ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

Şeyh Şamil ASLAN

COMPUTER CONTROLLED ROBOT CAR

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA-2006

ÇUKUROVA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

COMPUTER CONTROLLED ROBOT CAR

Şeyh Şamil ASLAN

YÜKSEK LİSANS TEZİ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 25/12/2006 Tarihinde Aşağıdaki Jüri Üyeleri Tarafından Oybirliği/~~Oyçokluğu~~ ile Kabul Edilmiştir.

İmza: ……..	İmza: …….	İmza: ……
Yrd.Doç.Dr. Turgay İBRİKÇİ DANIŞMAN	Prof.Dr. Süleyman GÜNGÖR ÜYE	Yrd.Doç.Dr. Murat AKSOY ÜYE

Bu tez Enstitümüz Elektrik-Elektronik Mühendisliği Anabilim Dalında hazırlanmıştır.

Kod No:

Prof.Dr. Aziz ERTUNÇ
Enstitü Müdürü

ABSTRACT

MSc THESIS

COMPUTER CONTROLLED ROBOT CAR

Şeyh Şamil ASLAN

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
INSTITUTE OF NATURAL AND APPLIED SCIENCES
UNIVERSITY OF CUKUROVA

Supervisor : Asst. Prof. Dr. Turgay İBRİKÇİ
Year : 2006 Pages: 93
Jury : Asst. Prof. Dr. Turgay İBRİKÇİ
Prof. Dr. Süleyman GÜNGÖR
Asst. Prof. Dr. Murat AKSOY

Usage of microcontroller has been grown up extremely in electronics and computer fields. Nowadays microcontroller based wireless control applications are begun to use in many applications. It is so important to control a system from central point, to prevent wasting time, a place where dangerous for human, a place where is not reachable, a place where it is so hard to work. The main aim of this study is both to control a remote device by establishing a serial communication between microcontroller and computer wirelessly and to get image and temperature data related to location where device is.

A model car has been chosen as a device (Robot) to be controlled. Three motors have been selected on robot car for forward-reverse motion, change direction (left-right) and 360° rotation of camera mounted on the top of the robot car. Control card on the robot car manages the related motors in accordance with signal coming from the computer and IR sensor. DC motor selected for forward-reverse motion of robot car can work in two directions and four different speeds. The different sequential speeds are executed by PWM (Pulse Width Modulation) signal. If the robot car face with an obstacle, it has an ability to stop without crashing to that obstacle. In addition to this, when the robot car enters into a dark area such as tunnel, far lamps are switched on. A couple of RF modem module has been used in order to provide wireless communication between computer and robot car. A user interface has been established on computer to send commands to robot and images received from wireless camera can be viewed on the computer screen simultaneously

Keywords: Robot Car, Microcontroller, Control card, RF Modem, User Interface

ÖZ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR KONTROLLÜ ROBOT ARAÇ

Şeyh Şamil ASLAN

ÇUKUROVA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Yard. Doç. Dr. Turgay İBRİKÇİ
Yıl : 2006 Sayfa: 93
Jüri : Yard. Doç. Dr. Turgay İBRİKÇİ
Prof. Dr. Süleyman GÜNGÖR
Yard. Doç. Dr. Murat AKSOY

Elektronik ve bilgisayar alanlarında mikrodenetleyici kullanımı son derece yaygınlaşmıştır. Bugünlerde ise mikrodenetleyici tabanlı kablosuz kontrol uygulamaları bir çok uygulama alanında kullanılmaya başlamıştır. Uygulamada zaman kaybını önlemek, insan hayatı için tehlikeli, ulaşılması ve çalışılması zor olan noktaları bir merkezden kontrol etmek çok önemlidir. Bu çalışmada esas amaç bilgisayar ve mikrodenetleyici kullanarak seri bir haberleşme sistemiyle uzaktaki bir aygıtı kablosuz olarak kontrol edebilmek ve aygıtın bulunduğu ortamın görüntüsü ve sıcaklığı gibi verileri de kablosuz olarak almaktır.

Kontrol edilecek aygıt (Robot) olarak bir model araba kullanılmıştır. Robotun ileri-geri hareket etmesi, yön değiştirmesi (sol-sağ) ve robotun üzerine yerleştirilen kameranın 360 derece dönmesi için toplam 3 motor seçilmiştir. Robot üzerindeki denetim kartı, bilgisayardan ve IR sensörden aldığı verilere bağlı olarak ilgili motorların kontrolünü yapmaktadır. Robotun ileri-geri hareketini sağlayan DC motoru iki yönde ve 4 ayrı hızda çalışabilmektedir. Motorun farklı hızlarda çalışması darbe genişlik modülasyonlu (PWM) işaret ile sağlanmıştır. Robot fiziksel olarak bir engelle karşılaşırsa, engele çarpmadan durabilme yeteneğine sahiptir. Bunun yanında robot, tünel veya karanlık bir ortam girdiğinde farları yanmaktadır. Robot ile haberleşmek için RF modem modülü kullanılmıştır. Bilgisayardan robota komut göndermek için bir kullanıcı arayüzü kurulmuş olup aynı zamanda da robot üzerinde bulunan kablosuz kameradan gelen görüntü de ekrandan izlenebilmektedir.

Anathar Kelimeler: Robot Araba, Mikrodenetleyici, Denetim Kartı, RF Modem, Kullanıcı Arayüzü

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisor Asst. Prof. Dr. Turgay İBRİKÇİ whose help and patience, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

I would like to express my gratitude to my committee members Prof.Dr. Süleyman GÜNGÖR who is the Chairman of the Department of Electrical and Electronics Engineering and Asst. Prof. Dr. Murat AKSOY for their support and very valuable discussions on my study.

I would like to thank to TEKFEN Construction and Installation Co., Inc. Electrical Works Vice President Mr. Celal ERBİL for his support, tolerance and understanding during my study.

