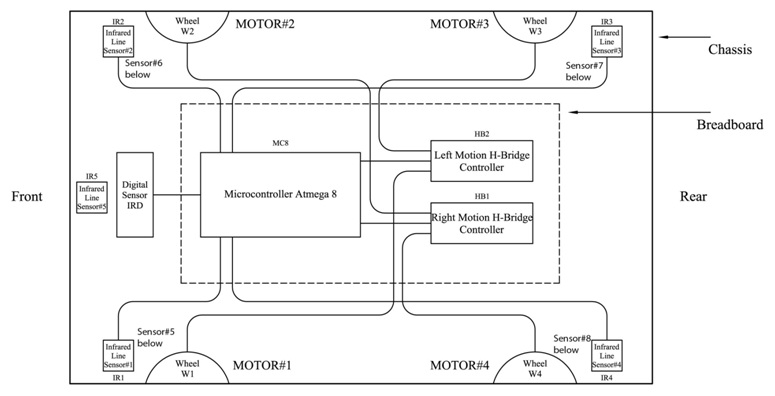


Figure 10 Robot with 8 Border Line Tracker

Figure 10 shows the sensor layout as well as the other components. Taking one of the sides of the car in focus, we have 2 sensors at the forward side and the 2 others on the back side of the car.

* Sensor of group A (Sensors 1, 2, 3 and 4) will stay in focus of the black line of the track while the other side sensors (Sensor group B - Sensor 5, 6, 7 and 8) will stay clear of the black line.

Hence when sensor A detects a border line it will stick to the black line and sensors B will make sure it keeps away from the black line. With this solution, the car will follow the black line whichever side the black line is on in respect to the car. In addition to the 8 sensors, there are 2 additional sensors in the front of the car; one is again for line detection while the other is for object detection. Figure 10 demonstrates the use of the front line detection sensor and the front object detection sensor respectively. The cost of implementing this solution is out of the reach of the project budget of $200 as shown in table 2. The observations of this alternative will then be put in use for other alternatives.

Table Approximate Price of the robot car with 8 side line sensors

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Quantity | Cost($) | Total Cost($) |
| Sensor | 10 | ~15 | 150 |
| Battery | 1 | 4 | 4 |
| Wheel | 4 | 10 | 40 |
| Motor | 4 | 13 | 52 |
| Atmega8 | 1 | 3 | 3 |
| Chassis | 1 | 5 | 5 |
| Voltage Regulator | 1 | 0.5 | 0.5 |
| **Total** |  |  | **254.5** |

## 4.3 Alternative Solutions and Evaluations

Table Con and Pros of two alternatives

|  |  |  |
| --- | --- | --- |
|  | Advantage | Disadvantage |
| Alternative 1 | * Cheaper * Easier to | * Unstable * Week * Unable to detect the black tape in front |
| Alternative 2 | * High sensitivity * Powerful and faster | * Very challengeable in programming * Hard to implement the software * Expensive |

Table Motors Comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sensors | Gear Ratio | Torque | Speed | Current | Size | Cost |
| Solarbotics GM2 | 224:1 | 50 in\*oz | 38rpm | 52mA | 55mm (2.17") x 48mm (1.89") x 22.7mm (0.894") | 7 $ |
| Solarbotics GM6 | 120:1 | 25 in\*oz | 145rpm | 88mA | 36.5mm x 40.7mm x 24.7mm (1.43" x 1.60" x 0.97") | 7 $ |
| Solarbotics GM8 | 143:1 | 43 in\*oz | 70rpm | 57.6mA | 53.8mm (2.12") x 47.8mm (1.88") x 22.9mm (0.902") | 7 $ |
| Solarbotics GM10 | 81:1 | 3.6 in\*oz | 370rpm | 24.6mA |  | 12 $ |
| Servo motor HS-311 |  | 49 in\*oz |  | 160mA |  | 10 $ |

GM6 has been chosen for our design due to the physical limitation. This motor has advantage of highest speed. Our team has to pick the smallest and fastest motor that can fit into the under chassis because the robot has four motors

Table Sensors Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensors | Output | Minimum Distance | Output Distance | Cost |
| GP2D12 | Analog | 10cm | 80cm | 12 |
| GP2D15 | Digital | - | Fixed at 24cm +/- 3cm | 15 |
| OPB704 | Analog | 3.8mm | - | 5 |

GP2D15 IR range sensor has been chosen for the object detection.

OPB704 has been chosen for the side sensors. It has the following features: [13]

* High sensitivity
* Low cost
* Dust protection

# 5.0 PROPOSED METHODOLOGY

Racinator is an autonomous racing robot that can navigate and race on a track that is bordered by two lines of black electrical tape. It is designed to successfully complete a racing. The Racinator has abilities to recognize the environment of the track; avoiding obstacles on its way and detecting which way to turn and follow. The robot is driven by Atmega8 Microcontroller that is loaded with a high level of C programming software. The robot uses a Quad Dual H-Bridge to control four GM 6 motors. And also, the Racinator uses 6xAA battery to supply power to the Atmega8 Microcontroller and the Quad Dual H-Bridge. The robot is easy to use, maintain and modify.

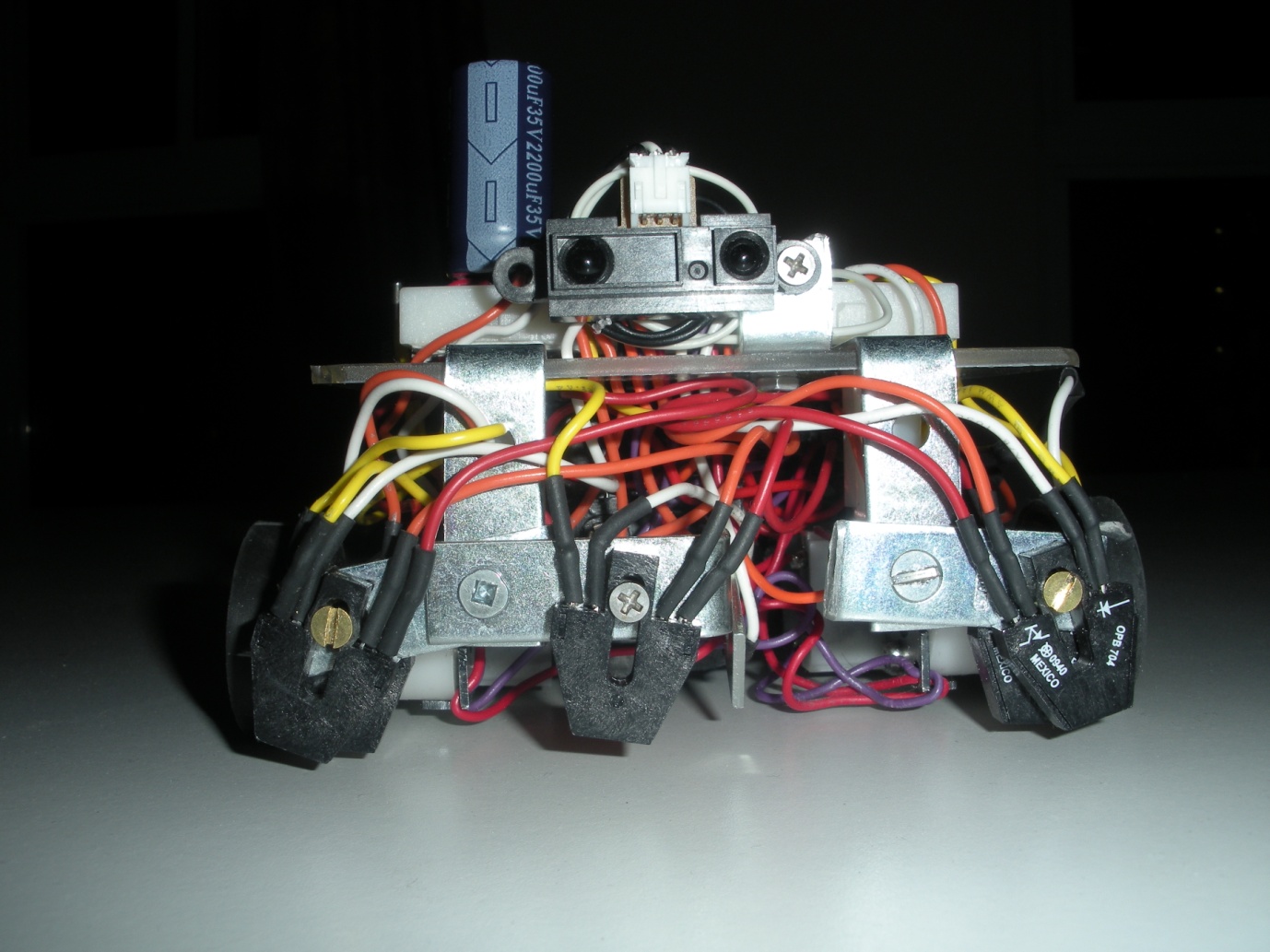


Figure Front View of Racinator

Table 6 Racinator specification

|  |  |
| --- | --- |
| Length | 13.1cm |
| Width | 11.6cm |
| Total Height | 7cm |
| Weight | 900g |

## 5.1 Hardware

### 5.1.1 Block Diagram

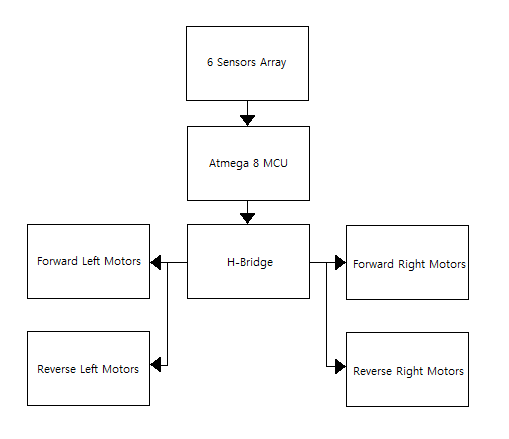


Figure Racinator Operation Block Diagram

The Racinator uses an Atmega8 Microcontroller to control both hardware and software functions of the robot. The robot operates autonomously within five seconds and decides which line to follow. The robot stays between the two lines of black electrical tape and avoids obstacles on the track. The black electrical tape and obstacles on the track are detected by five analogue infrared sensors OPB704 and one digital infrared sensor GP2D15.

