King Fahd University of Petroleum & Minerals

Computer Engineering Department

ROAMER

Prepared by

Turki Al-Ameeri 984917

For

COE 485: Senior Design Project

Fall 2004-2005 (Term 041)

Project Advisor: Dr. Elrabaa, Mohamed E.

Coordinator: Dr. Adnan Gutub

Nov 3, 2004

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A programmable car (Roamer) has been designed and implemented. The Roamer can be programmed to follow any path by the user with the ability to move forward, left and right. Once the required path is entered and the GO button is pushed, the Roamer becomes completely autonomous and executes the required path. The forward distance is given as multiple of units (from 1 to 9) with practically unlimited number of moves (forward, left or right) in any sequence. With distance and turning calibration capabilities, the Roamer can be programmed to move and turn on any surface. The Roamer achieves a turning resolution of 15° and can be calibrated to do any turn at this resolution. Also, the forward distance unit can be calibrated at a resolution of 9 cm. The complete Roamer has been successfully prototyped and tested. Support two types of turning control for DC motors "differential torque" and "single torque". Reed switches for feedback were utilized to detect and control the distance.

INTRODUCTION

A Roamer is an educational toy, usually a car, which teaches preschoolers *(age 4 to 7)* problem solving, strategic planning, and simple programming skills. The child is usually asked to design and program a path that takes the Roamer through a maze *(e.g. as in figure 1).*

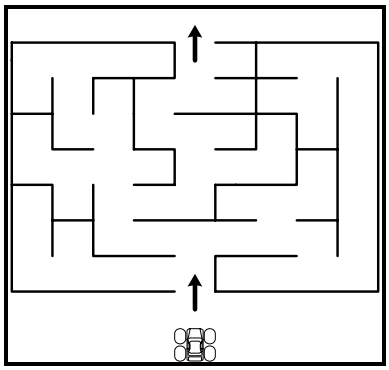

Figure 1:Roamer car goes through a maze.

Once a path has been entered and the GO button is asserted the Roamer should be completely autonomous and execute the required path.

The remaining of this report explains and discusses the ROAMER project from many aspects; specifications, assumptions, how it works, system design, gate level design, operating system, and finally the prototype and testing. Any other details like chips, parts, assembly code …etc are included in the appendix.

I. SYSTEM DESIGN

This section represents the stages required to build a complete ROAMER car. This includes the project specifications, assumptions, block design, gate level design and the operating used to operate and control the car.

A. Project Specifications

- Every thing is built on the car itself.
- Selectable turn angle: Angle is adjustable in the range from 0 to 360˚ at a resolution of 15˚.
- Directions of movement are: Forward, left and right.
- Movement units that can be entered are from 1 to 9 units.
- Adjustable unit is at a resolution of 9 cm.
- Number of entered instructions depends on the available memory.
- The car has an LCD display and a keypad to interface with the user.
- Operating system Response time should be within $1/4^{th}$ of a tire rotation time.

B. Assumptions

- The environment the car can move on is a complete flat floor.
- The speed is constant in any direction. the end and beginning are equal.
- The car doesn't deal with obstacles.
- The engine of the car is always running and doesn't break down.

C. General Design

The following figure shows the block diagram of the ROAMER. The main units used to build the ROAMER car are; DC motors, Movement Control Unit, Central Processing Unit (CPU), Input Entry Unit, Input Display Unit, and Feed Back System.