I would also like to thank to BOTAŞ Ceyhan Marine Terminal Operation Manager Mr. Süleyman Ersin ÖZEN and all my colleagues working in Baku-Tbilisi-Ceyhan Crude Oil Pipeline Project, Ceyhan Marine Terminal for their help, support and interest.

Finally, I would like to give my special thanks to my family one by one because their patient, encouragement and love enabled me to complete this study.

CONTENTS **PAGES**

NOTATIONS

µC : Microcontroller

RSSI : Received Signal Strength Inducator

PWM : Pulse Width Modulation

ADC : Analog Digital Converter

SRD : Short Range Device

PCB : Printed Circuit Board

LIST OF FIGURES **PAGES**

LIST OF TABLES **PAGES**

1. INTRODUCTION

Robots are needed for a variety of tasks and have wide applications in an ever-increasing number of fields including medicine, manufacturing, space and underwater exploration as well as safety and rescue operations. Examples are space or underwater exploration, moving in confined and restrained spaces such as narrow pipes and passageways as needed in earthquake rescue tasks, and moving objects that are too small or too big for humans to handle. Some tasks require performance beyond human capabilities such as a higher degree of repetitive precision, high-speed motion, or high levels of strength. In addition to this, there are some tasks which are dangerous for humans. Work in dangerous environments such as volcano craters, space and underwater exploration missions, chemical spill clean-up, nuclear waste disposal, explosive material manipulation, and tasks that require prolonged exposure to cold, heat, pressure, lack of air, or other conditions harmful to humans. (Manseur, 2006)

Mobile robots have many different uses in industry and become a very important branch of Robotics. They are viewed as an important improvement in automated transportation systems. It is likely that, once the technology is sufficiently advanced to ensure safe and reliable operation, automobiles, trucks, trains, airplanes, and possibly ships will be built with the ability to move autonomously. Recent developments and integration of various research areas such as the satellite-based Global Positioning System (GPS), wireless communication systems, sensor fusion techniques, intelligent highway systems, computer networking, computer vision, sensing, and machine cognition and intelligence systems will eventually combine to produce reliable autonomous transportation systems. (Manseur, 2006)

A robot can be viewed as a computer-controlled machine. As such it is the combination of a computer and a machine, which justifies the definition of robotics as an offspring of the industrial revolution and the information revolution. A robot can be broken into its main components in two subsets: hardware and software. The hardware can be further divided into mechanical and electrical or electronic and includes all the physical components such as motors, frame, gears, belts, sensors,

computers, wires, and cables. The software includes all the data stored in the memory chips, the programs that run the different components of the robot, and most importantly, the programs that embody the “artificial intelligence” of the robot. (Manseur, 2006)

This thesis presents an application of personal computer controlled mobile robot car that include a wireless camera mounted on the top of it. The electronic control hardware that serves as an interface between the mechanical hardware and motion-planning unit is realized. The motion-planning unit is a personal computer with a developed User Interface. It is provided that the mobile robot understands and executes the given instructions received from the personal computer. Image that is coming from the wireless camera on the top cover of robot car is being monitored via its receiver module and capture card.

2. PREVIOUS STUDIES

Imagine using your computer for more than just word processing and games. Just around the corner is the technology that will allow us to step outside of the computer and use it to control the outside world. We will be able to control everything from coffee makers to light switches while being miles away from either. We will incorporate this idea into the control of a remote control car.

The two major parts to the project is the computer software and the car hardware. The software consists of receiving input from the keyboard, decoding the input, updating the screen and outputting the data over the serial port to the transmitter. The graphical portion of the project consists of a picture of an object (the car) drawn to the screen that is used to represent the different movements of the car. For example, when the car has be instructed to move forward by pressing a specific key on the keyboard, the car on the screen will move forward. When the car has been instructed to turn right, the car on the screen turns right. The hardware transmits the data to the car, which then decodes the data through the microcontroller. The microcontroller will then output the data to its own port to control the light, horn and motor speeds. (Ward and Stoor, 1999)

The control software and hardware of a computer controlled mobile robot prototype that will realize the steps, which are necessary to achieve the goal procedure, is designed and implemented. Designed system is an in-door mobile robot. The electronic control hardware which serves as an interface between the mechanical hardware and motion planning unit is realized. The motion planning unit is a personal computer with a developed Graphic User Interface. It is provided that the mobile robot understands and executes the given instructions received from a personal computer. (Özen, Yıldız and UZUN, 2000)

Currently successful execution of many human-in-the-loop manipulation tasks directly depends on the operator's skill or a programmer's knowledge of the presumed environment in which the task will be performed. Computer mediation of human inputs can augment this process to permit easy and rapid incorporation of local sensory information to augment performance, provide variable performance

assist for output motions/forces, and hierarchical distribution of control and graceful degradation. Such mediated control has enormous potential to both reduce operator error and permit incorporation of greater autonomy into human/robot interaction.

We chose to examine two sets of tasks to help enhance a remote operator's performance: one involving precision in setting the forward velocity in the presence of variable loads/disturbances and the other improves safety by assisting the operator to avoid obstacles by carefully mediating the operator's joystick inputs. The overall goal is to endow a set of local reflexes that use local sensory information to override the user's input in order to enhance security, safety, and performance. In particular, we implement and evaluate the paradigm of mediated control for remotely driving a mobile robot system that will serve as our scaled inexpensive testbed. We discuss various aspects of the design and implementation of the Smart Car mobile platform. Beginning with the process of selection of the mechanical platform, we will discuss our motivation and reasoning for making several of the design choices necessary to create the testbed in the subsequent sections. In addition, we discuss the issues pertaining to implementation of two controllers. One for obstacle detection and the other for wheel RPM PID control. We present the motivation and detail of the actual implementation. We use commercial off the shelf (COTS) hardware and integrated the whole system. Validation and calibration is a very critical step in the whole process. While an ad-hoc solution approach would also suffice for general demonstration purposes, we are interested in using the Smart Car as a scaled testbed. In particular, we are also interested in obtaining quantitative data and hence we spent considerable time validating our system. We first calibrate the independent subsystems, which include the transmitter/receiver, sensors, and then consider the calibration of the controller's interaction with the entire system. A test setup with an external reference tachometer was created to validate open and closed loop response of the various controllers. Such mediated control has considerable significance and application in a wide range of applications ranging from "Smart Highway Systems" to semiautonomous exploratory rovers. (Gott, 2003)