* Two analogue infrared OPB704 sensors are mounted on each side of Racinator and extra one is mounted in front to detect black electrical tape.
* One digital infrared sensor GP2D15 is also mounted in front of Racinator to detect obstacles so that Racinator can go around the obstacle.

If obstacles are detected by six sensors, the signals are sent to the Atmega8 and H-bridge to activate motors. Racinator will move corresponding to the instructions given.

### 5.1.2 Functional Description

Communication

The Atmega8 microcontroller software will be loaded from the computer through the ISP, connected to pin 1 (RESET, this pin is connected to a pull up resistor (R1=10KΩ) to limit the current to 5mA, to prevent any current overflow to the system which can cause damages, as well a push button PB-1 will enable an interrupt to the microcontroller).

At start the software will initialize the USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) through pin 2 (RXD) & pin 3 (TXD), and communicate with the MAX 232 microchip to meet the RS232 standard voltages (Logic 0 = +3V to 15V, Logic 1= -3V to -15V), the signal will flow through a DB9 connector at a baud rate of 9600. The MAX232 is powered by the 5VDC capacitive (C1= .1µF) regulated voltage, and assisted by charged pump capacitors (C2=C3=C4=C5=.1µF) circuitry to achieve the appropriate DC/DC conversion. Data will be stored on the computer through a HyperTerminal consol, set to port COM2 (for testing purposes).



Figure Communication Schematic

Control System

At start the software will initialize a sequence to wait and go detect a black line to follow. Those actions will be determine by 5 analog sensors mounted strategically, next to each motor M1 to M4 and one mounted in front. Each sensor is powered to Vcc (5VDC) and a resistor (R5 to R9=125Ω) set for idle operation. IR1 to 5 is (OPB704) connected to the microcontroller Port C: 0 to 5 & Port B: 2 to 5 (set as analog inputs). The output signal from the analog infrared sensor will range from 0 to 1.63VDC & 0 to 2.54 cm (corresponding to the output voltage) for the reflective curve (cf. datasheet OPB704, figure: output distance, page: 9) optimal for sensing black lines. The purposes are to detect and follow the black line within 1.52 cm of this one (Corresponding to 0.4VDC). For better control, Vref will be set to an external reference voltage in the ADCMUX register, at pin 20 (AREF). The reference voltage is determined following this equation: ADC = (Vin\*1024/Vref), (Vin=0.45VDC, ADC=1024). To achieve a reference voltage of 0.4VDC, a voltage divider is implemented through an 8kΩ resistor R1 and a variable resistor R2 of 1kΩ, to achieve the desired reference voltage.

To detect, analyze and avoid objects ahead, a front mounted digital infrared sensor (GP2D15), with a focal detection range set at 24cm, is connected to pin 9 (PB6, set as an digital input) with a pull up resistor of 12kΩ (R3), for current limitation.

The H-Bridge (1 & 2) will allow the DC motors to operate into 2 regions of operations. First with a forward motion with positive voltage and positive current, a green LED (LTL-1CHG) will be lit corresponding to the motor operation, second a backward motion with positive voltage and negative current, a red LED (LTL-4222N) will be lit corresponding to the motor reverse operation. To achieve forward, backward, left and right motion, the microcontroller will send digitals signals through port D: 0 to 7 (set as digital outputs).

Two types of signal are implemented in the software. First of all, a pulse width modulated signal (PWM-M1 to M4) will enable the motor drivers at the H-Bridge (1 & 2) for speed control. As well, a forward/reverse control signal (F/R-M1 to F/R-M4, 0 to 3 VDC) will enable for the appropriate motion (pin 2 or 7 for motor M1/M2, pin 15 or 10 for motor M3/M4). A transistor (2N3904) logic inverter is implemented for reverse operation with one control signal, as we are limited with the number of outputs from the microcontroller. When the enable signal is low and the current is low, the transistor is cut off and the output terminal (collector, pin 2 or 15 of the H-Bridge) is pulled to 5VDC through a resistor (R=10kΩ), this correspond to a forward motion. When the enable signal is high, the transistor inverter circuit will act as a short circuit and set pin 2 or 15 to low and set pin 7 or 10 to high, corresponding to backward motion.

This circuit gives the option to the microcontroller to control each motor individually with speed and orientation control. The software will determine the appropriate sequence of the H-Bridge operation for the desired motion.

