San Jose State University
Department of Electrical Engineering

EE198B - Senior Project

Wireless Navigation Using RFID Network

By

Ivette Betancourt

Nima Pakravan

Lawrence Shortbull

Nathaniel Angat

Project Advisor: Dr. Ray Kwok

TABLE OF CONTENTS

TABLE OF FIGURES

1 Abstract

This project presents a first look into robotic navigation through a grid of RFID programmable tags. Applications can vary from automated warehouses to management of merchandise in transit through the use a robot that can retrieve and store information on RFID tags. This report will introduce the design and implementation of a robot with integrated RFID reader.

2 Introduction

Ever since the early stages of development of the robot in the mid 1950's, many researches have been made to enhance its performance and to reduce the need for human supervision of the machines as much as possible. .

In this project, we will introduce a navigating robot, which will navigate its way around the RFID based floor without the help of a human. The objective of this project is to use data collected from RFID tags, which are embedded on the ground and use it to navigate a robot. In a real situation the robot would be a version capable of handling cargo. The tags could be either built into the floor of a warehouse or stuck to the terrain to form a just-in-time storage ground. The robot would be able to find specific locations and execute specific tasks based on the information that it collects from the tags.

It is a perfect tool for the warehouse environment, as it will not malfunction in dusty or greasy area. Further, RFID technology is free of orientation problem. Therefore the robot can retrieve the information from the ID tags without having to line itself up with the orientation of the tags as it is in the case of laser. RFID technology is currently being used in the many warehouses for the inventory purposes. Robot is also being used in many similar cases to increase the speed, accuracy, and efficiency, while decreasing the losses as a result of job injury. The combination of these two technologies introduces a new product that can bring a higher speed and higher efficiency to many industries.

The project starts with detecting an RFID tag on a grid. The information received by the RFID reader is send to a microcontroller, which determines the location of the tag on the grid. Based on the location of the tag the microcontroller will send the appropriate command to the robot. The robot will navigate on the grid until it reaches its destination.

3 Organization Requirements

The project was divided in four parts. Each module could be implemented separately from each other, so we had the flexibility of developing all four parts simultaneously.

3.1 Group Members Responsibilities

- Nima Pakravan was in charge of designing, implementing, and testing an antenna with a higher reading range than the antenna purchased from Skyetek. In addition, he was in charge of testing the RFID reading system.
- Lawrence Shortbull was in charge of designing, implementing, and testing the communication link between the RFID reader and the robot.
- Nathaniel Angat was in charge of designing, implementing, and testing a navigation algorithm using a microcontroller.

- Ivette Betancourt was in charge of designing, implementing, and testing the chassis, motor controller, and motor driver of the robot.

4 Engineering Process

4.1 RFID System and Antenna

4.1.1 Design

After conducting extensive research during the first semester of this project, Skyetek Reader was chosen as the reader of this project. Skyetek reader operates at 13.56MHz and uses ISO-15693 tag transmission protocol. Data modulation is done using amplitude shift keying (ASK) and Manchester system is used for encoding the data.

In order to increase the reading range of the RFID reader, design of an antenna was considered over using the antenna provided by Skyetek. Loop antenna was chosen as our primary design since they are not easily detuned by hand effects and motions. They are used in a variety of handheld devices such as car alarm and garage door opener. Loop antennas are also not ground plane dependant.

4.1.2 Equipment Requirement

4.1.2.1 Design Research

- Books, Internet
- Other designs and approaches for low frequency antennas
- Skyetek antenna design approach

4.1.2.2 Hardware

- Standard coated copper wire – size 20
- Glue gun
- Wooden block
- Capacitors

4.1.2.3 Testing

- Network analyzer – network analyzer was used as a primary source for tuning the antenna to the resonant frequency and measuring impedance.

4.1.3 Material and Capital Required Per Unit Module

The table below presents the cost of the RFID reader and transponders.

Description	Quantity	Cost
Skyetek – M1-5 Reader	1	$134.00
TI Tag-it transponders	25	$112.25
	Total	$246.25

Table 4-1 Material cost for RFID system

The table below presents the material and capital required to construct the antenna.

Description	Quantity	Cost
Role of copper wire – 20AWG	1	$3.35
Wooden block	1	$5
Capacitors	3	$1.50
	Total	$9.85

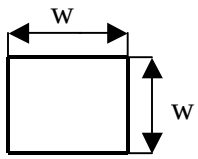
Table 4-2 Material cost for antenna

4.1.4 Process of implementation

Choosing the loop dimensions was the first design concern. The loop was to be chosen so it would match the size of the robot while making it as big as possible. Bigger loop antenna would provide better gain and also a larger bandwidth.
The inductance and capacitance of the antenna can be calculated from the equations below:

$$L_{square}[nH] = (N^2)(20.32)w[\ln(\frac{w}{a}) - 0.774] \qquad 4\text{-}1\text{-}1$$

N = number of turns
w = length of one side (inches)
a = wire radius (inches)



$$C[F] = \frac{1}{Lw^2} = \frac{1}{L(2pf)^2} \qquad 4\text{-}1\text{-}2$$

ω = angular frequency (radian)

Dimensions of the antenna (w) were chosen in the biggest size possible which is 3.6 inches on each side. Wire radius (a) for size 20 copper wires is given by AWG as 0.0168 inches. Knowing these two parameters, inductance and capacitance of the antenna can be calculated for various loop sizes.

Capacitive Reactance of the antenna can be calculated.

$$X_C = \frac{1}{2pf.C} \qquad 4\text{-}1\text{-}3$$

Inductive Reactance is also calculated by:

$$X_L = 2pf.L \qquad 4\text{-}1\text{-}4$$

Further, the DC resistance of the coil is obtained by:

$$R_{DC} = \frac{1}{sS} \qquad 4\text{-}1\text{-}5$$

S = cross-section area.

For copper wire, the loss is approximated by the DC resistance of the coil, when the wire radius is greater than $0.066/\sqrt{f}$ cm.

4.1.4.1 Calculations

Table below shows the inductance and capacitance calculations for various loop sizes.