Robots are one of the most widely used machinery in industry that their importance continues to increase with the development of microelectronics and micro mechanics. A machine should have tree components to be called as a robot: Sensors to check the environment, computational units to process the data taken from the sensors and motion units to transfer computed data to mechanical response. Although their limited usage “Mobile Robots” are a type of robots which continue to become widespread at hazardous environments or small areas where working could be hard for a human. (Yıldız and Uzun, 2005)

3. DESIGN OF THE MOBILE PLATFORM

Mobile platform, which means a vehicle selected big enough to be modified and installed new designed devices for this study has been purchased from a well known store in Turkey. Once that vehicle was including remote controller, control devices mounted in the car and rechargeable batteries etc. Remote controller unit and control device have been removed from the car to be able to achieve the study requirements that would be carried out controlling the car remotely from the Personal Computer. Just existing motors (moving and direction motor) and led for lighting are being used in this study. General view regarding the car is shown below.

Figure 3.1. General view of the car

3.1. System Overview

The robot will be controlled from a computer and the user can use the user interface to control the car motion by using different arrow keys on the keyboard. In order to design personal computer controlled robot car, a few subsystems were considered such as power supply, stepper motor driver, DC motor driver, Modem for wireless communication between robot car and computer.

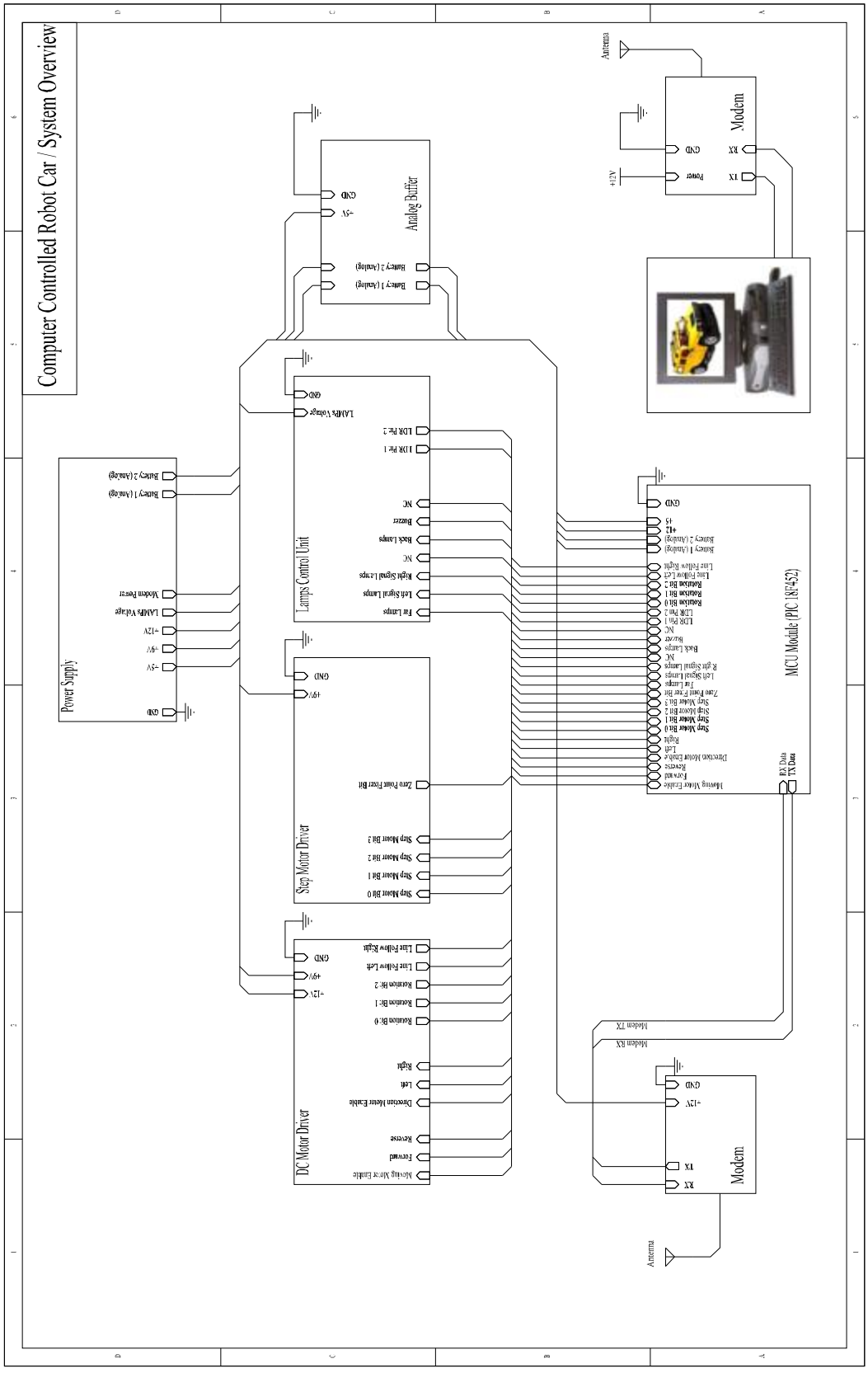


Figure 3.2. System schematic diagram

3.2. Power Supply

This power supply unit generates variety voltages such as 9V, 5V, 5.6V for Modem, Microcontroller, Stepper Motor, Lamps and Line follow device installed on the various section of the robot car via both 12V dry batteries mounted in the car. The fixed voltage power supply is useful in applications where an adjustable output is not required. Most digital logic circuits and processors need a 5V regulated power supply. Regulator IC as 7805 and 7809 have been used in this study to get a regulated and very stable +5V and +9V output voltage.