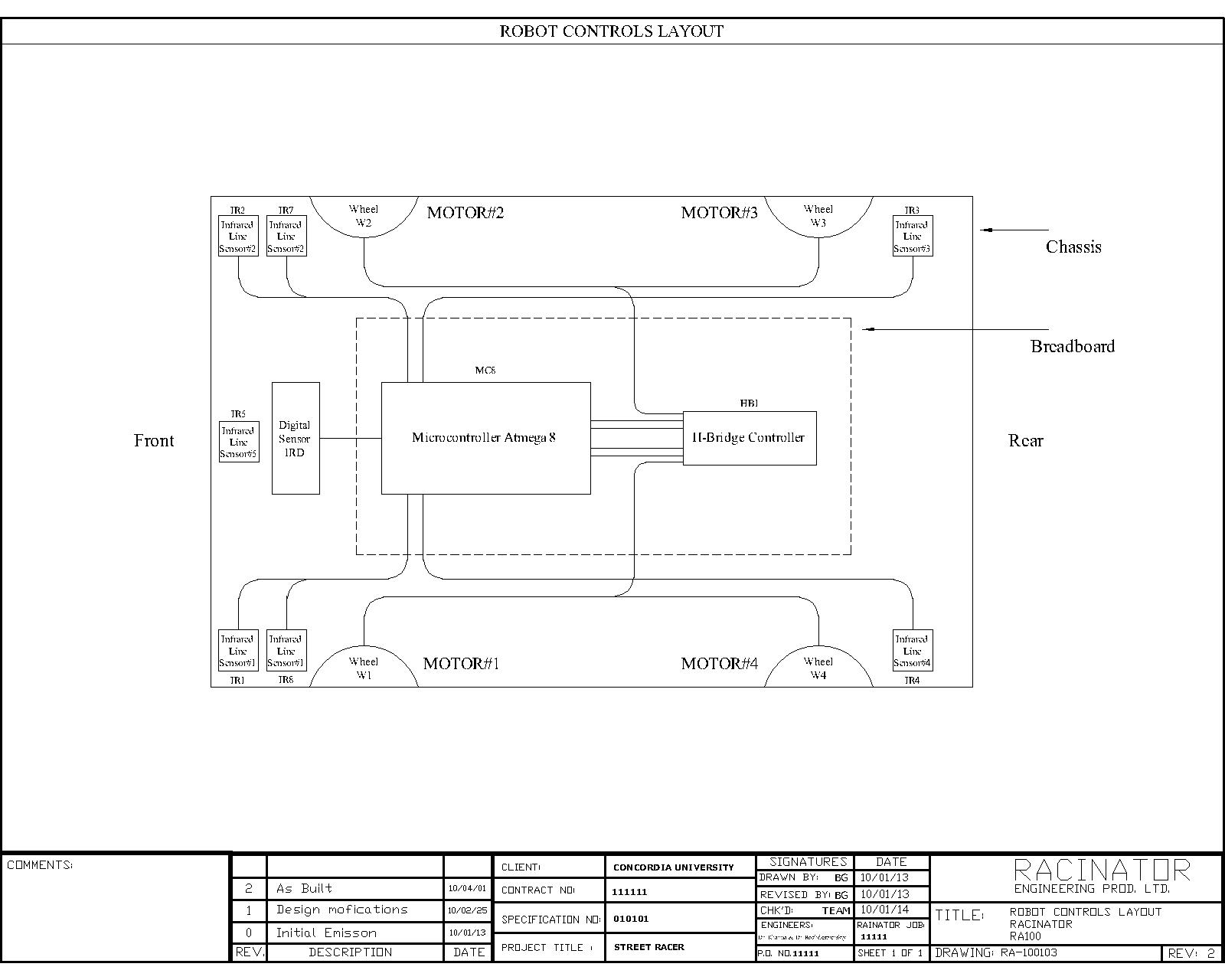


Figure Control System Schematic

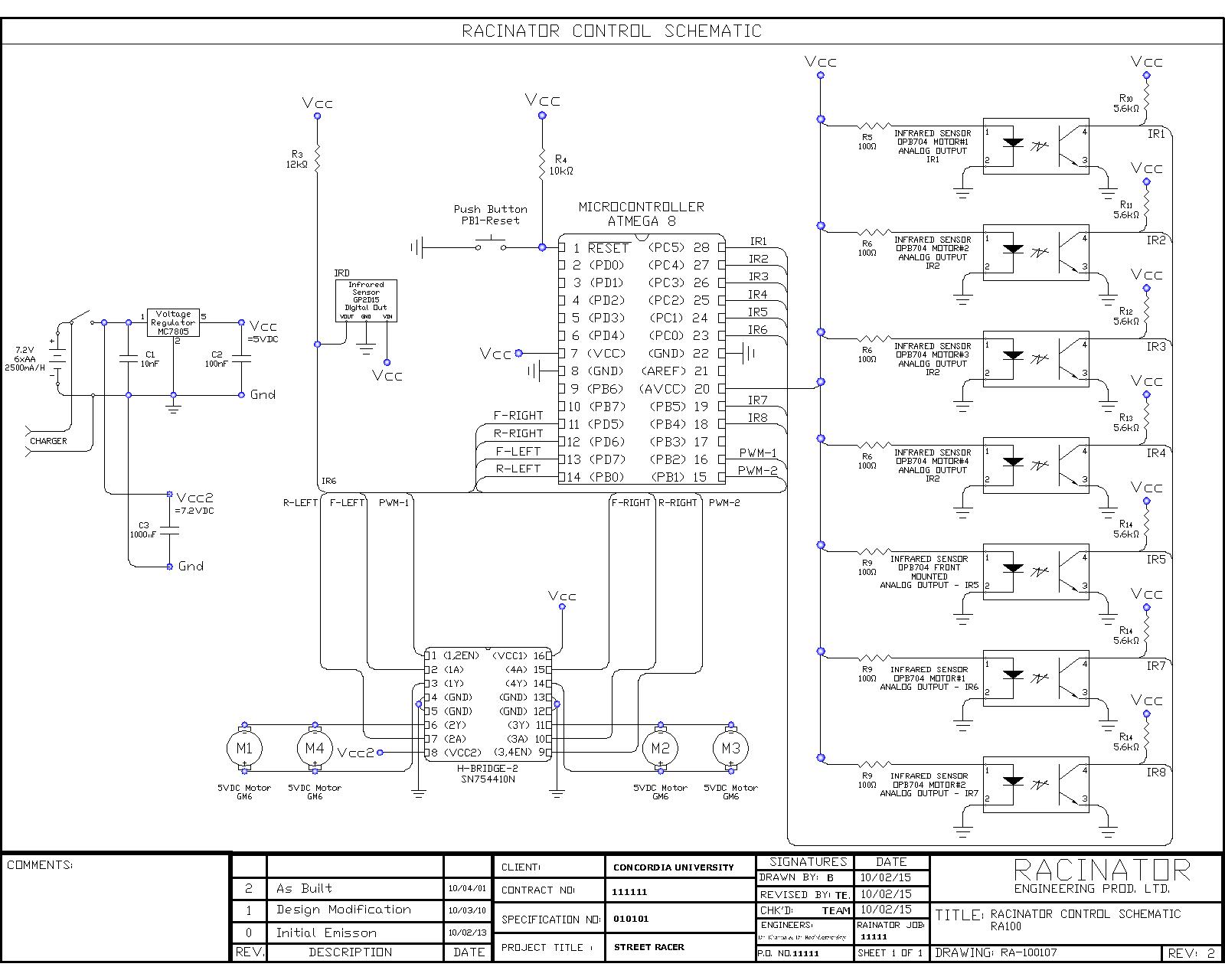
****

Figure Full Electrical Schematic

### 5.1.3 Turning

Two IR OPB704 sensors are mounted on the side of the robot. Without the side sensors, the robot will not detect the black line and turns at the corner. Therefore it will go out of bounds. Basically, these two sensors will detect the black electrical tape and decided to turn left or right. If the front side sensor loses the black line and so the car turns right until the front side sensors detect the black line again. If we have only one side sensor the car may not be parallel to the black tap and hence go out of bounds, for this reason there is a front side sensor and a back side sensor.

Table Sensor stats at turning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Front Left Sensor | Front Right Sensor | Back Left Sensor | Back Right Sensor |
| Turn Left | Off | On | Off | On |
| Turn Right | On | Off | On | Off |

### 5.1.4 Forward and Backward

In order to go forward, one side of two sensors both has to detect the black electrical line. It will keep going forward until the front sensor loses the black electrical tape. At any time when the sensor receives unexpected voltage the robot immediately stops and will operate turning or backward according to the situation. In case of the black tape places in front of the robot, the front side sensor will detect the black line and so the car stops otherwise it goes straight. Without the front sensors, the car will not notice a turn in the street where the black line is in the front of the car and therefore go out of bounds.

Table Forward and Backward operation

|  |  |
| --- | --- |
|  | Sensors |
| Forward | Front right sensors are on line and  Back right sensor is on the floor  \*note: considering only right side for forward |
| Backward | Both front sensors touch the line |

### 5.1.5 Avoid Object

In a situation the object is placed in front of the robot. The front IR GP2D15 sensor will detect it and go around the objects. Once GP2D15 sensor detects obstacles in its path as shown in this figure, the robot will shift over to the other side of the street. This GP2D15 sensor has an object detection range of 24cm from the car which is long enough to detect the object in front. Basically once the front sensor picks up an object, the car will shift to the opposite lane and follow other line.

### 5.1.6 List of Components

Table 9 List of Components

|  |  |
| --- | --- |
| Items | Quantity |
| Chassis (112x106mm) | 1 |
| Atmega8 Microcontroller | 1 |
| H-Bridge Motor Driver SN754410N | 1 |
| Battery Pack 6xAA, NiMH + Battery Charger | 1 |
| Analog Infrared Sensor OPB704 | 7 |
| Digital Infrared Sensor GP2D15 | 1 |
| GM6 Motor from Solarbotics | 4 |
| Resistors | 16 |
| Capacitors | 3 |
| ISP Header | 1 |
| LPT/ISP Cable | 1 |
| 5V Voltage Regulator | 1 |
| RW2 31mm Wheels | 4 |
| Wheel Bracket | 4 |
| 270 tie point Breadboard | 2 |
| Push Button | 1 |
| Total | 49 |

### 5.1.4 Description of Parts

Atmega8**:**

We use the Atmega8 microcontroller for the robot: It is used for the main architecture of Racinator. This programmed microcontroller controls all the components of the robot; H-bridge, LEDs, Motors and Sensors.

Voltage Regulator**:**

The voltage regulator 7805 is used to adjust output voltages and this is necessary components of Racinator: it will step down the 6xAA batteries to obtain +5v input voltage level for the ATmega8, motors, sensors, and LEDs.

H-Bridge IC:

The SN754410, H-Bridge, is an electronic circuit that is often used in robotics. This chip will drive the motors: This chip gives the ability to make the Racinator to move either in forward direction or in backward direction. The direction is controlled by applying the voltage in different direction across the motor. [7] In addition, the speed of the motor is controlled by the pulse-width modulation (PWM) signals generated by the Atmega8 microcontroller.

Mobility**:**

For the performance on mobility, we decided to go with the Solarbotics Gear Motor 6 (Baby GM2) to move the robot. This motor can rotate at up to 145 RPM with a supply of 5V. GM6 provides a large amount of torque 25 oz-in which is enough to run the Racinator. The ideal power consumption at 5.0V is 88mA. GM6 motor is adequate to Racinator. We decided to go with the smaller wheels that is feed to GM6; RM2 wheels. There will be 4 motors and 4 wheels used for the robot for its stabilization.

Sensors**:**

Two analogue infrared OPB704 sensors are mounted on each side of Racinator and extra one is mounted in front to detect black electrical tape. One digital infrared sensor GP2D15 is also mounted in front of Racinator to detect obstacles so that Racinator can go around the obstacle.

Battery Pack:

Rechargeable 6xAA battery pack is used to power the ATmega8, sensors, motors, and LEDs. The supply of this robot needs to give a steady 5V output to the circuit and motor. The 5 volts will be regulated by a 7805 voltage regulator. The pack has a rating of 2500 mA/h which is more than enough to run the robot for the 200 second time limit.

### 5.1.5 Assembly

1. Refer to the Table 6 components list for appropriate part number.
2. Verify motor dimension and wheel mounting.
3. Take 2 motors and align them to the one side of the robot.
4. Motor bracket can be purchased or build with 1/32” aluminum (form any hardware store), or any strong material. Cut to dimensions, drill and tap where necessary to attach the motors.
5. Spacers of 10 mm (x4) will be needed to clear the wheel from the chassis. Any strong material will suit.
6. The chassis can be purchased or build of any strong material, as Plexiglas, cut into 112x106mm
7. Prepare for final assembly, attach the spacers to the bracket and attach it to the motors.
8. Repeat those steps for the second half of the robot.
9. When Left and right half are completed, assemble as given in figure 14 to 16.
10. Place sensors at given in figures.
11. Now you have Racinator.