Square Design Capacitance	Inductance F	Inductance nF	# Turns	length side (inc)	wire radius
4.09986E-10	3.3601E-07	336.0098305	1	3.6	0.0168
1.63994E-11	8.40025E-06	8400.245763	5	3.6	0.0168
5.06155E-12	2.72168E-05	27216.79627	9	3.6	0.0168
1.02496E-12	0.000134404	134403.9322	20	3.6	0.0168
4.5554E-13	0.000302409	302408.8475	30	3.6	0.0168
1.63994E-13	0.000840025	840024.5763	50	3.6	0.0168
4.09986E-14	0.003360098	3360098.305	100	3.6	0.0168

Table 4-3 Capacitor Calculation

The increasing number of loops increases the inductance and therefore, decreases the capacitance. Smallest size capacitor that can be purchased is within a few pico-Farads. That imposed a limit on the # of loops that could be used in the design.

4.1.4.2 Problems

The primary issue in designing this antenna was tuning it to the resonant frequency. The wavelength is calculated at 13.56MHz:

$$l(m) = \frac{C}{f} = \frac{3E8}{13.56E6} = 22.12 \qquad \text{4-1-5}$$

Numerous designs were attempted at quarter wavelength, half wavelength, and full wavelength (near 60 loops). Bandwidth of only 14 KHz for the resonant frequency of the reader, and narrow bandwidth of the antenna made the design and tuning the antenna very difficult.

Various capacitor connections were also made to adjust the frequency and matching. Capacitors were connected in parallel, in series, or both to the circuit for each size to achieve the goal.

4.1.4.3 Final Design

Final design was achieved with 9 loops antenna. Capacitors were connected in series to ground. As seen in table 4-1-1, the inductance and capacitance for this size antenna is:

L = 27.2 uH
C= 5.06 pF

Two 10pF capacitors were connected in series with a variable capacitor forming a series resonant circuit. Final design is shown in figure 4-1.

Figure 4-1 Final Antenna

Calculations of other parameters are as follows:

From equation 4-1-3, $X_C = \frac{1}{2pf_0C} = 2319.6$

From equation 4-1-4, $X_L = 2pf_0L = 2317.4$

Therefore, the impedance is minimized since $X_L \approx X_C$, causing resonance condition.

From equation 4-1-5,

$$R_{DC} = \frac{1}{sS} = \frac{1}{(5.9E5\frac{1}{\Omega.cm})(0.136cm)} = 1.25E-5\frac{\Omega}{cm} = 3.2E-5\frac{\Omega}{inch}$$

4.1.5 Process of Simulation

Network analyzer was used as a primary testing equipment for tuning and matching the antenna. The antenna was first tuned to the resonant frequency and then the impedance and radiation pattern was analyzed.

4.1.5.1 Impedance

Impedance of the antenna was measured using the Network Analyzer. As shown in figure 4-2, Z = 29 – 10j.

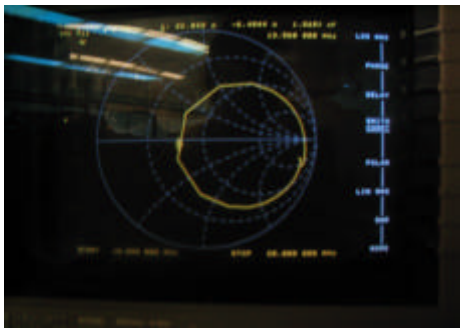


Figure 4-2 Antenna Impedance

After comparing to the impedance of the reader, it was concluded that matching network would not be necessary since the two impedances were very much close to each other. This could also be predicted since a series resonant circuit results in minimum impedance at the resonant frequency.

To conduct a test, the antenna was connected to the reader and a reading simulation was done using Skyeware software. The reading distance was measured to be about 2 inches, as compared to 1.5 inches using the Skyetek antenna.
Better reading distance can be obtained by adding an amplifier between the reader and the antenna.

4.1.5.2 Radiation Pattern

Connecting the antenna to S1 port of the network analyzer, and the Skyetek antenna to S2 port, the radiation pattern was measured every 30 degrees. Using S21 parameter, antenna's radiation was measured horizontally and vertically, both in front and back of the antenna at 30cm distance. Table below shows the result.
0 degree indicates an angle perpendicular to the center of the antenna.

Horizontal (Front)		Horizontal (Rear)	
Angle (degree)	Gain (dB)	Angle (degree)	Gain (dB)
90	-36	90	-44
60	-27	60	-27
30	-21	30	-22
0	-18	0	-18
-30	-20	-30	-21
-60	-24	-60	-26
-90	-40	-90	-42
Vertical (Front)		**Vertical (Rear)**	
Angle (degree)	Gain (dB)	Angle (degree)	Gain (dB)
90	-56	90	-50
60	-30	60	-27
30	-23	30	-24
0	-18	0	-18
-30	-24	-30	-22
-60	-31	-60	-30
-90	-54	-90	-52

Table 4-4 Radiation Measurements (S21)

Using excel, the data was plotted as shown in figure 4-3 and 4-4.

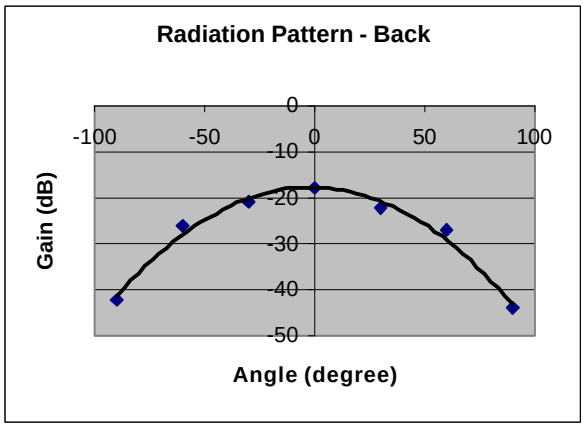


Figure 4-3 Radiation Pattern - Back

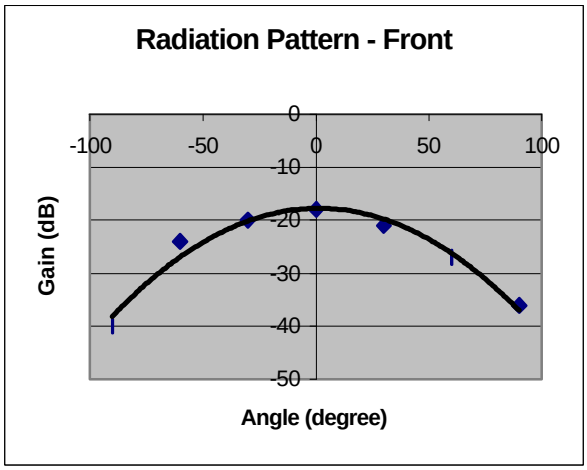


Figure 4-4 Radiation Pattern - Front

Analyzing the radiation pattern both in front and back of the antenna, it can be concluded that antenna constructs an oblong shape radiation pattern both in front and rear. Such pattern can be simulated as in figure 4-4.1.