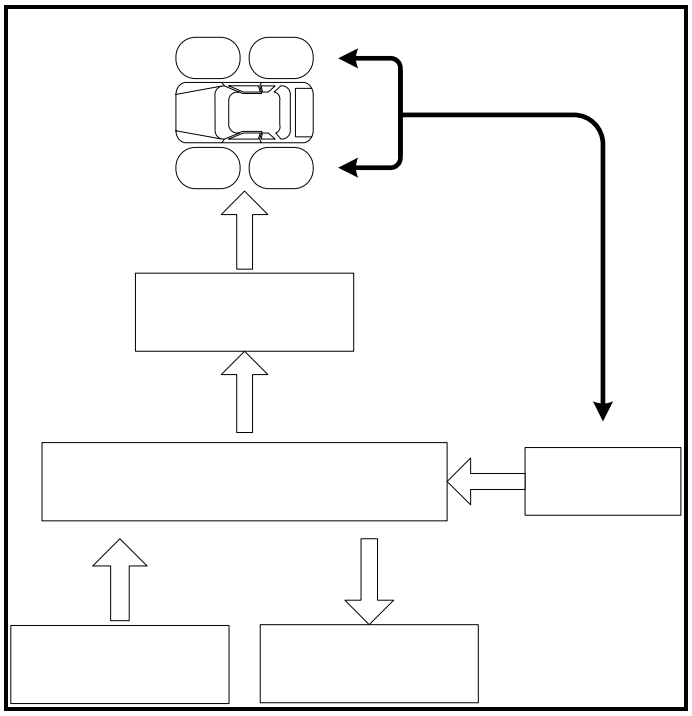

Figure 2: Block diagram

Input Unit:

This is the unit that interacts with the user through a keypad. The user can specify the distance, direction and the actions he/she wants to perform on the car through this unit.

Display Unit:

This unit consists of an LCD display the user can confirm his/her actions and inputs he/she would like to apply on the car.

Control Unit

Central Processing Unit

Central Processing Unit (CPU):

This unit controls all the other units. It takes the input from the input unit and sends it to the display unit to display it on the LCD. It gives the commands to the control unit to perform the right actions on the car. It monitors the movements of the car by observing the feedback system. Also the operating system is stored and executed in this unit.

Feedback System:

This unit is needed to verify that the specified action is performed correctly on the car. Through this unit the CPU can keep monitoring the currently executed action.

Control Unit

It is the interface between the CPU and the car. It takes the command from the CPU and applies it on the car. The action of the control unit is bounded and monitored by the feedback system.

D. Gate Level Design

Two designs before the final one have been developed. The following section discusses in details the three designs, namely DESIGN A, DESIGN B and DESIGN C. A comparison between the three designs that clarifies the final choice will be discussed.

1. DESIGN A

This is the first design used to implement the ROAMER car. The major units used to form the design are: CPU, input unit, store unit, interrupt unit, display unit, time unit, command unit and control unit. The following is a detailed explanation for DESIGN A.