The L7805 and 7809 are simple to use for logic applications. You simply connect the positive lead of your unregulated DC power supply (anything from 9VDC to 24VDC) to the Input pin, connect the negative lead to the common pin and then when you turn on the power, you get a 5 volt supply from the output pin. Sometimes the input supply line may be noisy. To help smooth out this noise and get a better 5 volt output, a capacitor is usually added to the circuit. In order to get 9 volt DC power, same application above mentioned for 5V DC power supply should be applied by using 7809 IC. (STMicroelectronics 1, 2006)

3.2.1. Implementation of Power Supply Circuit

The Power Supply unit, which feeds other devices on the robot car, is created in Figure 3.3. As it seen in Figure 3.3. various voltages such as 12V, 9V, 5V etc. have been obtained via both 12V dry batteries. Using L7805 and 7809 IC regulators provided some output voltages that are needed for Microcontroller, Modem, Stepper motor and Lamps control unit. The output voltages named Battery 1 and 2 divided by resistors are directly connected to Analog Buffers and then Microcontroller to be seen battery voltage on User interface.

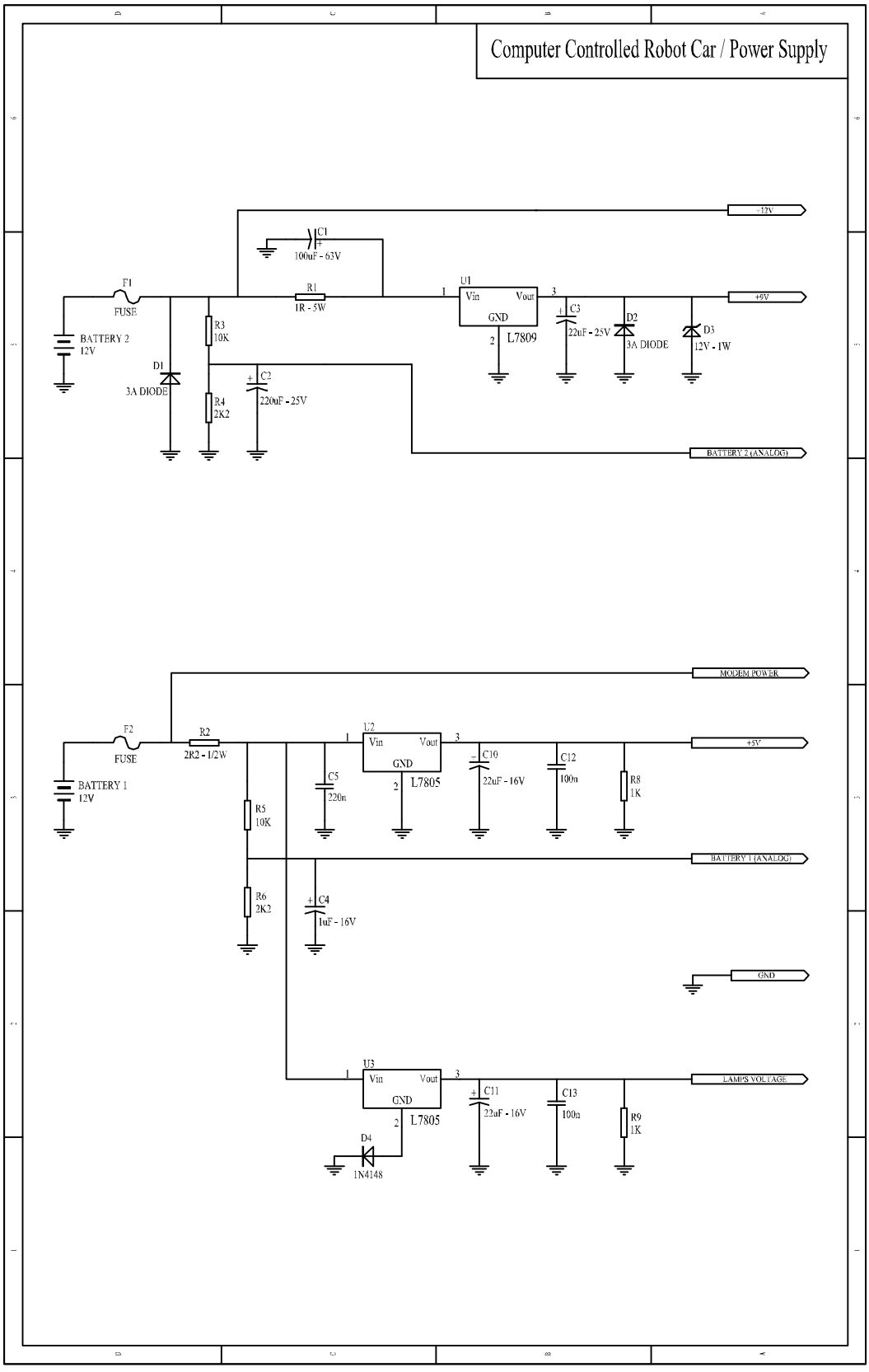


Figure 3.3. Power Supply

3.3. Stepper Motor Driver

Stepper Motor has been used to rotate the wireless camera 45° by 45°on the top of the car. A small box has been mounted on the stepper motor pin for creating a place for installation of the wireless camera and camera's battery and then camera mounted in that box is available to be rotated whenever user wants. When the power is applied to camera, it sends the images to computer wirelessly as a real-time. If user in front of computer wants to examine the images where the robot car is, he should push Q and W keys on keyboard to control direction of camera. Explanations regarding wireless camera can be found in Section 5. ULN2003 IC is used in this study in order to control stepper motor by signals coming from the main MCU unit.