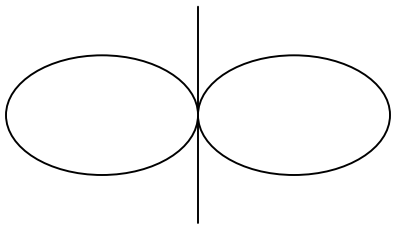


Figure 4-4.1

4.1.6 End Results

A simple loop antenna was made in order to enhance the reading distance of the reader. This distance was in fact increased for about half an inch as compared to the Skyetek antenna. Antenna matching was unnecessary as the reader and antenna had close impedance. Examining the radiation pattern, it was concluded that antenna constructs two oblong shapes both in front and rear, which is the case in most square loop antennas.

4.2 RFID System Communication

4.2.1 Design

4.2.2 Equipment Requirement

4.2.3 Material and Capital Required Per Unit Module

4.2.4 Process of Implementation

4.2.5 Process of Simulation

4.2.6 End Results

4.3 Navigation Algorithm

4.3.1 Design

There were two versions of the navigation algorithm that were attempted. The first was based on a staggered grid where every other row was offset by half the distance between tags. Developing the navigation code for this grid proved to be more cumbersome because this required the robot to make 30°, 60°, and 90° turns. The final version of the navigation code was based on a 5x5 matrix with no staggered rows. This version only required the robot to make 90° turns.

4.3.2 Equipment Requirements (Hardware and Software)

The PIC 16F628 8-bit microcontroller by Microchip was the microcontroller selected for this application. The PIC 16F628, which replaced the PIC 16F84, is a widely used microcontroller that is flexible enough for a variety of applications. With its 2Kbytes of program memory, 224 bytes data memory, 128 bytes in the data EEPROM, the PIC 16F628 was the logical choice for this particular application.

Hi Tech Software's PICC Lite is a free ANSI C Compiler for various Microchip microcontrollers. This application translates source code written in the C language into assembly language, which can be programmed into the microcontroller's Flash memory. The 16F628 wasn't specifically associated with PICC Lite, but the 16F627 was. The only difference between the 16F628 and the 16F627 is that the 16F628 has more memory. So when programming the 16F628, the 16F627 must be selected in PICC lite.

MPLab Integrated Development Environment (IDE) is an integrated toolset for the development of microcontroller applications utilizing Microchip's microcontrollers of the PIC12/16/17/18 and dsPIC families of microcontrollers. MPLab provides an environment for application development and enhanced debugging. It also serves as a central, unified graphical user interface for other Microchip and Third Party software and hardware development tools.

4.3.3 Material and Capital Required Per Unit Module

The table below shows the cost of material and capital required per unit module.

Description	Quantity	Cost
Microcontroller	1	$3.95
PICC Lite C Compiler	1	Free
MPLab IDE	1	Free

Table 4-5 Material Cost

4.3.4 Process of Implementation

4.3.4.1 RFID Grid

The RFID grid in which the robot will be navigating in is an area that measures 120cm by 120cm. The RFID tags are place in a 5x5 configuration. The spacing between tags is 30cm, roughly the distance of 3 antenna lengths. Horizontal rows and vertical columns are numbered 1 through 5. Figure 4-5 shows the layout of the RFID grid. This was deemed to be the optimal configuration based on the range of the RFID reader. During initial testing, the reader demonstrated a reading range of approximately 3cms. This limited the distance we could space the tags from each other. If the tags were spaced any further from each other, the probability of the robot not hitting a tag would increase.

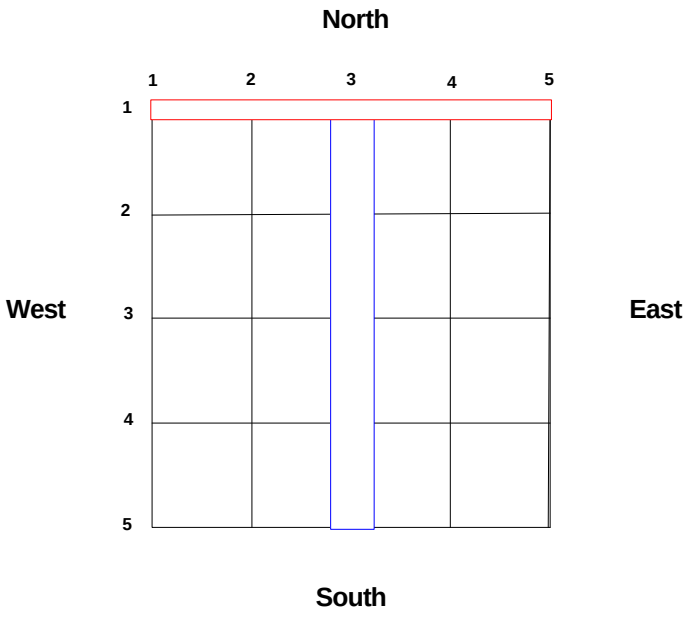


Figure 4-5 RFID Grid

4.3.4.2 Navigation Algorithm

Navigating in the grid is a systematic process. It is assumed that the robot is first placed on the outer edges of the RFID grid, facing the center. The robot will then proceed to navigate itself and find the tag located in row 1, column 3.The navigation algorithm starts by first determining what direction the robot is facing, and then selects the appropriate sub-algorithm.