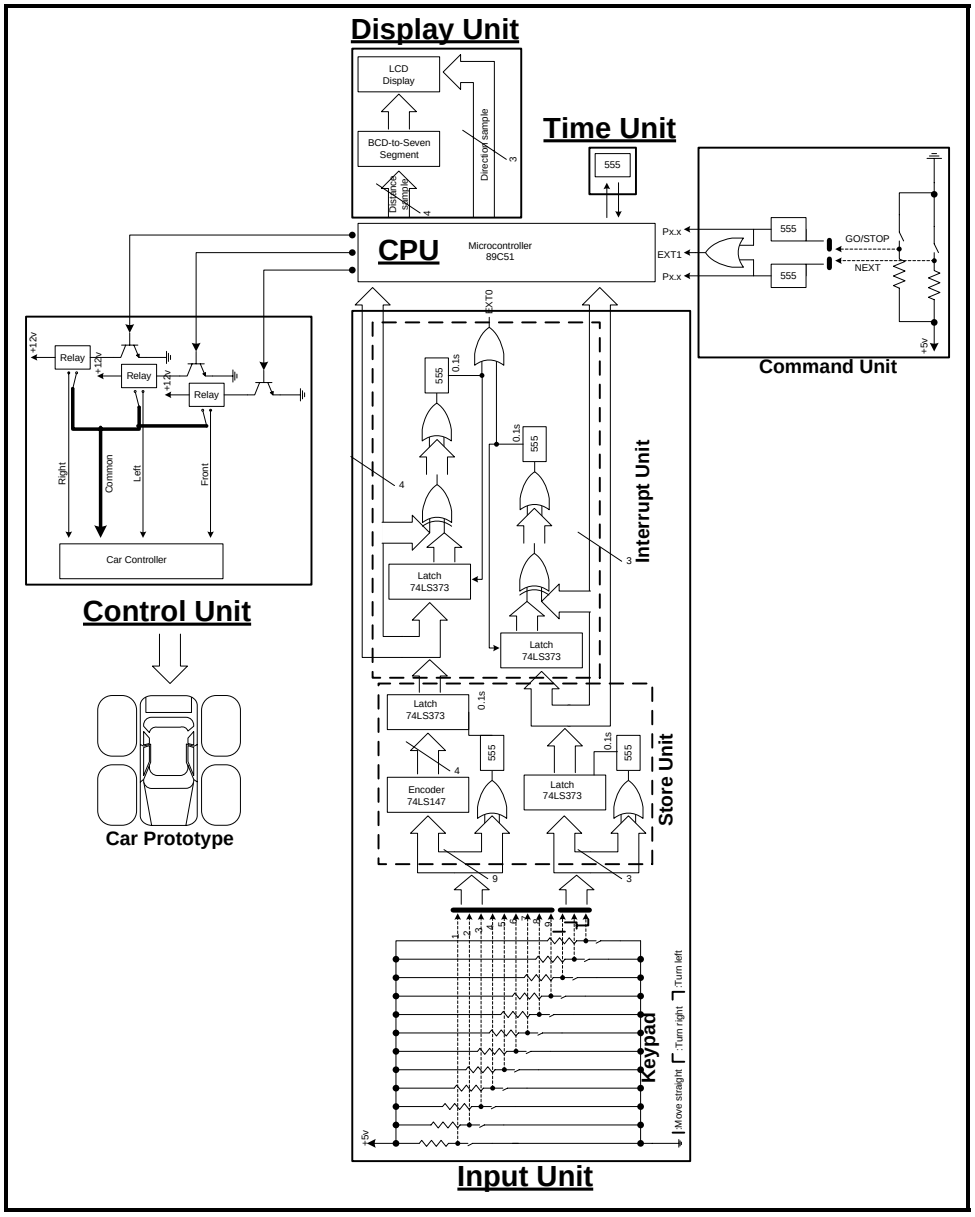

Figure 3: DESIGN A

Input unit:

The input unit in this design consists of four subunits. Keypad, store unit, interrupt unit and command unit. These four units are combined together to complete and perform the input unit functions. The following four paragraphs describe the four subunits:

Keypad consists of press buttons to represent the numbers and directions. The default state for these bottoms is HIGH (+5V). Nine buttons for numbers, three for directions and two for commands that is a total of 14 buttons. When a button is pressed, it generates a low signal.

Store unit encodes the numbers in the keypad buttons using BCD encoder. Then stores the code generated from the current pressed button until it is detected by the interrupt unit which informs the CPU of the new data. The same goes for the direction buttons except that it is not encoded.

Interrupt unit takes the stored code from the store unit and generate an interrupt signal to inform the CPU with the new data. Since the Microcontroller have only 2 interrupt pins, the interrupt unit is originally used to extend the interrupt capacity by reserving only one of the interrupt pins for all the input buttons.

Command unit is used to establish or stop the execution of the stored data and commands. The command buttons are part of the keypad but connected to a different interrupt pin in the microcontroller.

Display unit:

This unit takes two inputs from the CPU; a BCD code that represents the numbers, and three digit code representing direction. While the direction code is plugged directly to the LCD the BCD code is decoded using BCD-to-7_Segment chip to display it on the LCD.

Central processing unit (CPU):

The CPU used here is a Microcontroller AT89C51, it stores/executes the operating system and controls all the other units. The datasheet of the microcontroller is included in appendix A.

Time unit:

It is the feedback system of this design. It is used to adjust and calibrate the movement of the car by calculating the time. But this system suffers unacceptable drawbacks. For example, if the car starts moving slower because of the surface, the time specified for the car to finish one meter may finish before the car can cross even half meter.

Control unit:

This unit is used to control the DC motors of the car according to the command sent from the microcontroller. The command is a three-digit code that specifies the direction of movements.

2. DESIGN B

Based on DESIGN A, the only difference is the input unit. In order to reduce the hardware the store unit and interrupt unit were removed from input unit and their functionality were replaced by software. This puts much weight on the microcontroller and the operating system becomes more complicated, but reducing the cost and space was the first priority. The remaining units are still as same as in DESIGN A. This design still does not satisfy our needs because feedback system still depends on time. Another problem is raised which is the power consumption. Even though the hardware is reduced, the relays which are used in the control unit still consume a lot of power and that became a disadvantage which prevents us from using a battery to operate the car. *(Advantages and disadvantages of this design are included in table 2 after the discussion of DESIGN C).*