3.3.1. Implementation of Stepper Motor Circuit

When MCU PIC 18F452 send positive voltage (+5V) to input pins of ULN2003A, outputs related to the inputs will be grounded after applying input voltage. As it shown in Figure 3.7, positive voltage has been applied to common cable of stepper motor so that it is being controlled by giving zero (0V) to another cables of that stepper motor. The following figures are showing what kind of stepper motor has been used in this study and how stepper motor works according to signal applied to coils numbered 1,2,3,4.

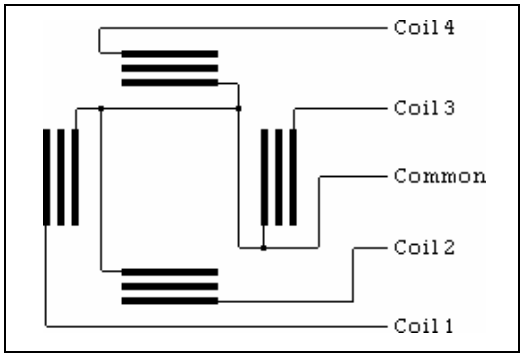


Figure 3.4. Stepper motor coil wiring
(www.doc.ic.ac.uk/~ih/doc/stepper/others, 1998)

The following figure explains that each successive coil is energized in turn.

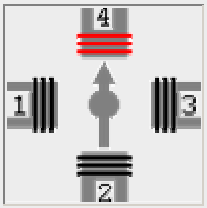
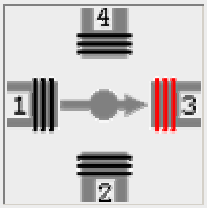
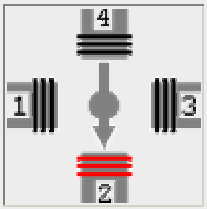
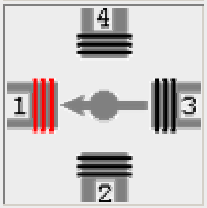
Coil 4	Coil 3	Coil 2	Coil 1	
on	off	off	off	
off	**on**	off	off	
off	off	**on**	off	
off	off	off	**on**	

Figure 3.5. Single-Coil excitation
(www.doc.ic.ac.uk/~ih/doc/stepper/control2/sequence.html, 1997)

The following figure explains that each successive pair of adjacent coils is energized in turn.

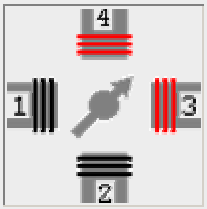
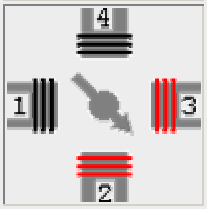
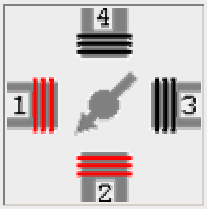
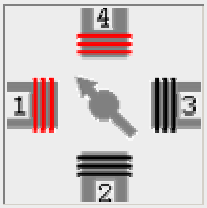
Coil 4	Coil 3	Coil 2	Coil 1	
on	**on**	off	off	
off	**on**	**on**	off	
off	off	**on**	**on**	
on	off	off	**on**	

Figure 3.6. Two-Coil excitation
(http://www.doc.ic.ac.uk/~ih/doc/stepper/control2/sequence.html, 1997)

When user in front of the computer wants to change the direction of the wireless camera on the robot car to have a look at the environment where robot car is, microcontroller send signals to the related pin of ULN2003A ICs as the user pushes Q and W button on keyboard. The stepper motor connected to ULN2003A will turn in accordance with the signals come from main MCU Unit. The following figure is created in order to rotate the camera 45° by 45° in a horizontal position.