4.3.4.3 Direction Finding

The flowchart to determining direction is shown in Figure 4-6. When the robot is first placed anywhere on the edge of the grid, facing the center, the RFID microcontroller

(the microcontroller tasked to direct the RFID reader to obtain tag information) obtains tag information from reader. This tag information is the identification number

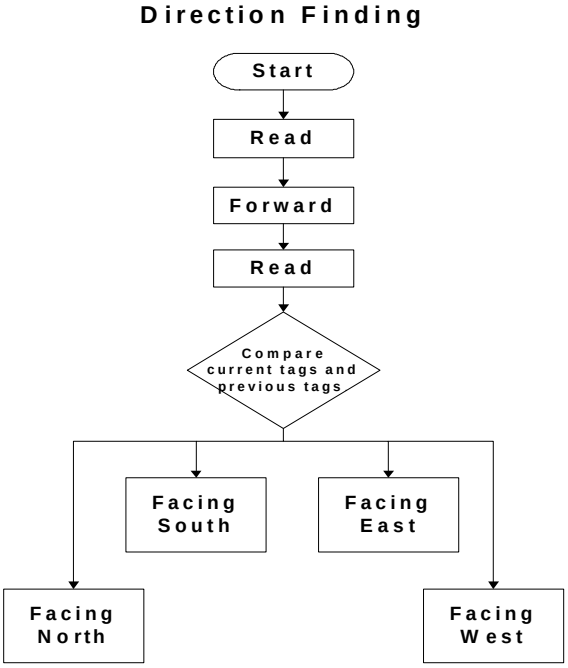


Figure 4-6 Direction Finding Flowchart

found on the tag. The RFID microcontroller then matches the retrieved tag information to a pre-assigned list of where this particular tag will be found on the grid. The tag value from the pre-assigned list is the information that will be sent to the NAV microcontroller (the microcontroller tasked to perform the navigation algorithm). The converted tag information will now be labeled as the previous tag info and put into the navigation algorithm memory for use later in the process.

After the robot has retrieved and converted the initial tag information, it will proceed forward until it finds a new tag. The NAV microcontroller sends the command to the robot microcontroller (the microcontroller tasked to control the robot) which tells the robot to move forward. When a new tag is found, the RFID microcontroller tells the NAV microcontroller to tell the robot microcontroller to stop. The RFID microcontroller will then retrieve and convert the new tag information and send it to the NAV microcontroller. The NAV microcontroller will then label this new tag information as current tag information. After this process of obtaining previous and current tag information, the navigation algorithm can now determine what direction the robot has been placed.

The navigation algorithm compares the current tag information (ctagrow, ctagcol) with the previous tag information (ptagrow, ptagcol) to determine the robot's initial direction. If ctagcol = ptagcol and ctagrow < ptagrow, the robot is facing North. If

ctagcol = ptagcol and ctagrow > ptagrow, the robot is facing South. If ctagrow = ptagrow and ctagcol > ptagcol, the robot is facing East. If ctagrow = ptagrow and ctagcol < ptagcol, the robot is facing West. Based on this initial direction, the algorithm will proceed to one of the four sub-routines labeled Facing North, Facing South, Facing East, and Facing West.

4.3.4.4 Facing North

After the algorithm has determined that the robot is facing North, a command is sent to the robot microcontroller to move forward until a tag is found. The current tag information is labeled as previous tag information for use later in the process. When a new tag is found, new tag information obtained and labeled as current tag information. The navigation algorithm checks if the current tag row value is equal to 1. If not, the algorithm goes into a loop of moving forward and retrieving new tag information until the current row tag value is equal to 1. When the current tag row value is 1, the algorithm checks if the current tag column value is equal to 3. If it is, the target tag has been found. If it isn't, another check is made. If the current tag column value is less than 3, the robot is told to make a right turn. If the column value is greater than 3, a right turn is made. After the robot clears the turn, the current tag information is labeled as previous tag information and the robot moves forward until it finds a new tag and retrieves new tag information. The algorithm then checks if the current tag column value is equal to 3. If it is, target tag has been found. If not, the NAV microcontroller goes into a loop of finding new tag information until the current tag column value is equal to 3 and the target tag is found. Figure 4-7 shows the Facing North flowchart.

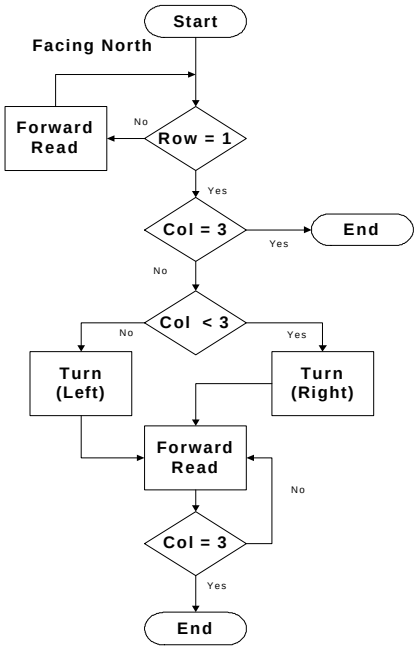


Figure 4-7 Facing North Flowchart

4.3.4.5 Facing South

When the algorithm has determined that the robot is facing South, it checks if the current tag column value is less than 3. If it is, the robot makes a left turn. If it isn't, the robot makes a right turn. The current tag information is now labeled as previous tag information. At this point, the robot is either facing East or West so the algorithm proceeds to the Facing East sub-routine. Figure 4-8 shows the Facing South flowchart.

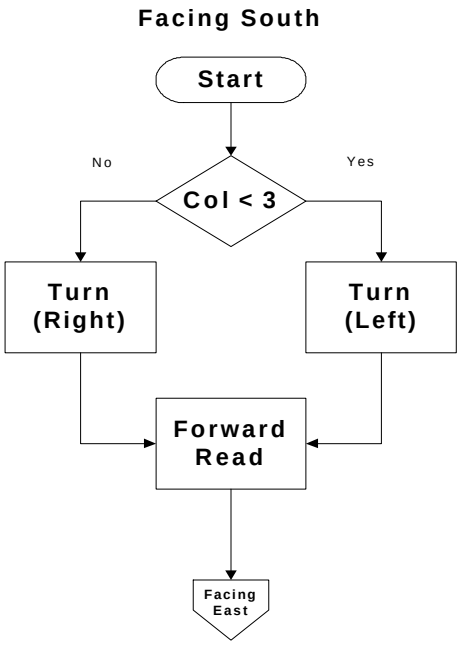


Figure 4-8 Facing South Flowchart

4.3.4.6 Facing East

When the algorithm finds itself in the Facing East sub-routine, it checks if the current tag column value is less than 3. If it's not less than 3, the algorithm proceeds to the Facing West sub-routine. If it is, it checks if the current tag column value is equal to 3. If it isn't, the algorithm will go into a loop and the robot will move forward and to find new tag information until the column value is equal to 3. Once the column value is equal to 3, the algorithm checks if the current tag row value is equal to 1. If it is, the target tag is found. If not, the algorithm checks if the current tag column value is less than 3. If it is, then the robot makes a left turn. If the column value is greater than 3, a right turn is made. The algorithm then goes into a loop and moves forward and obtains new tag information until the current tag row value is equal to 1. At this point, the target tag has been found. Figure 4-9 shows the Facing East flowchart.