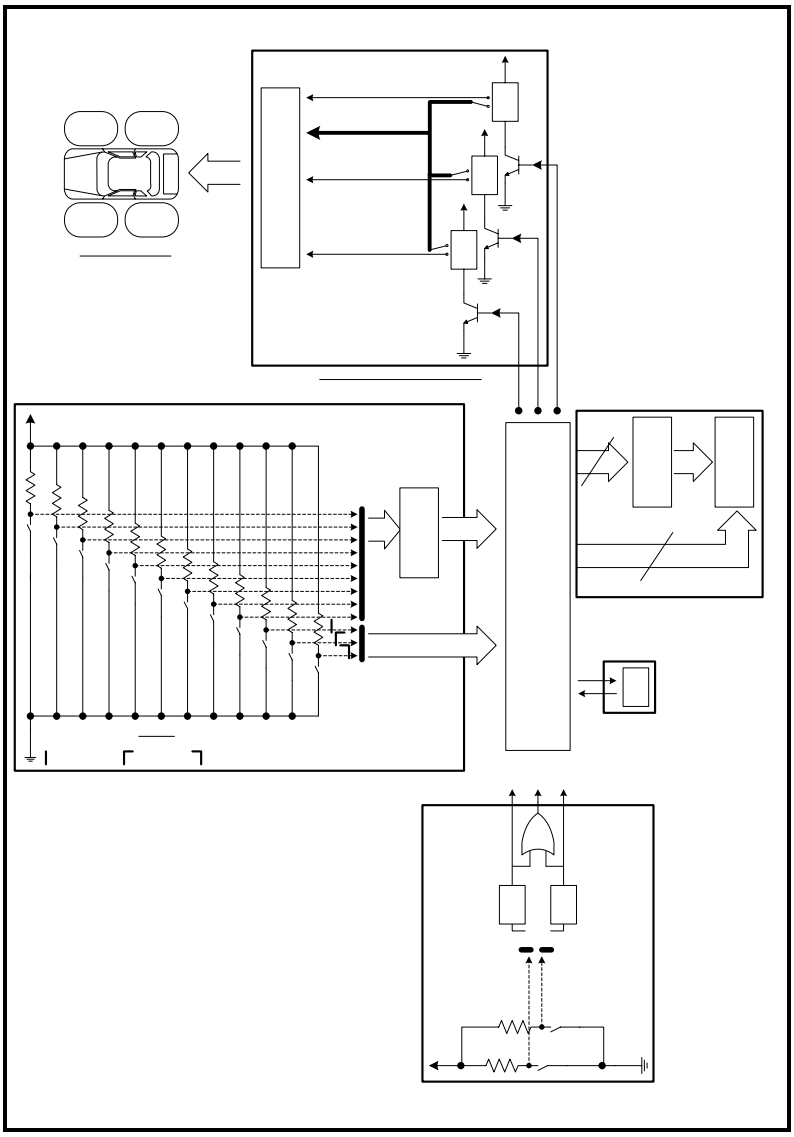

Figure 4: DESIGN B

3. DESIGN C

This is the final and the implemented design. It provides all the function and capabilities needed to operate the ROAMER car in a good form. The following sections discuss the components of DESIGN C in details.