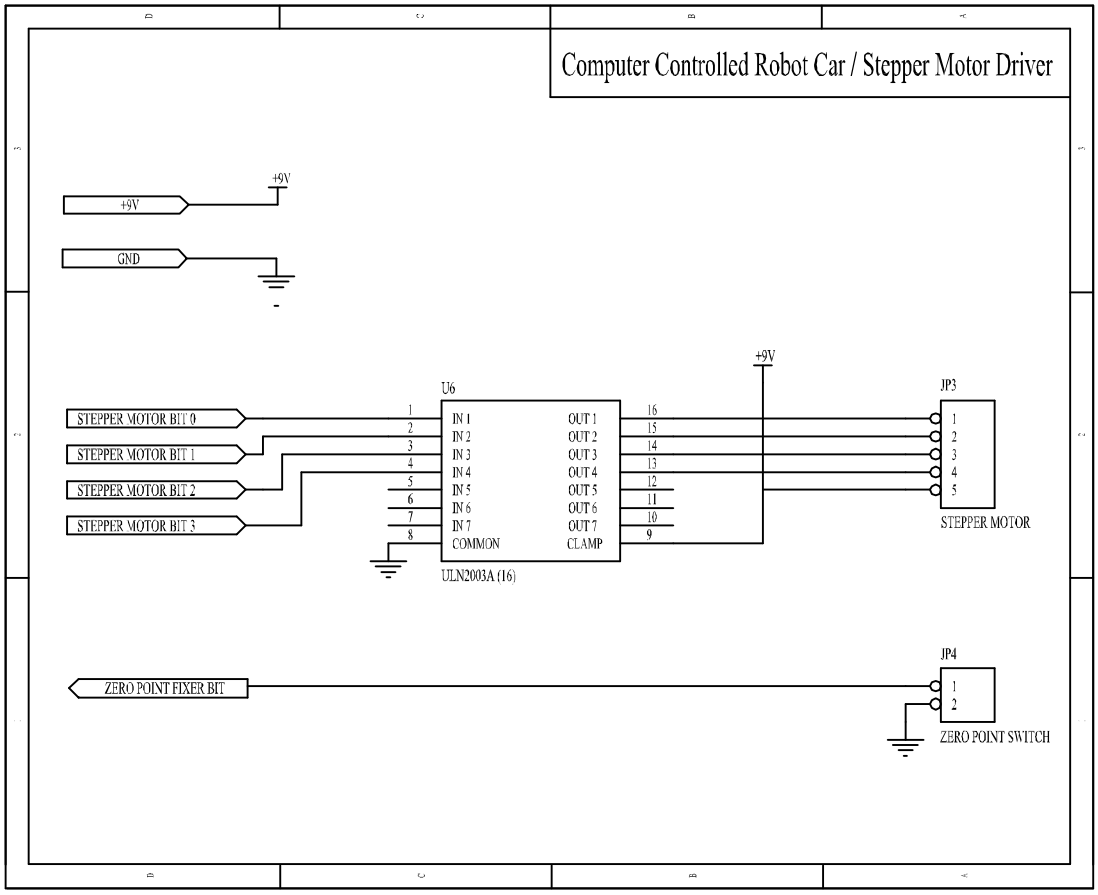


Figure 3.7. Stepper motor driver

3.4. DC Motor Driver

To carry out direction (Right and Left) and motion (Forward and Reverse) applications, 2 DC motors have been used in this study. These motors are original parts of the car. Direction and motion motors have been chosen 9V and 12 V by manufacturer of the car, respectively. Two L6203 ICs have been selected to be controlled direction and motion motors in accordance with the signal coming from the MCU Unit.

3.4.1. Implementation of DC Motor Drivers

Motion motor's speed can be adjustable in four steps by pushing arrow keys on keyboard and these 4 steps can be considered as a gear used to increase and

reduce the speed of the car as it is in real life. This increment and reduction of the car speed has been realized by PWM signal coming from MCU Unit.

Pulse Width Modulation is a technique to provide an output logic one for a period of time and a logic for the balance of the time. The PWM signal is still digital because at any given instant of time, the full DC supply is either on or off fully. The voltage or current source is supplied to analog load by means of a repeating series of on and off pulses. The on-time is the time during which the DC supply is applied to the load, and the off-time is the period during which that supply is switched off. (Peter, 1997)

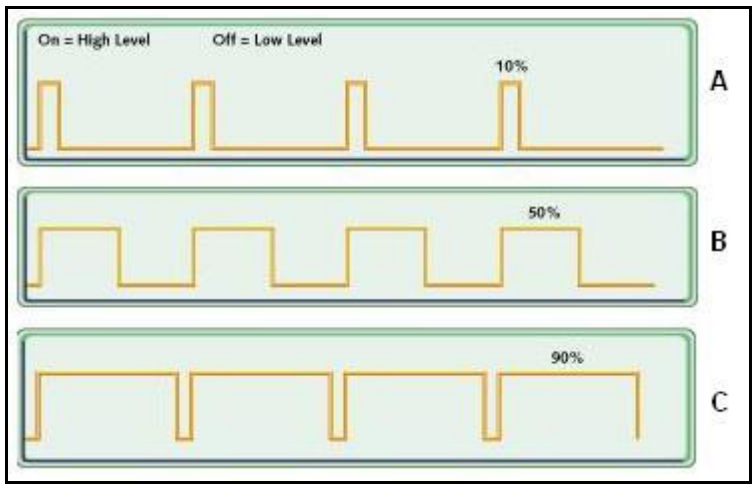


Figure 3.8. PWM Signal a) 10% duty cycle b) 50% duty cycle c) 90% duty cycle (Netrino Technical Library, 2001)

Figure 3.8 shows three different PWM signals. Figure 3.8.a shows a PWM output at a 10% duty cycle. That is, the signal is on for 10% of the period and off the other 90%. Figures 3.8.b and 3.8.c show PWM outputs at 50% and 90% duty cycles, respectively. These three PWM outputs encode three different analog signal values, at 10%, 50%, and 90% of the full strength. If, for example, the supply is 9 V and the duty cycle is 10%, a 0.9 V analog signal results. (Netrino Technical Library, 2001)

The following circuit diagram was created in order to drive both motors mounted in the car originally. When user wants to move the car in forward direction, forward arrow buttons on keyboard should be pushed. Another arrow buttons can be pushed how user has robot car moved. The aim of using ICs on circuit is to drive

motors in two directions (Forward and Reverse) according to the signals coming from the MCU unit. The motion motor can be moved in both directions in 4 speeds as gear by PWM signal. The mentioned about the PWM signal has been generated by MCU unit and then applied to L6203 dmos full bridge driver. Speed 1, 2, 3, 4 are %49, 69, 88, 99 duty cycle, respectively. While PWM signal is being sending to L6203 IC, enable pin on L6203 IC needs to be activated at the same time. The following figure is created in order to drive both motors, respectively.