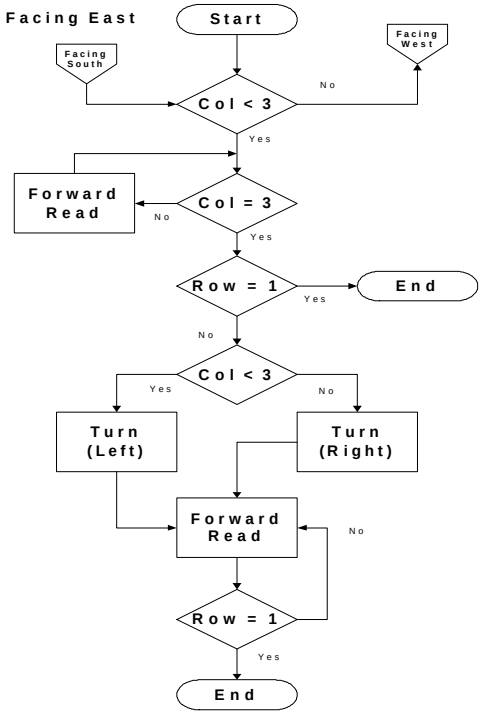


Figure 4-9 Facing East Flowchart

4.3.4.7 Facing West

When the algorithm is in the Facing West sub-routine, it performs a routine similar to the Facing East routine, but with one difference. After the robot makes its way to Column 3 and checks if the current tag column value is less than 3, if this condition is true (ctagcol < 3), the robot makes a right turn. If it is false (catgcol > 3), the robot makes a left turn. Figure 4-10 shows the Facing West flowchart.

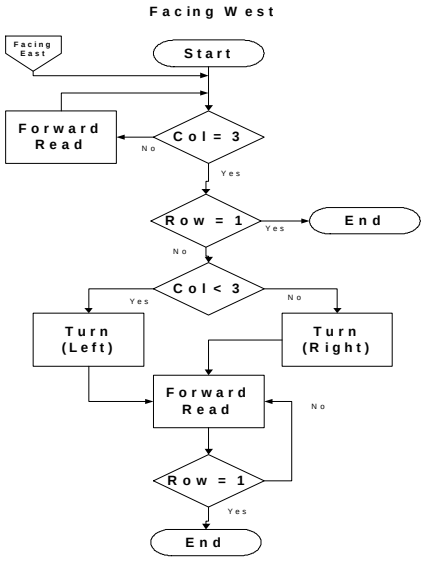


Figure 4-10 Facing West Flowchart

End Results

Some problems were encountered in developing the navigation solution for this project. One problem encountered was the difficulty in trouble shooting the original navigation source code. Source codes as complicated and as involved as this one can become a very tedious debugging job. Following the logic can be so meticulous and a lot of attention to detail must be taken. Focusing too much on the logic can take one's eye away from focusing on the syntax of the code. This was a major problem for this project. A lot of time was devoted to troubleshooting and debugging the syntax of the original navigation code. A solution to this problem is to break up a programming project into smaller, manageable parts. Coding debugging smaller building blocks where the whole routine fits on the screen makes it easier to catch syntax and logic errors and saves precious time that can be devoted to other tasks.

Another problem that was encountered in developing the nav code occurred due to the fact the original nav code was written in Visual C++. An incorrect assumption was made that the code could be written and debugged in Visual C++. Once the code compiled and was working, the assumption was it could simply be cut and pasted into PICC Lite and the code would then be ready to be placed into the PIC 16F628's Flash memory. Once the final version of the nav code was working and ready, cutting and pasting it over to PICC Lite and burning it into the microcontroller wasn't as easy as that. There were commands in PICC Lite where the code would use one of the two I/O ports of the microcontroller. There were I/O commands in PICC Lite that were outside the scope of the C Language. These I/O commands resembled assembly language commands, which was the language the microcontroller was programmed in. A possible solution to this problem is writing the nav code in assembly instead of C. This would simplify the development process because there wouldn't be two programming languages that needed to be considered.

4.4 Robot

4.4.1 Design

Two different designs were attempted to arrive at a working motor controller and motor driver solution. The first one used a programmable Gate Array Logic (GAL) device as motor controller and a relay-based circuit to act as motor driver. This first design proved to have some faults that were improved by a second attempt. The second design used a microcontroller approach instead as motor controller and an integrated dual motor driver. This second effort put us a lot closer to a reliable robot motor control module. Later on this paper, we explore the implementation, advantages, and shortcomings of each design.

4.4.2 Equipment Requirements

The following is a list of hardware and software resources that were needed to design, build, and test the robot:

4.4.2.1 Project Concept Research

- Books, Internet
- Examples and approaches used by other people
- Specifications, application notes on candidate components

4.4.2.2 Software Tools

- Lattice's ispLEVER 3.1 Programmable Logic Development Environment
- Microchip's MPLAB Integrated Development Environment for PIC microcontrollers

4.4.2.3 Prototype Construction

- EETools' TopMax Universal Device Programmer
- Microchip's PICSTART Plus microcontroller programmer

4.4.2.4 Testing, Debugging, and Fine Tuning

- Digital Multi-Meter
- Digitizing Oscilloscope
- Bread-board Prototyping Board
- Microchip's PICSTART Plus Debugger

4.4.3 Materials and Capital Required Per Unit Module

The table below presents the material and capital required to construct the robot.

Part Description	Quantity	Cost
Chassis	1	$15
Motors + Gearbox	1	$10
Cordless Phone Batteries	1 pack	$10
AA Batteries	3	$4.90
	Total	$40

Table 4-6 Robot Bill Of Materials

The following table presents the material and capital required per unit for the first motor controller prototype.

Description	Quantity	Cost
Gate Array Logic	1	$1
Electromechanical Switches	4	$16
Transistors	4	$3
Prototype Board	1	$5
	Total	$25

Table 4-7 First Motor Controller BOM

The table below the material and capital required per unit for the final motor controller design.