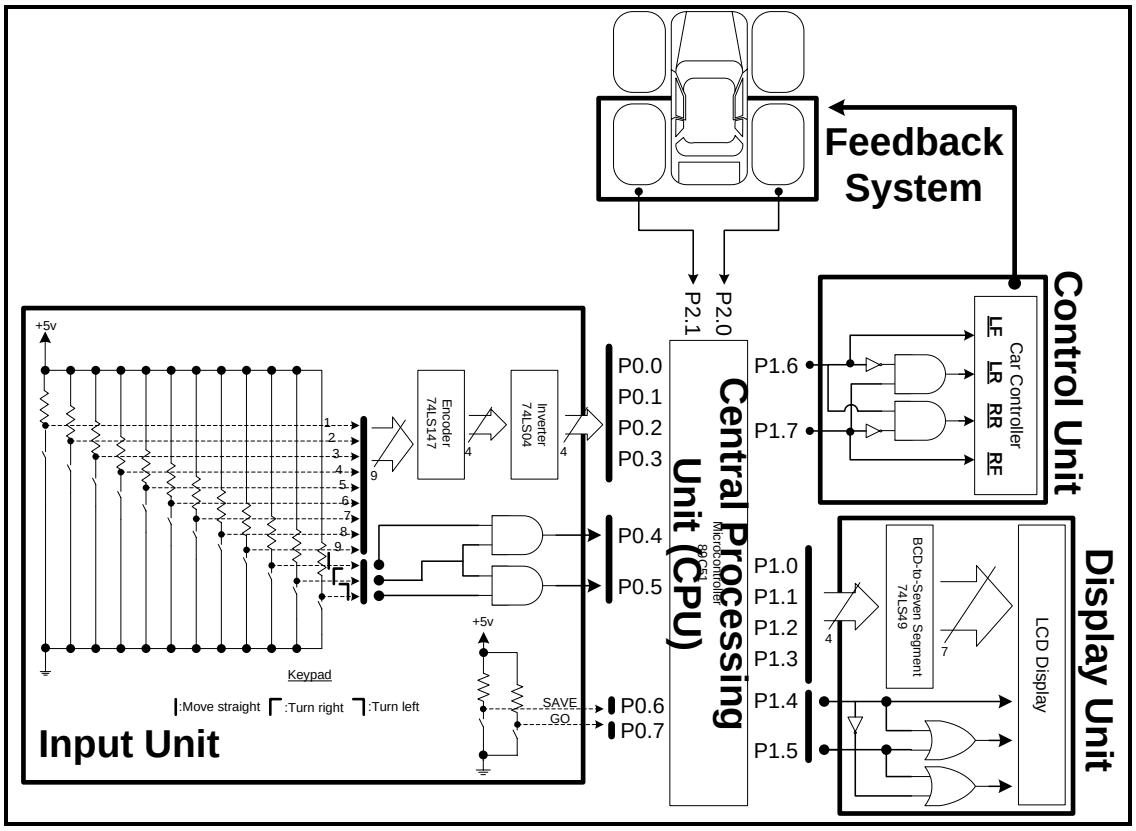

Figure 5: DESIGN C

Input Unit:

This unit consists of press buttons. The default state of these buttons is HIGH (+5V). The numbering buttons are encoded using the BCD system to reduce the pins needed from 9 to 4 and plugged to port 0 of the microcontroller. Two other buttons "save" and "go" plugged to 2 pins of port 0. Also the direction buttons are encoded and reduced from 3 to 2 pins in order to fit in P0.x. that is a total of 8 pins.

Central Processing Unit (CPU):

The CPU is the AT89C51 microcontroller. Port P0.x of the microcontroller is reserved for the input unit, P1.x for any outputs, P2.x for the sensors of the Feedback System and P3.x for the interrupts and the other special functions of the microcontroller.

Display Unit:

This unit takes two inputs from the CPU; a BCD code represents the numbers, and two-digit code represents direction. The direction code is decoded to 3 digit signals before it is plugged to the LCD. The BCD code is decoded using BCD-to-7_Segment chip to display it on the LCD.

Control Unit

Because the relays which were used in DESIGN A & B consume a lot of power, it was decided to remove them in this design and find a better solution. As a result, a new circuit was designed and it functions as follows.

It takes a two-input code from ports P1.6 & P1.7 of the microcontroller and decodes it to four signals that controls the car's DC motors. The codes are demonstrated in *(Table 1)*.

Table 1: Resulted signals from decoding port P1.6 & P1.7.

P1.7	P1.6	LF	LR	RR	RF
0	0	0	1	1	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	1	1

LF: Left motor, moves forward.

LR: Left motor, moves backward.

RR: Right motor, moves forward.

RF: Right motor, moves backward.

Feedback System:

It is a magnetic based feedback system, Used to calculate the perimeter of the wheels. Through that system the microcontroller can detect correctly the crossed distance. This system was proved to be much more effective than the time based feedback system, because it is not affected by the surface or speed. And the car won't stop until it reaches the specified distance.

4. Comparison

This section shows a comparison between the three designs in term of advantages and disadvantages of each one. And from the following table DESIGN C become the best choice for implementing the project.

Table 2: Comparison between designs A, B and C.

	Advantages	**Disadvantages**
DESIGN A	• Minimize the pressure on the CPU.	• Has a lot of hardware which consumes *space*, *power* and *cost.* • Feedback system based on time. • Can't support additional devices (e.g. sensors).
DESIGN B	• less in cost and space compared to DESIGN A.	• Depends heavily on the operating system and the CPU speed. • Feedback system based on time. • Can't support additional devices (e.g. sensors).
DESIGN C	• Balances the functions between the hardware and software. • Less in space, power and cost compared to DESIGN A & B. • Feedback system based on the wheel perimeter. • Can support up to additional six devices (e.g. sensors, cameras …etc).	• More complex operating system is needed to support and control the plugged devices.