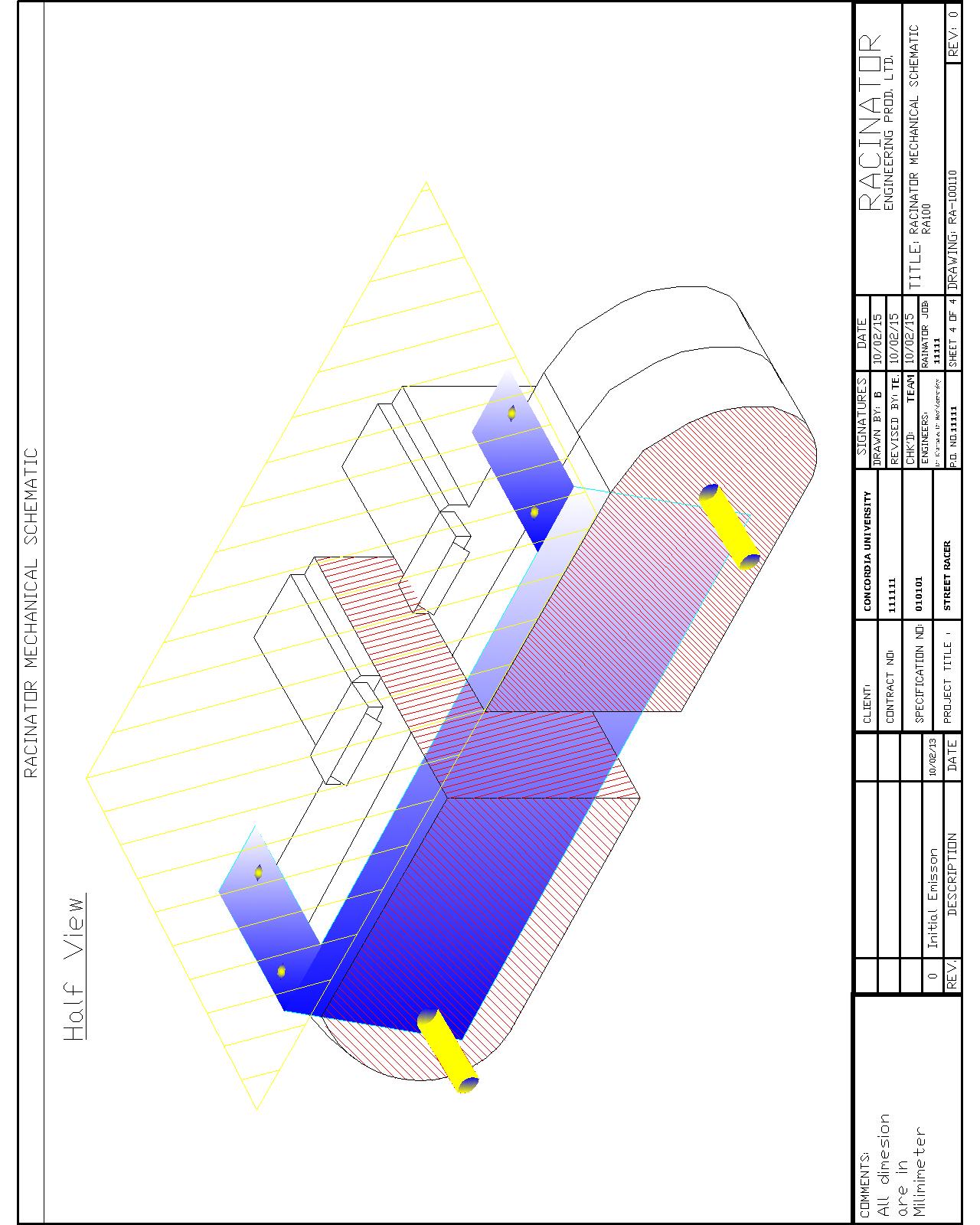
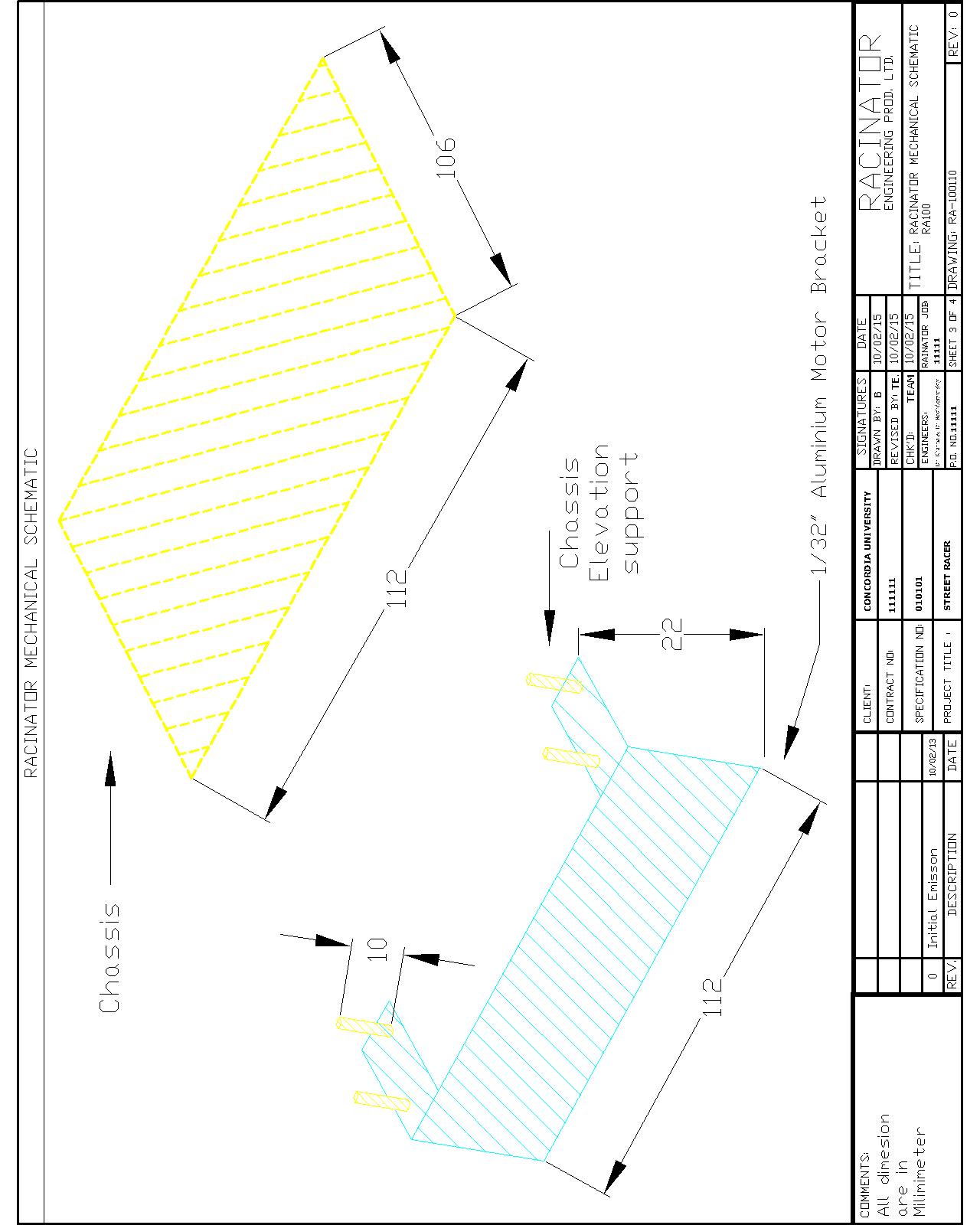
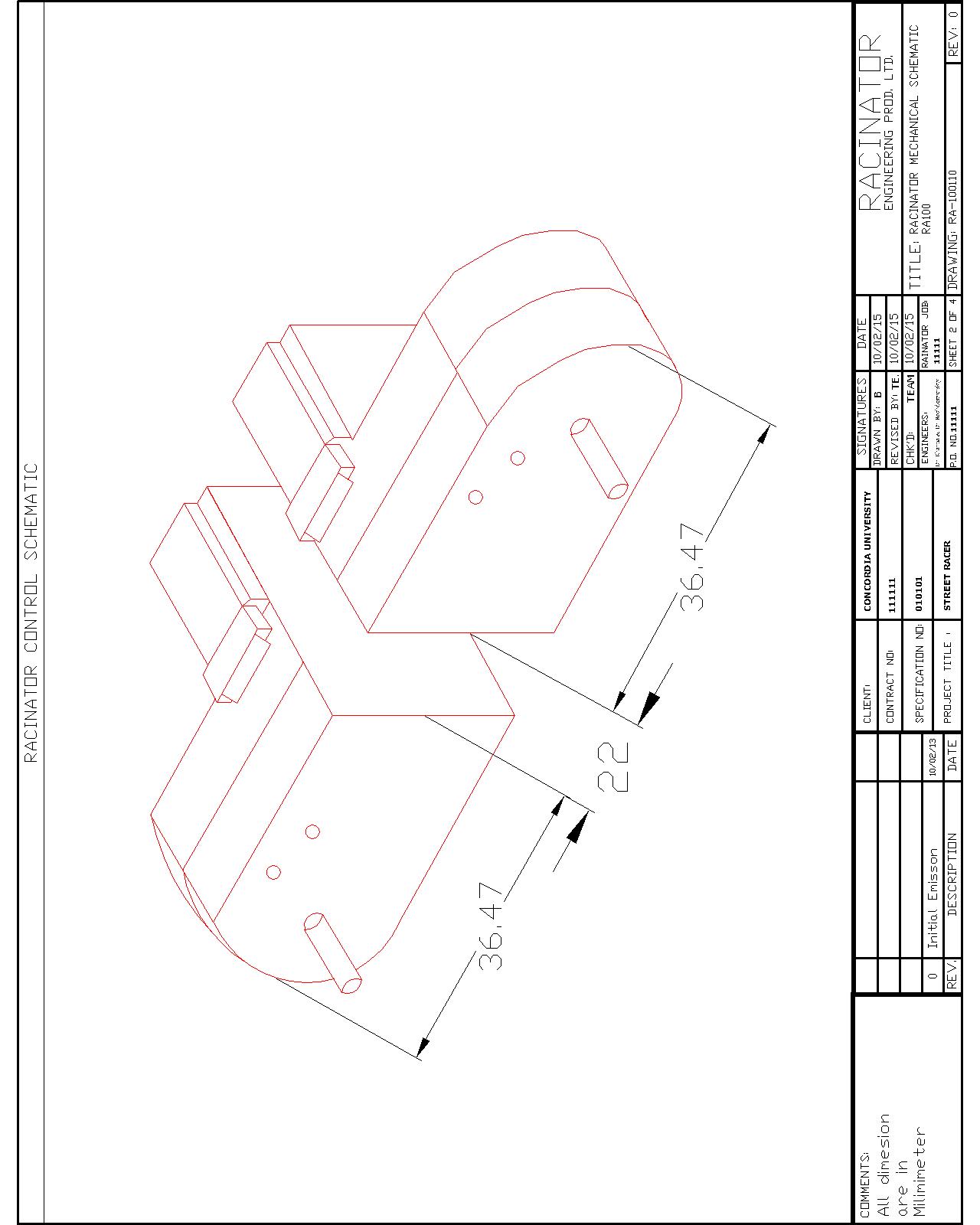