Description	Quantity	Cost
Microcontroller	1	$3.95
Dual Motor Controller	1	$20
Prototype Board	1	$5
	Total	$28.95

Table 4-8 Final Motor Controller BOM

4.4.4 Process of Implementation

The robot module can be easily explained by breaking it into hardware and software. As illustrated in figure 4-11, the robot interfaces with the navigation module waiting for an instruction to execute.

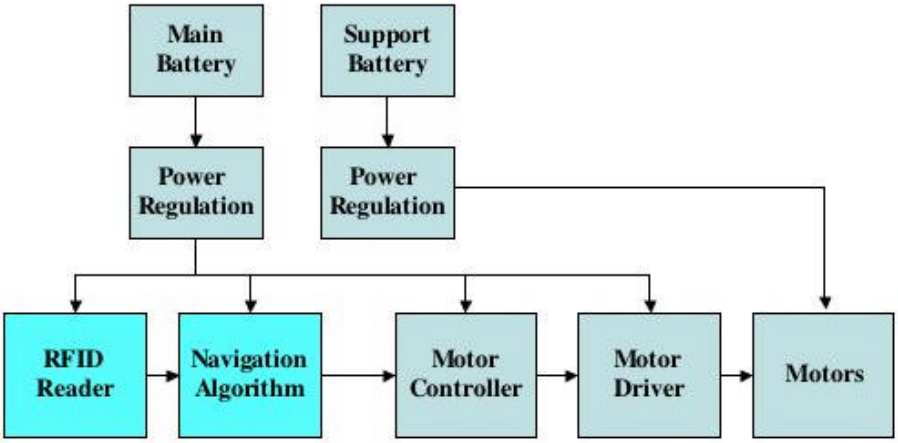


Figure 4-11 Hardware Block Diagram

As part of the hardware to control the robot, we used two motors controlled by a motor driver. The motor driver follows instructions send by the motor controller. The output of the motor controller is based on information received from the Navigation Algorithm Module. Logic and motors are power by separate batteries.

4.4.4.1 First Design

To implement the hardware and software of the robot, two prototypes were designed and constructed. The figure below presents the block diagram of the first design.

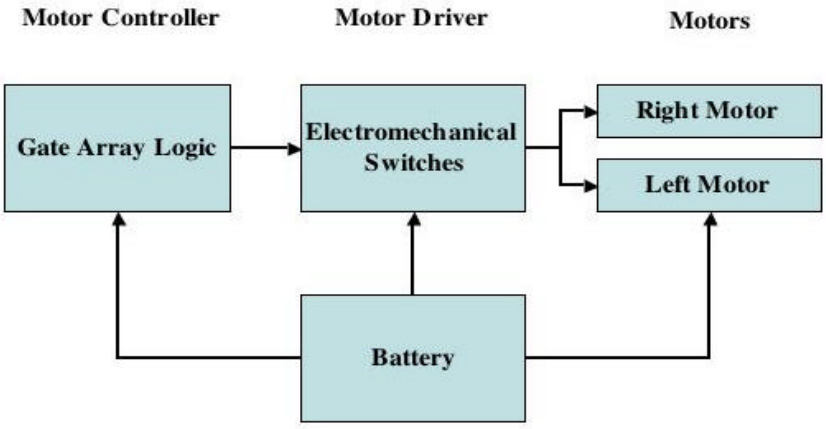


Figure 4-12 First Design Hardware Block Diagram

In this design, electromechanical switches, which were controlled by a programmable Gate Array Logic (GAL) chip, drove the motors. The entire system was powered by only one battery.

4.4.4.2 First Design Software Implementation

The motor controller was implemented using logic equation in the GAL. Four outputs from the GAL switched transistors that controlled the relays. These last ones, at the same time, would direct current flow through the motors. The following figures present the truth table, K-maps, and equations used for this design.

	INPUT			OUTPUT			
Instruction	A	B	C	MR+	MR-	ML+	ML-
Turn Left	0	0	0	0	1	1	0
Reverse	0	0	1	1	0	1	0
Turn Right	0	1	0	1	0	0	1
Forward	0	1	1	0	1	0	1
Stop	1	0	0	0	0	0	0
Reserved	1	0	1	0	0	0	0
Reserved	1	1	0	0	0	0	0
Reserved	1	1	1	0	0	0	0

Figure 4-13 First Design SW Truth Table

Right motor+ logic K-map:

$$M_R^+ = \overline{A}B\overline{C} + \overline{A}\overline{B}C$$

C\AB	00	01	11	10
0	0	1	0	0
1	1	0	0	0

Figure 4-14 Right Motor+ K-Map

Right motor- logic K-map:

$$M_R^- = \overline{A}\overline{B}\overline{C} + \overline{A}BC$$

C\AB	00	01	11	10
0	1	0	0	0
1	0	1	0	0

Figure 4-15 Right Motor- K-Map

Left motor+ logic K-map:

$$M_L^+ = \overline{A}\overline{B}$$

C\AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Figure 4-16 Left Motor+ K-Map

Left motor- logic K-map:

$$M_L^- = \overline{A}B$$

C\AB	00	01	11	10
0	0	1	0	0
1	0	1	0	0

Figure 4-17 Left Motor- K-Map

Logic equations are converted into a fuse map using a programmable logic development environment provided for free to students. Later on, this fuse map can be "burned" to a PAL or GAL using a universal device programmer like EETools' TopMax. Fig 4-18 illustrates the environment used to write the GAL program.

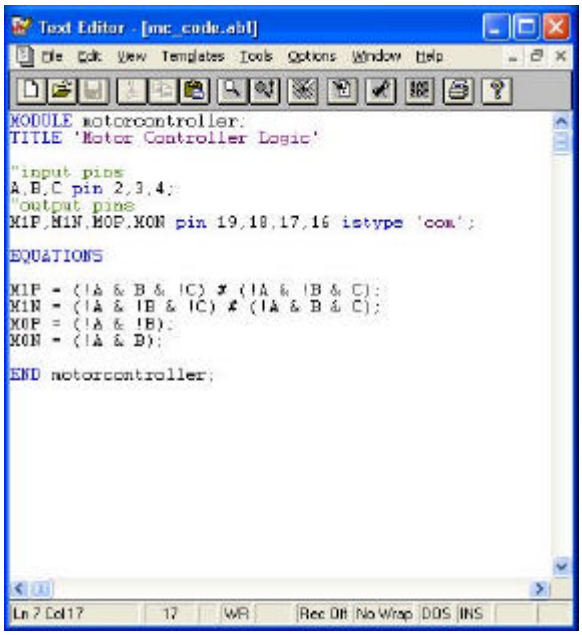


Figure 4-18 GAL Program Environment

Figure 4-19 illustrates a waveform simulation generated by the equations. This corresponds to the intended motor current flow behavior to independently activate each motor in either direction of motion (forward or reverse).