II. Operating System

A. Flow chart

To operate the car, a small real time operating system is needed. For that the following chart displays the basic functions needed in the operating system in order to operate the car and respond to the user instructions without delay.

In addition to the main program, an *interrupt driven actions* were designed and taken care of in order to complete and extend the capabilities of the real time operating system.

For Example, *STOP function is designed as interrupt function. So whenever the user request a cancellation for the current operation, the STOP function will interrupt the operating system and take action based on the user request.*

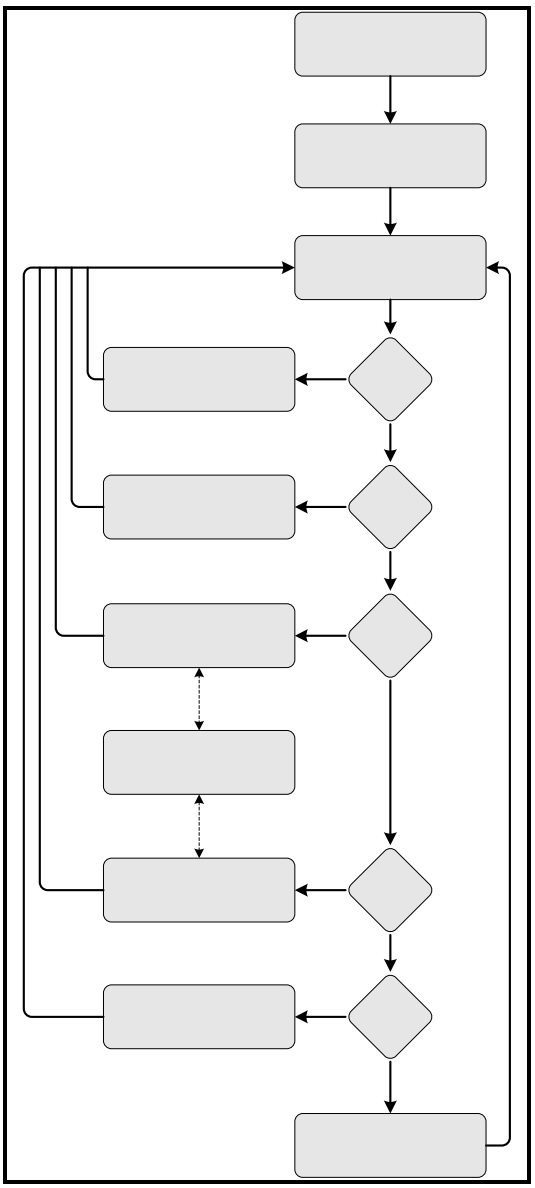

Figure 6 : Flow chart.

Reset function:

Resets the registers (R0, R1, R2, R3, R4, R5, R6, and R7) and the data array to zero. It also reassigns the default values of the movements units in the memory. The result of executing the Reset functions is in the following table….

Table 3: Contents of the registers after executing Reset function.

Variable	Stored value	Description
R0	34H	Address of the Data array [4].
R1	0	Temp.
R2	0	Stores the CALIBRATION Dir.
R3	0	Used to check the input signals to the MC.
R4	0	Temp.
R5	0	Store the new calibration number generated specified by the user.
R6	0	Store the No. of steps.
R7	0	Store the Dir. unit.
Data array [0]	10H	Steps of front unit.
Data array [1]	10H	Steps of right unit.
Data array [2]	10H	Steps of left unit.
Data array [3]	4BH	Data array size.
Data array [4 -97]	0	Stores the path information.

Check function:

Keeps track of all the buttons plugged into the microcontroller (Port 0) and call the appropriate function according to the pressed button.

Display function:

Confirms the button pressed by the user by displaying it on the LCD screen. The display function interacts with the LCD through port 1 because it is reserved for all outputs.

Save function:

Stores the path information into the data array in the microcontroller's RAM and points to the next empty memory address in the RAM.

Go function:

Takes the path information from the data array one-by-one and executes it using the "car controller function" through port 1.

Car-Controller function:

This function can't be executed alone it should be called and executed by other function like "Go" or "Calibration" functions. This function controls the car DC motors by interacting with the car-controller hardware through port 1 of the microcontroller.

Calibration function:

Adjusts and determines the needed wheel rotations to achieve the specified distance. It uses the car-controller function to operate the car. Then it waits for the user signal to stop the car and stores the number of rotations.