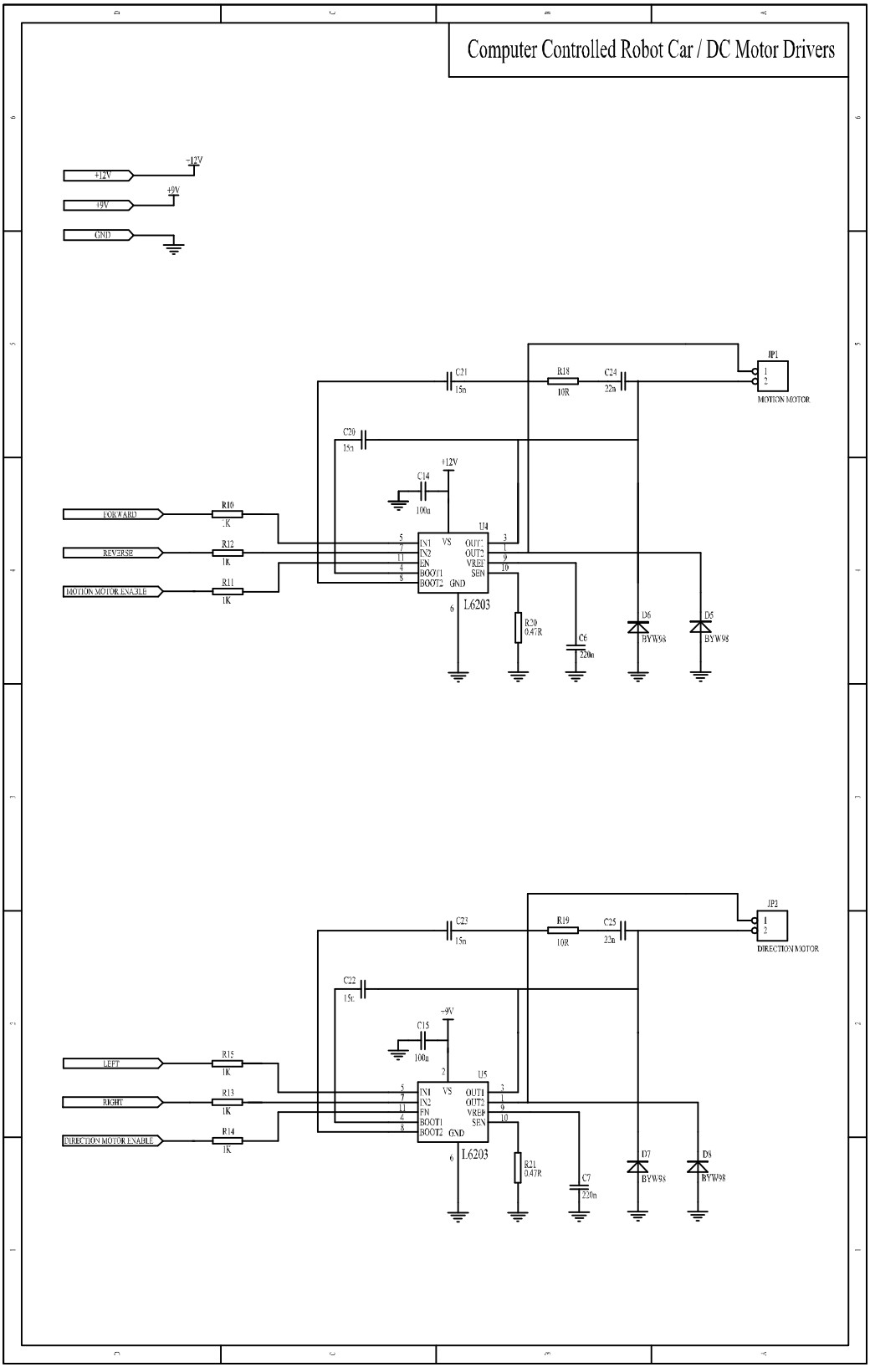


Figure 3.9. DC motor driver

3.5. MCU Unit

A microcontroller could be likened to the “brains” of the robot. It can be programmed by the designer to accomplish the task at hand. It is responsible for sending commands to other individual systems in the robot, receiving data from external devices, and coordinating all activities. (Ard and Skipper, 2004)

The microcontroller is necessary to comply with design requirements, as well as to unify all of the components of the system. The microcontroller plays the role of coordination in this study, as it must be able to input and output to all of the external components. PIC18F452 microcontroller manufactured by Microchip is selected to use for its ease of use, wide availability, wide range of features and recent technology. The following figure is showing the pin configuration of microcontroller used in this study.

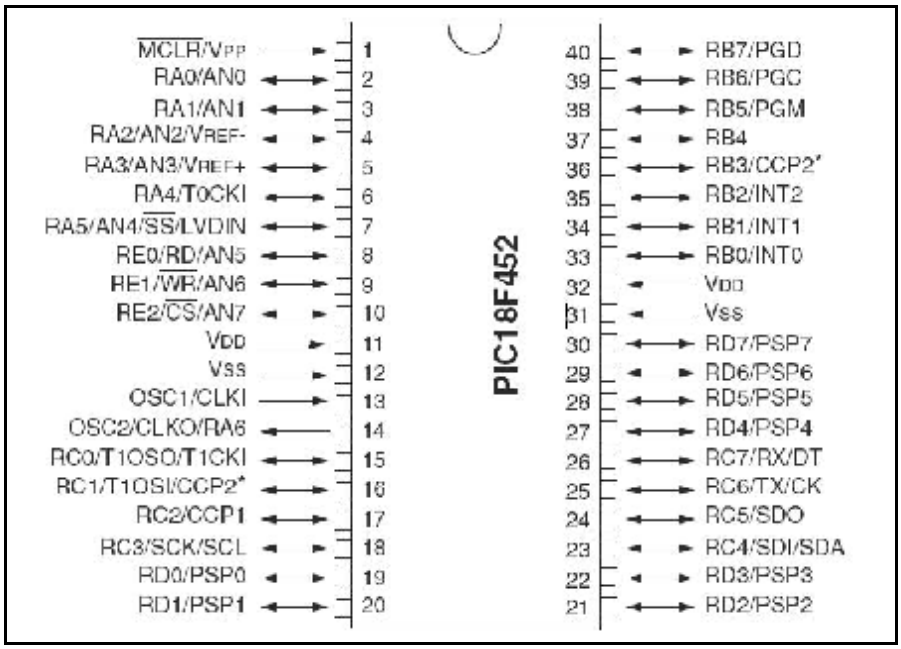


Figure 3.10. PIC 18F452 pin configuration
(Microchip Technology Inc. PIC 18FXX2 Datasheet, 2006)

The following table is showing the microcontrollers features to be compared with the others. As it seen in Table 3.1, the microcontroller PIC 18F452 has wide features for applications that need to be improved as requirements or contents of applications increase in future.