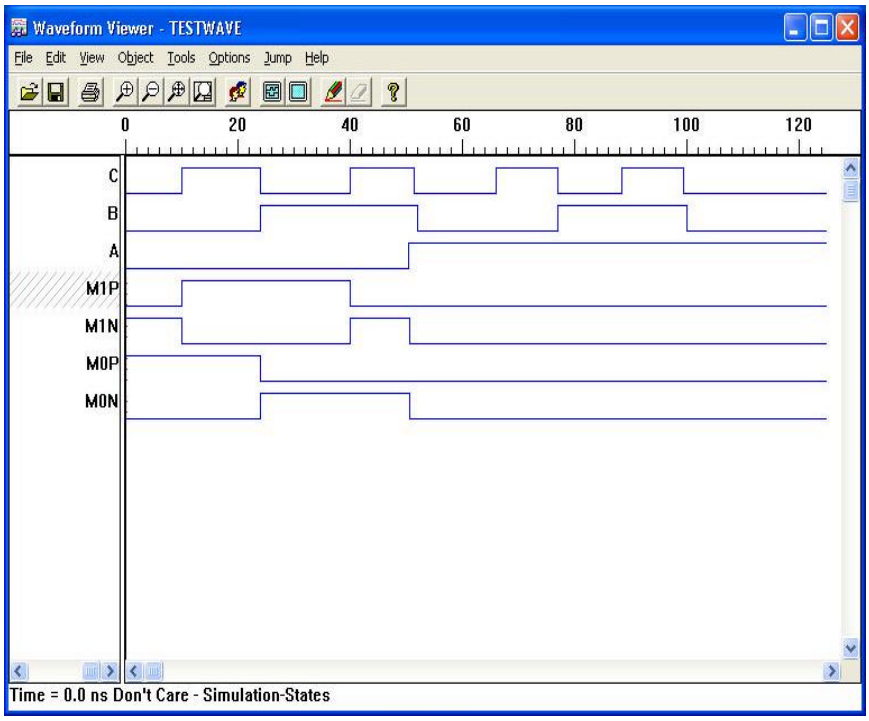


Figure 4-19 Waveform Simulation

4.4.4.3 First Design Hardware Implementation

Signals send by the motor controller cannot drive the motors directly without some conditioning. The current required to drive the relays would probably damage the programmable logic device. Instead, transistors are used in between each relay and its corresponding GAL output. Each motor is controlled by two relays. Each relay can be independently activated causing current flow in any direction through each motor. The direction of current flow determines the direction of motor turn. Figure 4-20 presents the circuitry design based on the electromechanical switches for one motor.

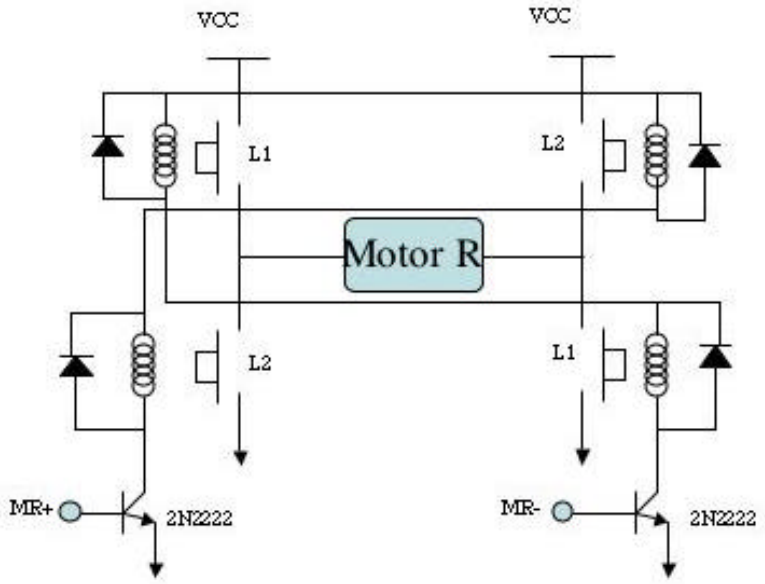


Figure 4-20 Relay Circuit for Right Motor

4.4.4.4 First Design Problems

Several problems were encountered after the implementation of this first design. The first problem was that one battery did not provide enough power to logic when the motor driver activated the motors. This was due to a drop in voltage caused by an increase in current demanded by the motors. Also, programmable logic is not very flexible and cannot easily provide timing control. Since the output to the motors was present immediately and for the same length of time an input was received. Finally, this design resulted on an open-loop motor control methodology, since it did not provide any kind of feedback.

4.4.4.5 Final Design

To solve some of the problems posted by the first design, a second approach, based on a different design philosophy was attempted.

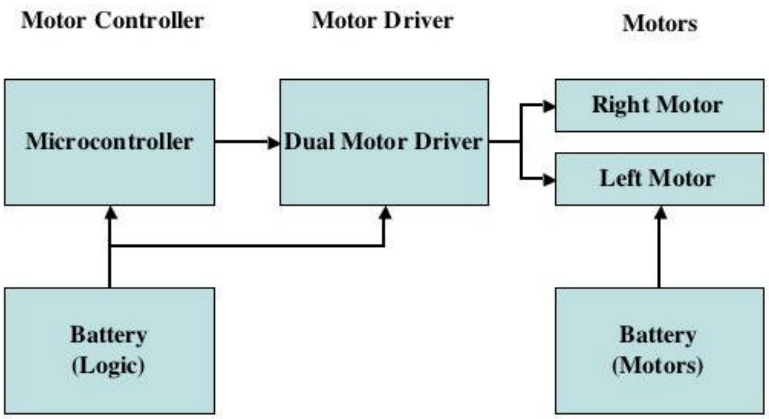


Figure 4-21 Final Design Hardware Design

As shown in figure 4-21, the final design is powered by two different battery sources. The motors are driven by a dual motor driver, and the motor controller was implemented using a microcontroller.