B. Program algorithm

In the following section a pseudo code of the program algorithm is written and the assembly code is included in appendix B.

MAIN function

Begin

- Reset all needed parameters
- Call CHECK function

End MAIN

CHECK function

Begin

- Keep checking Port 0 *(Numbers, Directions, NEXT and GO)*
- IF Number Or Direction = 1 THEN
  Call DISPLAY
  ELSE IF NEXT = 1 THEN
  Save and shift to the next parameter in the storage array
  ELSE IF GO = 1 THEN
  Call GO Function

End CHECK

DISPLAY function

Begin

- Display the last pressed button on the LCD.

End STORE&DISPLAY

SAVE function

Begin

- Save the current displayed number and direction
- Shift to the next empty element in the array.

End SAVE function

CALIBRATION function

Begin

- Get the direction of calibration
- Wait for “Go” signal
- Operate the car and calibrate the number of tire rotations till you receive “Stop” signal.
- Store the new number of rotations.

End UNIT CALIBRATION

CAR CONTROLLER function

Begin

- Fetch a command from the data array.
- Execute the command.

End CAR CONTROLLER

III. PROTOTYPE

For this project a successful prototype is made to prove the theory. This section shows the parts used to build the prototype. A detailed description with snapshots for each part is given. Also the datasheets of the used chips and parts for this car are attached in appendix A. After that the procedure of testing the prototype is recorded and included in a CD attached with the report.

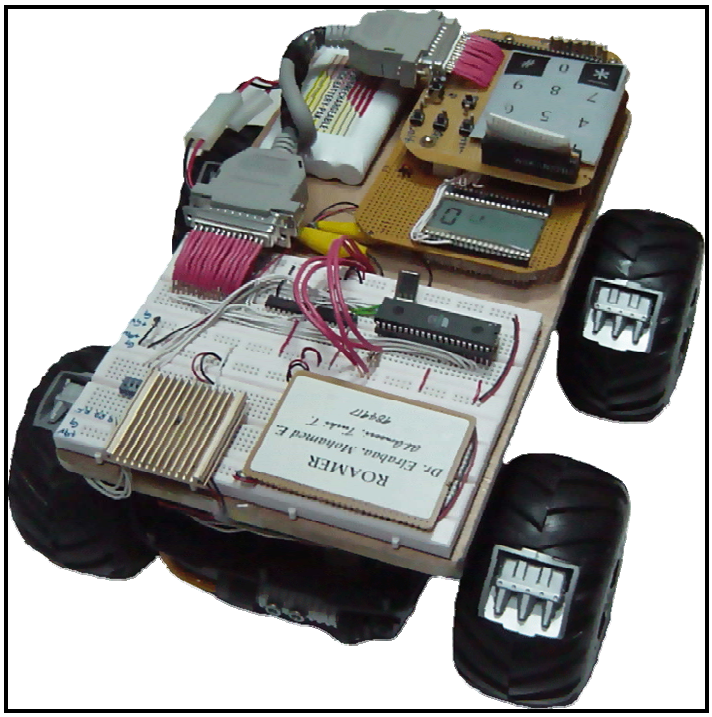

Figure 7: A picture of the final ROAMER prototype.

A. Operating the prototype

1. Press ON/OFF button.
2. Specify the distance 1-9.
3. Specify the direction.
4. Press NEXT for another entry.
5. Press GO/STOP button.

B. Building the prototype

As said earlier in DESIGN C that the prototype is formed using 5 units; input unit, display unit, CPU, feedback system and car controller. Here is a detailed description for parts used to form each unit.

Input Unit:

Typical phone's keypad with 12 button (9 for numbers and 3 for directions) and 5 additional buttons for (save, go, calibration, stop and reset). The default state for all the button is HIGH (+5V).

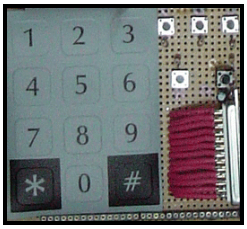

Figure 8: Keypad.

Central Processing Unit (CPU):

The CPU of this project is the AT89C51 Microcontroller. It has an on chip code memory equals 4K and 128 byte of data RAM.

Figure 9: AT89C51 Microcontroller.

Display Unit:

A 3.5in LCD with 7-segment display form is used here. The job of the LCD is to display the current pressed number and direction after receiving it from the CPU to confirm the action of the user. The symbols displayed are as in *Table 4*.