Figure Mechanical Schematic I

Figure Mechanical Schematic II

Figure Mechanical Schematic III

Figure Mechanical Schematic IV

## 5.2 Software

### 5.2.1 Description of the Algorithm

* The program starts with a 5 seconds delay.
* The side sensors try to read the line to check if the robot is on the left or right side of the track.
* If the side sensors detect any line then the robot will follow that line.
* Else it will turn right 90° to see if the front object detection sensor signals an object.
* If there is an object we turn 100° to the left and drive forward to get closer to the left line.
* Else the robot has to turn 75° to get closer to the right line.
* Once a side line is read, it will follow that line.
* While following the line:
* If a front side sensor stops detecting the line that it is following then it will turn towards that line till it catches it again, this case means that a turn occurred.
* If the front line sensor detects a line then it will turn left if it was following the right line or vice versa.
* If the front object sensor detects an object then it will turn 45° from the line it was following and drive till it detects the line on the other side of the street and start following it.
* Refer to Figure 18 for the Logic Flow Chart.

### 5.2.2 Logic Flow Chart



Figure Logic Flow Chart

### 5.2.3 Function Description

Our program contains eight files:

Code.c, Driving.c, Initializations.c, InitializationPWM.c, Line\_Obj.c, Turning.c, TimeDelay.c, Cases.c, Global.h

* Code.c is where the main commands are executed; the usual C main function is placed there. Thus, we call all the functions from here most of the time. First we initialize the robot, by setting the right ports. We call functions from Initializations.c for that. Also, the motors and the H-bridge ports are set. The PWM set-up function is called from InitializationPWM.c. After the initializations, a line search is done. A function is called to see if the robot is on a line, i.e. we check if our side sensors are aligned to a line. When we find a line we record into a variable if the right or left side of the robot is sensing that line. That way we know if we are on the right or left side of the track. After that, our program basically retrieved the conditions and set the appropriate commands by calling the functions from our included files.
* Initializations.c enables the ports for the H-bridge to communicate with our microcontroller.
* InitializationPWM.c contains the setting of the timer registers for the PWM signals to be active. There is a function there when called with the right parameter will change the duty cycles to the motors.
* Global.h has the global variables that should be accessed by our entire program.
* Timedelay.c functions allow us to delay commands we think appropriate for a certain cases.
* Turning.c make us turn with various angle, the latter being the passing parameters to the left and right functions.
* Driving.c has the functions that make the robot go forward and backward by enabling the appropriate ports of the H-bridge. We modify the speed by varying the PWM.
* Line\_Obj.c monitors the robot moving along the track lines. That’s where we instruct the robot to get on the track after observing the data from the sensors. We try, there, to accurately give the right commands so that the robot always follows the left or right line.
* Cases.c is where the logic of the program stands. All the possible cases or situations are examined there. By fetching the sensors’ inputs we construct some kind of logic table. This will help to make decisions in the Line\_Obj.

# 6.0 PROJECT MANAGEMENT

## 6.1 Team Formation

1. Abdullah Kadhim

Abdullah Kadhim is the team leader of group 11 Racinator. He is currently in the program of computer engineering software option. He provides motivation as well as inspiration for the robot design.

1. Ouben Jeong

Ouben Jeong is a 3rd year electrical engineering student. He has an intermediate level of knowledge and information that are related to electrical engineering. He has done a few related courses such as electronics I & II and programming methodology I & II. Additionally, he is taking COEN315 this semester which may become useful when designing circuits.

1. Evans Durandisse

Evans Durandisse is the other computer engineering student in the team. He also took the software option. His job is to work on the programming algorithm with the help Abdullah Kadhim. He is in his third year.

1. Benjamin Gerlicher

Benjamin Gerlicher is an Electrical/Electronics Engineer student. His main role is the Electrical & Mechanical design of the robot. His work experience and course taken through his studies are appropriate for his participation.

## 6.2 Working Agreement

### 6.2.1 Group Organization

* Two hours of group meeting is scheduled every week before Monday’s lecture
* If necessary, an extra time of group meeting is held on Saturday or Sunday
* Any emergency, conflict, late or absent to the meeting is reported ahead of time
* Sharing tool called DropBox, phone numbers and Emails are used for communication
* All collected data and assignments will be organized and stored in Dropbox folder
* Work hard and be responsible on their own work
* Check emails regularly and stay in contract with group members

### 6.2.2 Meeting Format

* Objectives of the meetings should be defined ahead of time by the leader
* In case the team member has to leave early or come late, the person has to announce it in advance
* All team members should come to meetings organized
* Be prepared for the meeting
* Arrive on time and stay until the end
* The maximum length of meeting is set to three hours.
* Meeting is held on the 10th floor of the Hall building (Laboratory)
* Maximize working time during the meeting
* Concentrate and focus on the topic which has to be discussed and shared.
* Give full attention to the person who speaks
* Do not be shy to express ideas: share ideas and information

### 6.2.3 Decision Making

* Understand the shared ideas, alternatives and information perfectly to make a decision
* All members have to agree with team decision.

### 6.2.4 General Guidelines

* Everyone has equally and fairly divided work to do.
* Everyone respects others and their point of view.
* All team members keep in contact regularly.
* Don’t be lazy
* Invest enough time on this group project
* Meet deadlines
* Avoid unnecessary conflicts
* Try to have fun and enjoy

## 6.3 Work Breakdown Structure

### Capture26.3.1 Initial Work Schedule

Figure Initial Work Schedule

### 6.3.2 Final Work Schedule

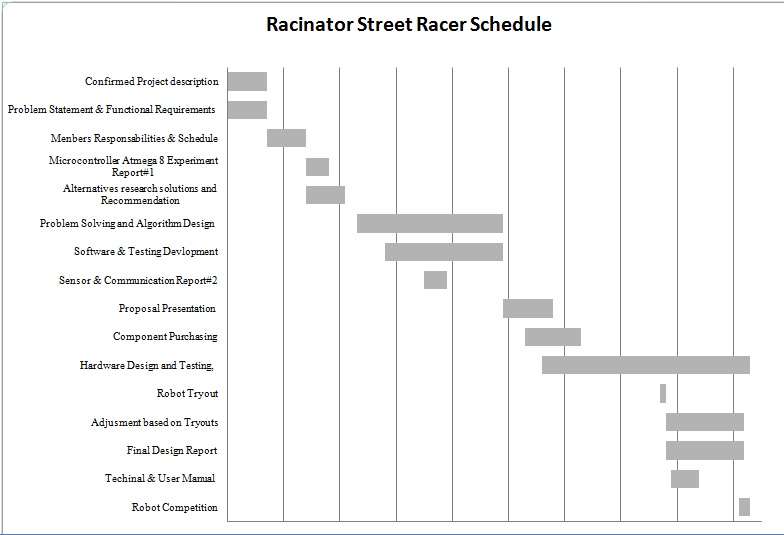


Figure Final Work Schedule

## 6.4 Behavioural Checklist

### 6.4.1 Phase1 – Defining the problem

|  |  |
| --- | --- |
| Decision making | Your Team |
| 1. Recognize constraints and limitations impacting the problems | 4 |
| 1. Worked to define functional requirements in specific terms | 5 |
| 1. Locked for biases and assumptions in order to clarify or redefine the problem | 5 |
| 1. Did not rush to conclusions regarding solutions | 5 |
| 1. Used objective facts and specific research to help define the problem | 4 |
| Project management |  |
| 1. Helped the team clarify goals and objectives | 5 |
| 1. Clearly conveyed performance expectations and standards | 5 |
| 1. Helped the team to clarify individual roles and responsibilities | 5 |
| 1. Demonstrated appropriate flexibility with regard to meeting times and places | 4 |
| 1. Actively participated in creation of the team’s working agreement | 4 |
| Communication |  |
| 1. Consistently asked probing and open-ended questions to clarify issues and increase understanding | 4 |
| 1. Restated and paraphrased what others had said | 4 |
| 1. Gave full attention to others when they were speaking | 5 |
| 1. The written problem statement was well organized with clear main point(s) and supporting information | 3 |
| 1. Ensured that grammar, punctuation, and spelling of written documents were correct | 4 |
| Collaboration |  |
| 1. Consistently solicited input from other team members | 5 |
| 1. Acknowledged others’ contributions and ideas respectfully | 4 |
| 1. Recognized and responded to others’ feelings and concerns | 5 |
| 1. Was friendly toward others and sought to build rapport | 5 |
| 1. Demonstrated patience with others | 5 |