4.4.4.6 Final Design Software Implementation

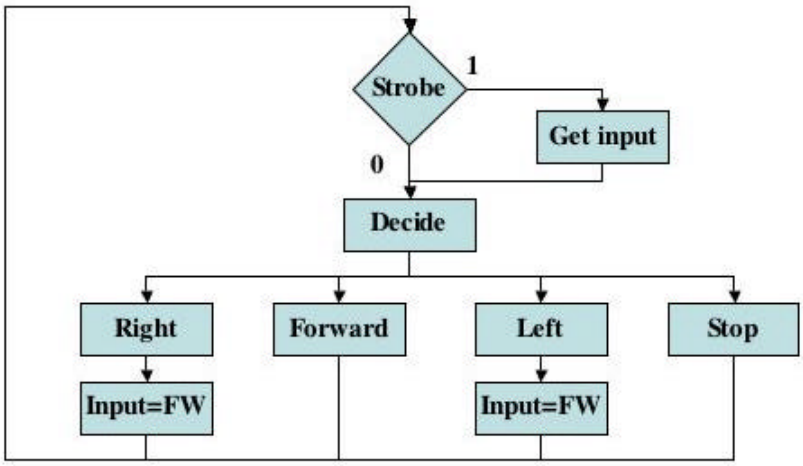


Figure 4-22 Process Flow Chart

Figure 4-22 presents the process flow chart of the microcontroller, which was programmed to wait for an input and send the appropriate command to the motor driver. This time the microcontroller regulates the length of movement to achieve automated turns. The navigation module will only have to issue a turn command and the motor controller will execute it correctly. All software was written in assembly language using the MPLAB development environment for PIC microcontrollers, and it was programmed into the built-in flash of the microcontroller using a PICSTAR plus programmer. The code programmed into the microcontroller can be found in Appendix 1.

4.4.4.7 Final Design Hardware Implementation

The microcontroller communicates with the motor driver through a serial interface. In addition, the motor driver has the required integrated power transistors to safely turn on and off each motor. Since this is a dual-motor driver, it can control both motors and their speeds. The motor driver and microcontroller specifications can be found in Appendix 5 and 4 respectively.

In this design, logic is powered by three AA batteries (4.5V) and the motors are powered by a cordless phone battery pack, which supply 3.6V and can drive 650mA.

4.4.4.8 Final Design Problems

The problem encounter in this design was to accurately calibrate turns to achieve exactly 90 degrees, which was the intended turning angle. The robot presents an average turn error of 2.7 degrees. Turns were calibrated using a software delay.

One solution to accurately calibrate robot turns will be to use shaft encoders and replace the software delay function by a shaft encoder step counter. This would increase the accuracy of the turn by removing some of the error caused by the motor reacting to an "instantaneous" lack, or presence, of current when the motor is turned off, or on, respectively. Furthermore, the calibration of turns depends on the surface on which the robot navigates

4.4.5 Process of Simulation

4.4.5.1 RFID Reader Simulator

This simulator tests the motor controller without needing to read RFID tags or having a navigation algorithm. A second PIC microcontroller was programmed to simulate reader commands. Once this microcontroller is connected to the motor controller, the robot executes a predefined list of movements in a time interval dictated by the microcontroller. The navigation algorithm is bypassed.
The code programmed in this second microcontroller can be found in Appendix 2.

4.4.5.2 Navigation Algorithm Simulation

This simulator tests the motor controller and RFID reader without needing a navigation algorithm. Another PIC microcontroller was programmed with a simple navigation algorithm. This microcontroller is connected to the RFID reader module and the motor controller. Therefore, the robot is capable of extracting the RFID tag address, matching it to an entry on a predefined movement table, and executing a movement.
The code programmed into this microcontroller can be found in Appendix 3.

4.4.6 End Results

Despite the problems encounter with the first design, the robot module was successfully completed using a second design approach. The RFID Reader Simulator showed that the robot accepts instructions and moves correctly. Since the RFID tags are placed close to each other on the grid design, the 2.7-degree average turn angle error does not represent a tremendous deviation.

5 Conclusion

Completion of this project brings a new product to the world of industry to increase speed and efficiency while reducing the loss. To implement the idea, project was divided into four different parts: RFID system and antenna, RFID system communication, Navigation algorithm, and Robot. Each part was handled by a member of a group.

In developing this project, new and innovative solutions were needed to tackle the design challenges that were encountered. Each problem was dealt with further research and trial and error method in a timely manner. Various objective of electrical engineering was used in developing this project including radio frequency, digital design, and signal processing. Overall the learning objective of this project provided an opportunity to research beyond the academic requirements.

Total estimated cost of this project was approximately $364, which was below the estimated $400 budget.

6 Appendixes

6.1 Appendix 1 - Motor Controller Code

```
;*************************************************************
; Processor: PIC16F628 at 4 MHz using internal RC oscillator
; Function:  Motor Controller
; Hardware:  Onboard Microcontroller
; Filename:  MC.asm
; Author:    Betancourt, Ivette

;*************************************************************
```

6.2 Appendix 2 - RFID Reader Simulator

```
;*************************************************************
; Processor: PIC16F628 at 4 MHz using internal RC oscillator
; Function:  Give Navigation Commands
; Hardware:  Onboard Testboard
; Filename:  MC7-2.asm
; Author:    Betancourt, Ivette

;*************************************************************
```

6.3 Appendix 3 - Navigation Algorithm Simulation

```
;*************************************************************
; Processor: PIC16F628 at 4 MHz using internal RC oscillator
; Function:  Give Navigation Commands
; Hardware:  Onboard Testboard
; Filename:  MC7-2.asm
; Author:    Betancourt, Ivette

;*************************************************************
```

6.4 Appendix 4 – Microchip PIC6F628 Data Sheet

6.5 Appendix 5 - Dual Motor Controller User Guide

6.6 Appendix 6 Initial Navigation Algorithm

/* This program will direct the robot to find a path to the target tag.

Written by: Nathaniel Angat

*/

6.7 Appendix 7 Final Navigation Algorithm

/* SIMPNAVFINAL

This program will tell the robot how to navigate in the RFID grid.

Written by: Nathaniel Angat
Assumptions:
1) Tagcol and tagrow values are constantly being updated by the first microcontroller.