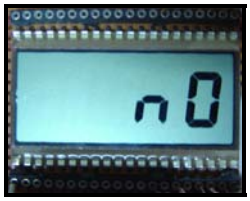

Figure 10: LCD Display.

Table 4: Symbols defenition.

2	
1	
3	
7	

Control Unit

It is a small circuit which acts as an interface between microcontroller and the car's DC motors. It supports all needed directions forward, right and left.

Figure 11: Control unit.

It supports two types of turning a "single turn" and "differential turn". The single turn is applied by enabling only one motor and fixing the other for turning *(Figure 12, shows the jumpers settings to enable the "single turn").*

Figure 12: Mode of "single turn".

The differential turn is applied by enabling both motors but with different directions for turning *(Figure 13, shows the jumpers settings to enable the "differential turn").* The single turn is better for saving power, but differential turn is better to achieve smaller angle fo turn.

Figure 13: Mode of "Differential turn".

Feedback System:

Two solutions were suggested to replace the "time based" feedback system. First, an attached cable that acts like a switch and it is fixed on the car's wheel. But it was not a good choice because the quality was not reliable and only two sensing point give a chance of error for half wheel (18 cm).

Figure 14: Attached cable.

The second solution is a magnetic based feedback system. There are 4 magnets fixed on each back wheel of the car *(Figure15)*. Those magnets are the sensing points for the Feedback System which uses magnetic sensors fixed on the car's body *(Figure16).*

Figure 15: Four magnets fix on the back wheel.

Figure 16: Magnetec sensor fix on the car's body.

That means for each four signals sent from the Feedback system to the CPU, a complete round is done by the car's wheel. Since the perimeter of the wheel is equal to 36 cm that means for each 4 signals a 36 cm is passed. This mechanism was proved to be a good solution for calculating the distance also the response time is within the requirement.

After many experiments, it is found that more than four sensing point would make the magnets closer to each other and they start acting like one whole magnet. Besides, the arrangement of the magnets is not arbitrary; For example, In *Figure.17* each magnet holds the same polarity as its nearby magnet. As a result, an isolation field is created between the two magnets.

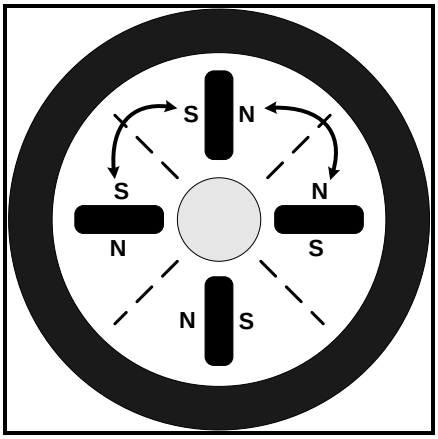

Figure 17: Magnets holds the same polarity as thir nearby.

But in *Figure 18* each magnet holds a different polarity from its nearby magnet. As a result, a virtual magnet is created between the two magnets and is acting as a sensing point. The problem is that the virtual magnet is NOT stable and is moving and affected by the surroundings specially if there is a current or anther magnet near to it.

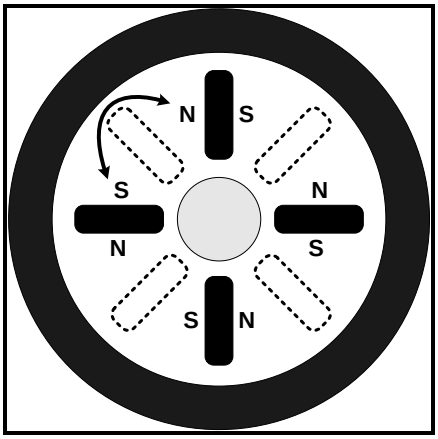

Figure 18: Magnets holds the different polarity from thir nearby.

VI. TESTING

The test used for the prototype is demonstrated in a CD, and goes as follows:

1. Draw a specific path.
2. Program the car to move the same as the drown path.
3. Compare the resulted path with the drawn path.

CONCLUSION

At the end the whole picture of the project became clearer and many aspects appeared to give a high performance if replaced For example, a car with smaller wheels would give a high accuracy if used instead of the current ones. Finally I would like to thank all the people who supported me. And special thanks to Dr.Elrabaa who was supporting me from the beginning until the end of the project.