### 6.4.2 Phase2 – Formulating the solution

|  |  |
| --- | --- |
| Decision making | Your Team |
| 1. Identified several alternatives before selecting a solution | 5 |
| 1. Objectively evaluated alternatives in relation to criteria derived from functional requirements and constraints | 5 |
| 1. Avoided rushing to judgment on other’s ideas | 4 |
| 1. Built on other’s ideas and suggestions | 4 |
| 1. Encouraged unusual and creative ideas | 5 |
| Project management |  |
| 1. Thoroughly identified all tasks and activities that had to be completed | 5 |
| 1. Established realistic deadlines and time estimates | 5 |
| 1. Began documenting team progress and action steps | 5 |
| 1. Used a Gantt chart or other planning tool to track progress | 5 |
| 1. Modified tasks and timelines as new information was obtained | 3 |
| Communication |  |
| 1. Gathered information and research from multiple and relevant sources | 4 |
| 1. Clearly articulated how the proposed solution meets design requirements | 3 |
| 1. Appropriately documented references and resources used | 4 |
| 1. Used graphics and diagrams to illustrate points and ideas | 4 |
| Collaboration |  |
| 1. Involved others in discussions and decisions by soliciting their input | 4 |
| 1. Invited questions and comments from team members | 4 |
| 1. Encouraged contrary opinions | 5 |
| 1. Allowed time for debate and discussion | 5 |
| 1. Sought to ensure balanced participation among all team members | 5 |

### 6.4.3 Phase 3 – Developing models and prototypes

|  |  |
| --- | --- |
| Decision making | Your Team |
| 1. Established clear criteria for evaluating design performance | 4 |
| 1. Effectively used design reviews to identify ways of improving the design | 4 |
| 1. Carefully interpreted results from analysis and tests | 5 |
| 1. Was willing to change results or make modifications based on results from analyses | 5 |
| 1. Planned and conducted design analyses in systematic manner | 4 |
| Project management |  |
| 1. Helped identify the right mix of skills and abilities needed to complete work | 4 |
| 1. Clearly defined priorities and work that needed to be completed | 5 |
| 1. Accurately determined what kinds of resources would be needed, including | 4 |
| 1. Followed through on commitments to complete work on time | 5 |
| 1. Helped the team establish and use high-performance standards | 5 |
| Communication |  |
| 1. Developed well-written progress reports | 5 |
| 1. Clearly documented performance results from tests and analyses performed | 5 |
| 1. Solicited feedback and input from others | 4 |
| 1. Provided feedback to others in a constructive and nonthreatening way | 4 |
| 1. Took the time to self-assess performance and improvement opportunities | 5 |
| Collaboration |  |
| 1. Helped to clarify confusion and conflict over roles and responsibilities | 5 |
| 1. Supported others when they needed help or were pressed for time | 5 |
| 1. Recognized and respected individual differences in interpersonal style | 5 |
| 1. Discussed with others how to capitalize on style differences/similarities within the team | 4 |
| 1. Encouraged accountability among team members and discouraged social loading | 5 |

### 6.4.4 Phase 4 – Presenting and implementing the design

|  |  |
| --- | --- |
| Decision making | Your Team |
| 1. Attempted to anticipate potential problems in advance | 5 |
| 1. Developed and used a checklist to help ensure the readiness of your design | 4 |
| 1. Sought input from others outside of your team in order to enhance overall decision quality | 4 |
| 1. Remained calm in the face of unexpected results or problems | 4 |
| 1. Used brainstorming or other idea-generation techniques to solve unexpected problems | 5 |
| Project management |  |
| 1. Regularly reviewed team and individual performance throughout the project | 5 |
| 1. Can articulate lessons learned from this project, both good and bad | 4 |
| 1. Can specify ways to improve performance and results in the future | 5 |
| 1. Kept good records and documented work performed during all phases of the project | 5 |
| 1. Helped the team adhere to high standards throughout the life of the project | 5 |
| Communication |  |
| 1. Carefully prepared presentation materials in advance | 4 |
| 1. Practiced making your presentation before giving it | 4 |
| 1. Provided constructive feedback about what and how material was presented | 5 |
| 1. Effectively used available technology to enhance the quality of your presentation | 5 |
| 1. Prepared a through written final report that was clear and easy to follow | 5 |
| Collaboration |  |
| 1. Ensured that all team members participated in the final presentation | 4 |
| 1. Shared credit for team success and performance | 4 |
| 1. Carefully reflected on the team’s overall effectiveness in terms of collaboration | 4 |
| 1. Helped others to improve their collaboration skills | 5 |
| 1. Celebrated team successes and accomplishments | 5 |

## 6.5 Feedback on Team Function

The working agreement we made was respected by all team members and has been going well so far. This agreement contains all the information we needed to create the potential of success for the project. It was very helpful to manage our project. We have reviewed and modified them by voting if needed.

We regularly had one or two group meetings per week as it is scheduled. The topics had been announced by the leader through emails before every meeting so all team members could come to meetings prepared and organized. Everyone respected others point of view, ideas and information.

Most of the meetings at the beginning of the semester took less than two hours however at the end, the meetings held more than five hour since there are many issues to be discussed.

All team members have followed most of the working agreement. But we sometimes have failed to follow the rules. For example, we all agreed to have group meetings to be held on the 10th floor of the Hall building however most of our meeting was held in the leader’s house. In addition, sometimes, some members could not complete their assigned work on time for our team deadline but it was not a big deal since we helped each other to finish the assigned work.

We equally and fairly divided the work. All team members have been work so hard, keep in contact regularly and put efforts in this project.

# 7.0 TESTING AND RESULTIS

## 7.1 Straight driving calibration

Our car has four small wheels mounted on DC motors. The rate of rotation of each wheel is different. Therefore, we had to adjust them so that the car goes straight. Mechanically we try to align the wheels as straight as possible. However when we load the program for the car to drive forward it tilted on one side. Intuitively, we figured out the cause of this problem; the speed is higher on one side of the car. So by decreasing or increasing the Pulse width modulation, i.e. the voltage signal, on the appropriate side we compensate for the uneven rotation of the wheels. The car went adequately straight afterwards. To follow a road is another matter. The car has to go between two lines. To control the movement of the robot car an H-bridge is used. Our straight driving is essentially monitored by our line sensors. On each side there are two sensors: one in the front and another in the back. The robot goes straight when both sensors are aligned on the lines’ track. At the beginning, the car moves in a jerking motion. After some trials and modification of the sensors position we got the desired results.

## 7.2 Turning calibration

The H-bridge allows for controlling the direction of the wheels rotation. By setting the appropriate ports of the H-bridge chip the left or right wheels can rotate clockwise or counter clockwise. There is another way to turn. When unbalancing the PWM of the left and right wheels the car turns. But, the PWM has to be set precisely to control the turns. After testing, we opted for the reversing of wheels rotation through the H-bridge which is more accurate and less complicated to accomplish. Curvy tracks seem to be easier for our robot than sharp ones. The problem lies more on the sensors than one the ability of the robot to turn. The fast change of the sensors reflectivity in a sharp turn needs a very accurate programming. To make those sharp turns we slow down the robot a little bit. So we neglected speed for maneuverability.

## 7.3 Sensors reading

At the beginning of our project we wanted to follow the track with precision by using analog detection. Using the Analog to Digital Convertor (ADC) of our microprocessor we had hoped to come to the desired outcome. Analog sensors give us continuous reflectivity detection. By having a wide range of shade describing the floor we would have a great advantage of maneuverability. It turns out to be more complicated. Using the ADC for one sensor we arrived at controlling our robot. But, when we put more than one sensor our program did not respond the way we expected. The code seems to be crashing. We abandoned the ADC option. We used the analog sensors as digital. We encounter some problems with that. The voltage range of the sensors is between 0 and 5 volts. But the Atmega8, the microprocessor we are using reads as high (1) values from 3 volts and up, and low (0) from zero to 1 volt. So we have to discard values from 1 to 3 volts. Placing the sensors at the right height and angle to the floor was crucial. Small displacements would give us different values. As far as we were in the project we could not go back with digital sensors, considering the deadline for the competition. So we adjusted our sensors as good as we could, hoping they are going to stay fix all the time.

## 7.4 Troubleshooting

In case some minor problems are encountered, here are some solutions you can follow in order to make your robot work again.

### 7.4.1 Hardware Trouble Shooting

Table HARDWARE Trouble Shooting

|  |  |  |
| --- | --- | --- |
| Problems | Causes | Solutions |
| The power of the robot does not turn on | Batteries are not inserted | Make sure the battery is inserted. If you do not, please insert the battery |
| Batteries are not inserted correctly | Check the battery sign and reinsert the battery |
| Battery power is low | Insert a new battery or recharge it |
| Battery pack is not properly connected to the breadboard | Re-wire the connection as given electrical schematics in the technical manual or call us |
| The robot does not move on the floor | Switch is not placed in “on” position | Place the switch in “on” position |
| The wheels do not rotate | Motor and wheel are not properly connected | Check to see if the motor control wires are connected properly between the breadboard and motor. If not, please call us for help. |
| Software Issue | Contact us |
| The robot does not follow the black electrical tape | Dirt disturbs | Wipe off any dirt on the sensors |
| Sensors are defected | Call us for a replacement kit |
| The robot runs at dark environment | The robot might not work at dark environment. Try to test in light environment |
| The robot does not detect obstacles | Dirt disturbs sensors | Wipe off any dirt on the sensors |
| Sensors are defected | Call us for a replacement kit |
| The robot runs at dark environment | The robot might not work at dark environment. Try to test in light environment |

### 7.4.2 Software Trouble Shooting

Table SOFTWARE Trouble shooting

|  |  |  |
| --- | --- | --- |
| Problems | Causes | Solutions |
| Modified programs cannot be uploaded | USB/ISP is not connected properly | Make sure they are connected to the right ports. |
| Atmega8 Microcontroller is defected | Replace Atmega8 Microcontroller with a new chip and try again to upload programs |
| Software crashes | Unknown | Try to reboot your computer |
| Memory overflow | Atmega8 Microcontroller has limited amount of memory | Reset the memory |

# ANNEX

## A. REFERENCES

1. <http://www.im.fme.vutbr.cz/pdf/14_4_277.full.pdf>
2. [Moravec, Hans](http://en.wikipedia.org/wiki/Hans_Moravec) (1988), *Mind Children*, Harvard University Press, p. 3
3. [Dynamic Vision for Perception and Control of Motion](http://www.springer.com/978-1-84628-637-7), a 2007 book by Ernst D. Dickmanns
4. Cracknell, Arthur P.; Hayes, Ladson (2007) [1991], *Introduction to Remote Sensing* (2 ed.), London: Taylor and Francis
5. Work by Drawn and animated by [Mike1024](http://commons.wikimedia.org/wiki/User:Mike1024), 9 March 2008
6. <http://www.solarbotics.net/>
7. <http://www.robotshop.ca/parallax-qti-line-follower-appkit-for-the-boe-bot-1.html>
8. <http://www.robotshop.ca/pob-technology-line-sensor-1.html>
9. [de Berg et al. (2000)](http://en.wikipedia.org/wiki/Visibility_graph#CITEREFde_Bergvan_KreveldOvermarsSchwarzkopf2000), sections 5.1 and 5.3; [Lozano-Pérez & Wesley (1979)](http://en.wikipedia.org/wiki/Visibility_graph#CITEREFLozano-P.C3.A9rezWesley1979)
10. Work by [Ibma](http://commons.wikimedia.org/w/index.php?title=User:Ibmua&action=edit&redlink=1), 2008(2008)
11. Rolf Schneider, *Convex bodies: the Brunn-Minkowski theory,* Cambridge University Press, Cambridge, 1993(2008)
12. Example graphic for Minkowski addition by [Stephan Beyer](http://commons.wikimedia.org/w/index.php?title=User:Sbeyer&action=edit&redlink=1), 2005-11-25
13. http://www.optekinc.com/datasheets/OPB703-705\_70A-70D-B.PDF

## B.DETAILS OF BUDGET

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Unit Price $** | **Total $** |
| Chassis (112x106mm) | 1 | 5 | 5 |
| Atmega 8 Mircocontroller | 1 | 3 | 3 |
| H-Bridge Motor Driver SN754410N | 1 | 1 | 1 |
| Battery Pack 6xAA, NiMH + Battery Charger | 1 | 20 | 20 |
| Analog Infrared Sensor OPB704 | 7 | 5 | 35 |
| Digital Infrared Sensor GP2D15 | 1 | 15 | 15 |
| GM6 Motor from Solarbotics | 4 | 7 | 28 |
| Resistors | 16 | 0.25 | 4 |
| Capacitors | 3 | 0.25 | 0.75 |
| Push Button | 1 | 0.25 | 0.25 |
| ISP Header | 1 | 2 | 2 |
| LPT/ISP Cable | 1 | 3.5 | 3.5 |
| Voltage Regulator | 1 | 0.5 | 0.5 |
| RW2 31mm Wheels | 4 | 4.5 | 18 |
| Wheel Bracket | 4 | 1.5 | 6 |
| 270 tie point Breadboard | 2 | 7 | 14 |
| **Total $:** | 49 |  | **156** |

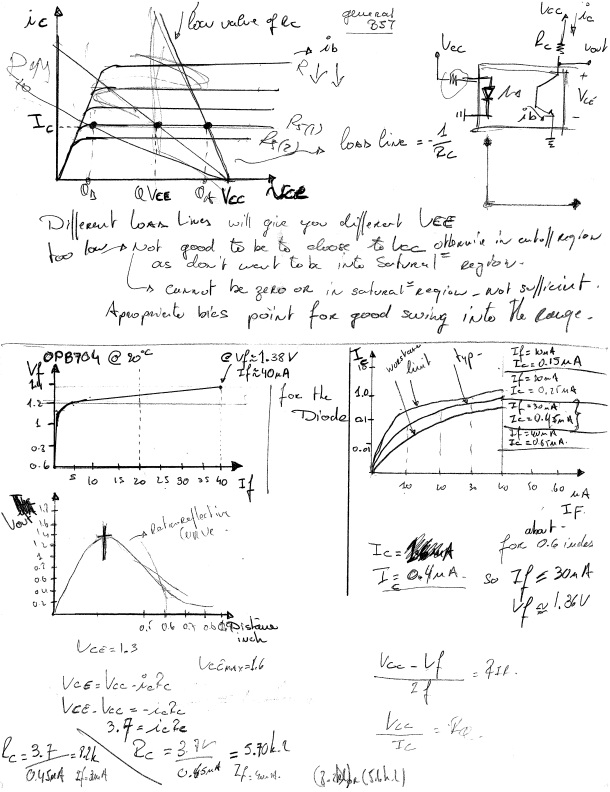
|  |  |  |  |
| --- | --- | --- | --- |
| **Labour Description** | **Hours** | **Hourly Rate** | **Total $** |
| Benjamin Gerlicher - Electronics Engineer | 75 | 15 $/h | 1125 |
| Ouben Jeong - Electrical Engineer | 75 | 15 $/h | 1125 |
| Evans Durandiss – Computer Enginner | 75 | 15 $/h | 1125 |
| Abdullah Kadhim – Computer Engineer | 75 | 15 $/h | 1125 |
| ***Total $:*** |  |  | **4500** |

Figure Bill of Material Budget distribution

Figure Labour vs. Parts Budget Distribution

## C. APPENDIX

### Calculations



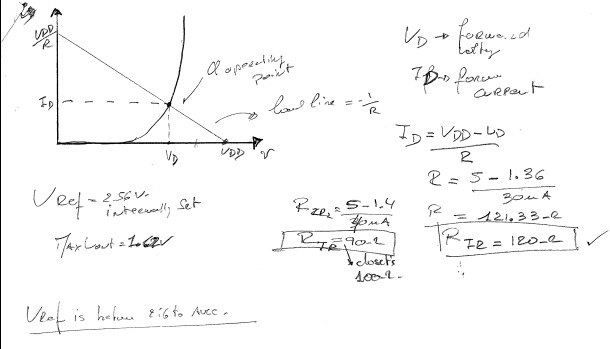


Figure Calculations

### CODE

#### CODE HEADER

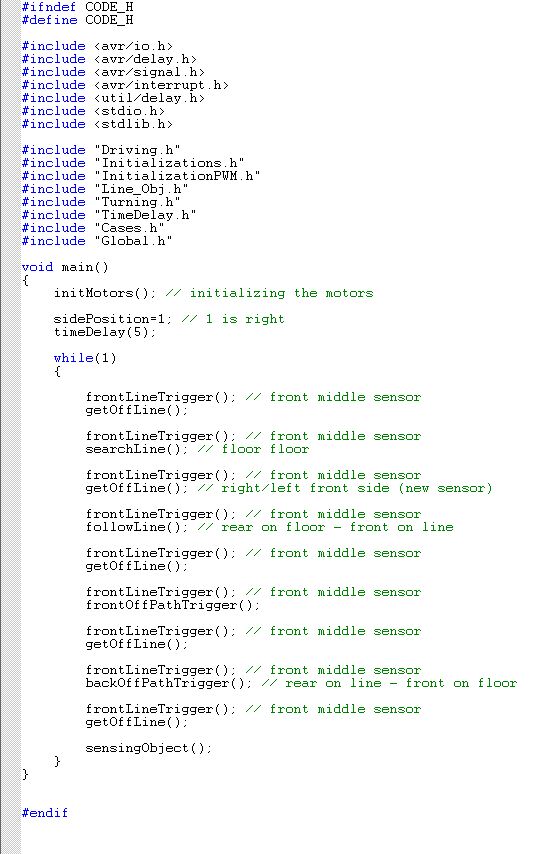


Figure 26 CODE.H

#### CASES HEADER

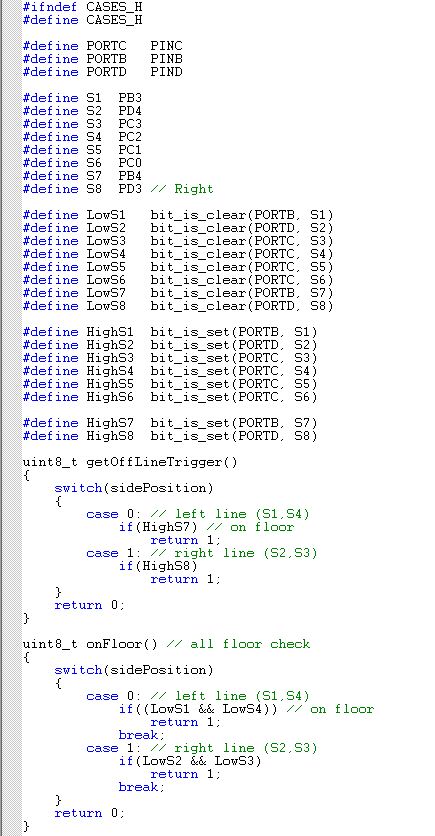




Figure 27CASES.H

#### DRIVING HEADER

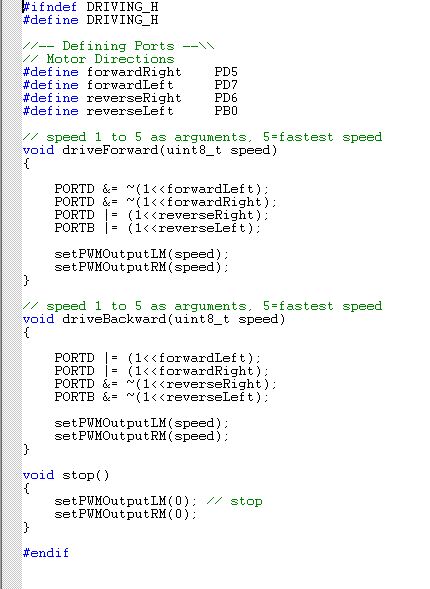


Figure 28 DRIVING.H

#### GLOBAL HEADER

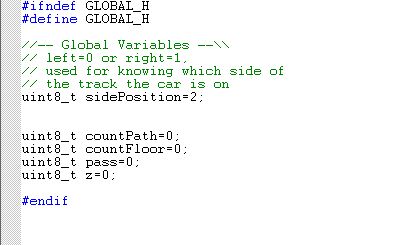


Figure 29 GLOBAL.H

#### INITIALIZATION

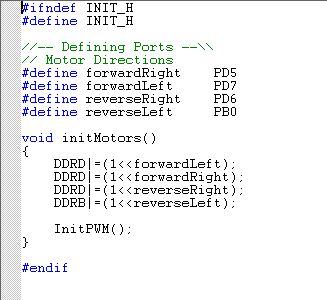


Figure 30 INITI.H

#### INITIALIZATION PWM

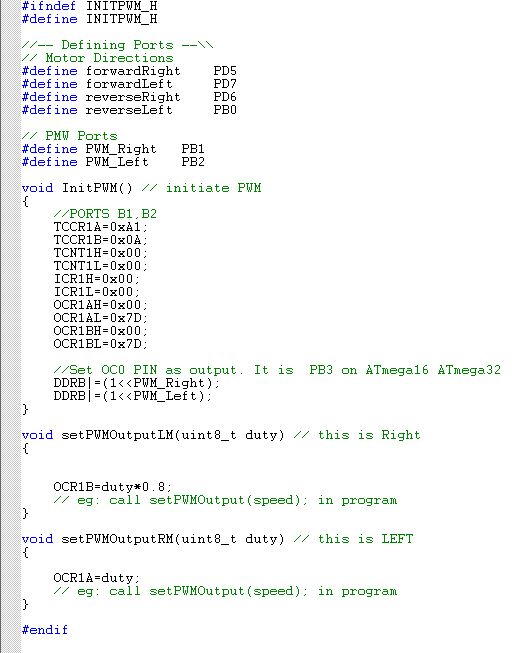
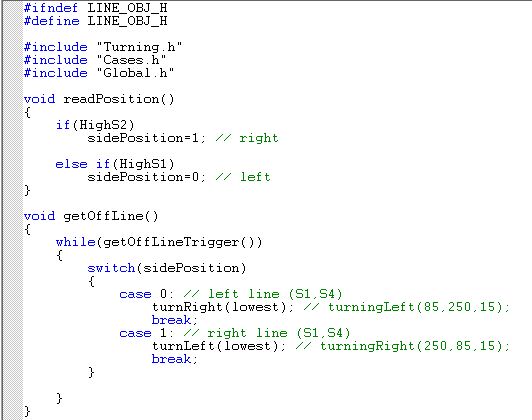


Figure 31 INITPWM.H

#### LINE OBJECT HEADER



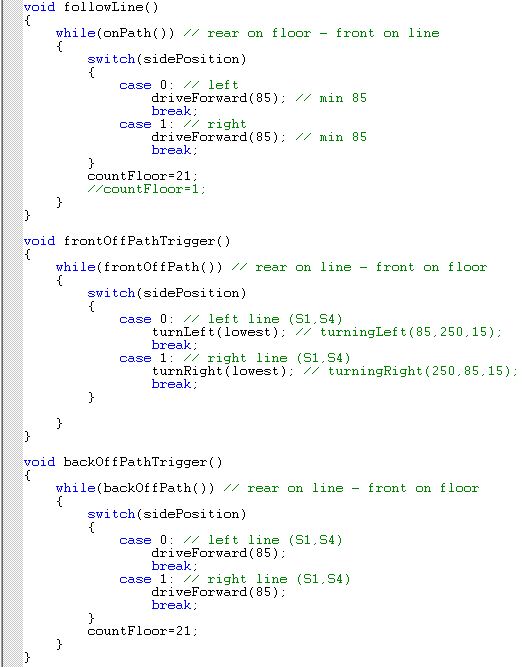


Figure 32 LINE\_OBJ.H

#### TIME DELAY HEADER

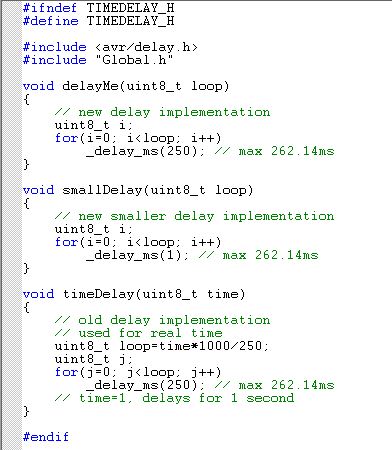
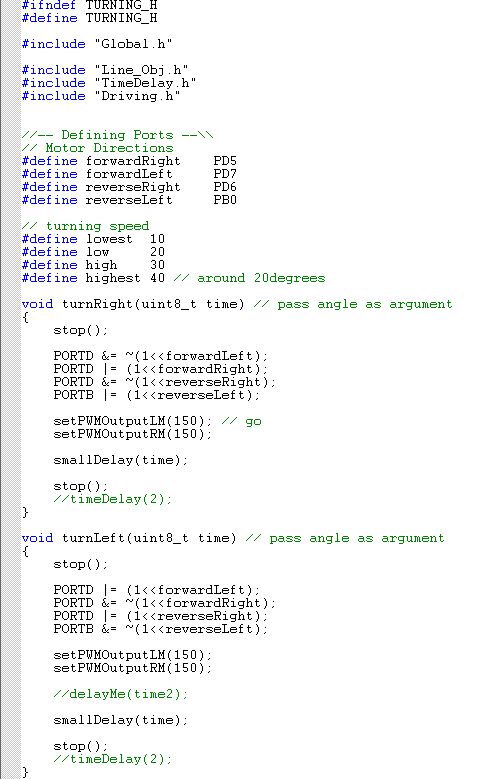


Figure 33 TIMEDELAY.H

#### TURNING HEADER



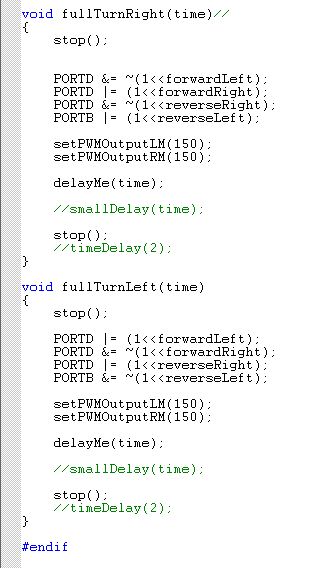


Figure 34 TURNING.H