Table 3.1. PIC Microcontroller device features
(Microchip Technology Inc. PIC 18FXX2 Datasheet, 2006)

Features	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Operating Frequency	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz
Program Memory (Bytes)	16K	32K	16K	32K
Program Memory (Instructions)	8192	16384	8192	16384
Data Memory (Bytes)	768	1536	768	1536
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	17	17	18	18
I/O Ports	Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART
Parallel Communications	—	—	PSP	PSP
10-bit Analog-to-Digital Module	5 input channels	5 input channels	8 input channels	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

In this study, pin assignments for all ports have been defined as follows.
(Input:1 and Output:0)

- set_tris_a(0b11111111)
- set_tris_b(0b00000001)
- set_tris_c(0b00111010)
- set_tris_d(0b00000001)
- set_tris_e(0b11111000)

Microcontroller PortA is used for analog signals coming from both batteries and temperature sensor. RC0 and RC1 are used for serial communication with Modem. Other ports are digital input and outputs. The following circuit diagram was created in order to be proceed signal coming from the computer and managed the subsystems such as DC motor driver, lamps control unit etc.

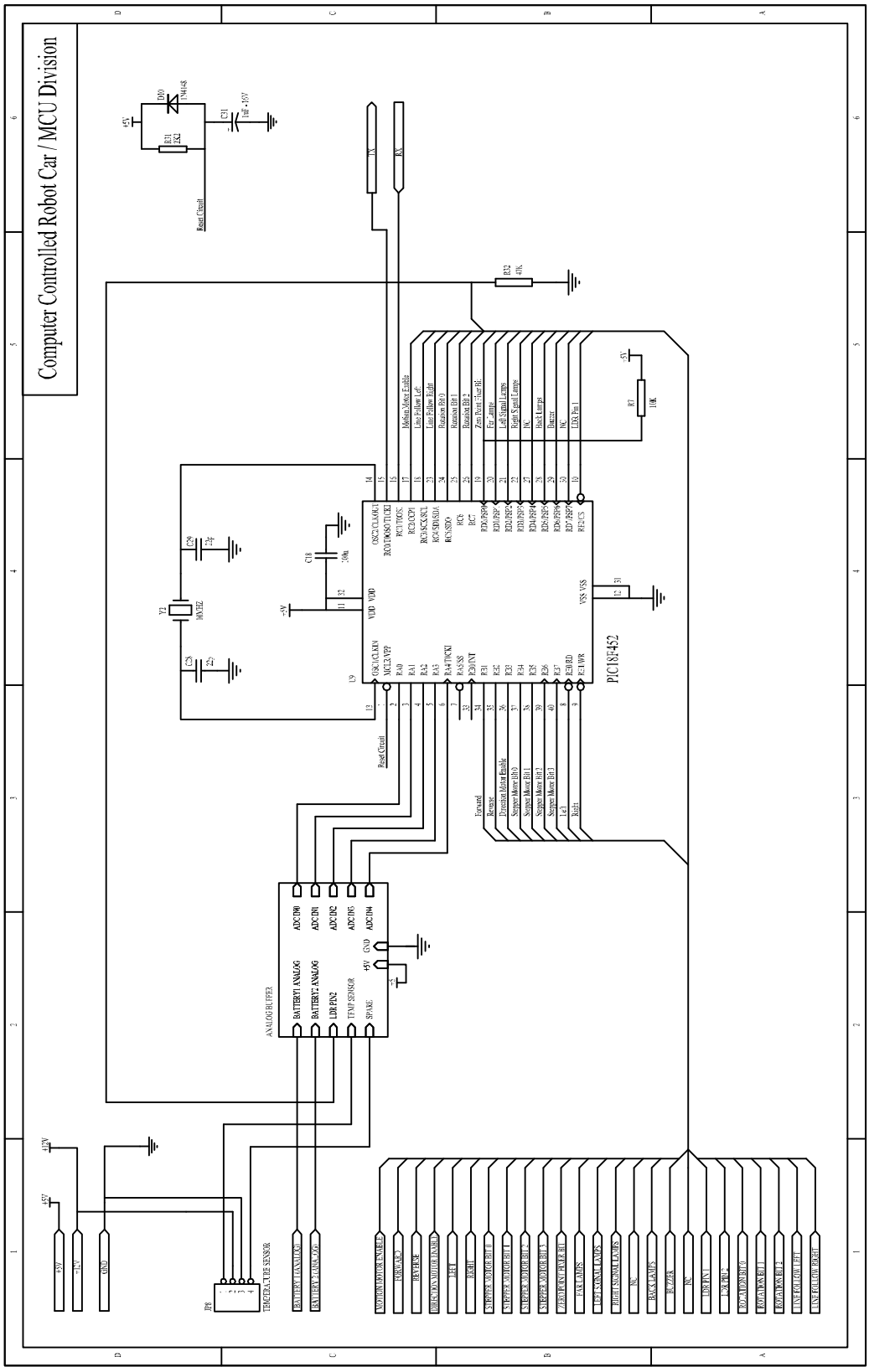


Figure 3.11. MCU Division

3.6. Analog Buffer

Two LM324N Low power quad operational amplifiers have been used in this study in order to carry two dry batteries, temperature sensor and LDR's signals to MCU Unit.

3.6.1. Implementation of Analog Buffer Circuit

Two LM324N Low power quad operational amplifiers are configured as a unity-gain voltage follower. The voltage follower is a very good buffer configuration. It is also a good way to test the chips. A simple way to establish the gain for this configuration is to note that the feedback from the voltage output to the negative input, often called the summing junction, wire or short circuit. Thus the summing junction is at the potential of the output voltage. Gain is the output/input. Thus since the input voltage equals the summing junction which equals the output voltage then the ratio of the output voltage, output/input = 1. The output follows the input (no sign inversion) and hence the term voltage follower. The input impedance is high since the current is very, very, low (assumed to be zero). Thus no loading occurs and it buffers the input, thus the name buffer amplifier. Using the characteristic of the non-inverting voltage follower of high impedance to limit the loading on the sensor, the output of the buffer stage should be a faithful transfer of the sensor voltage but with increased power available to drive the next stage.
(Jeong, 2006)

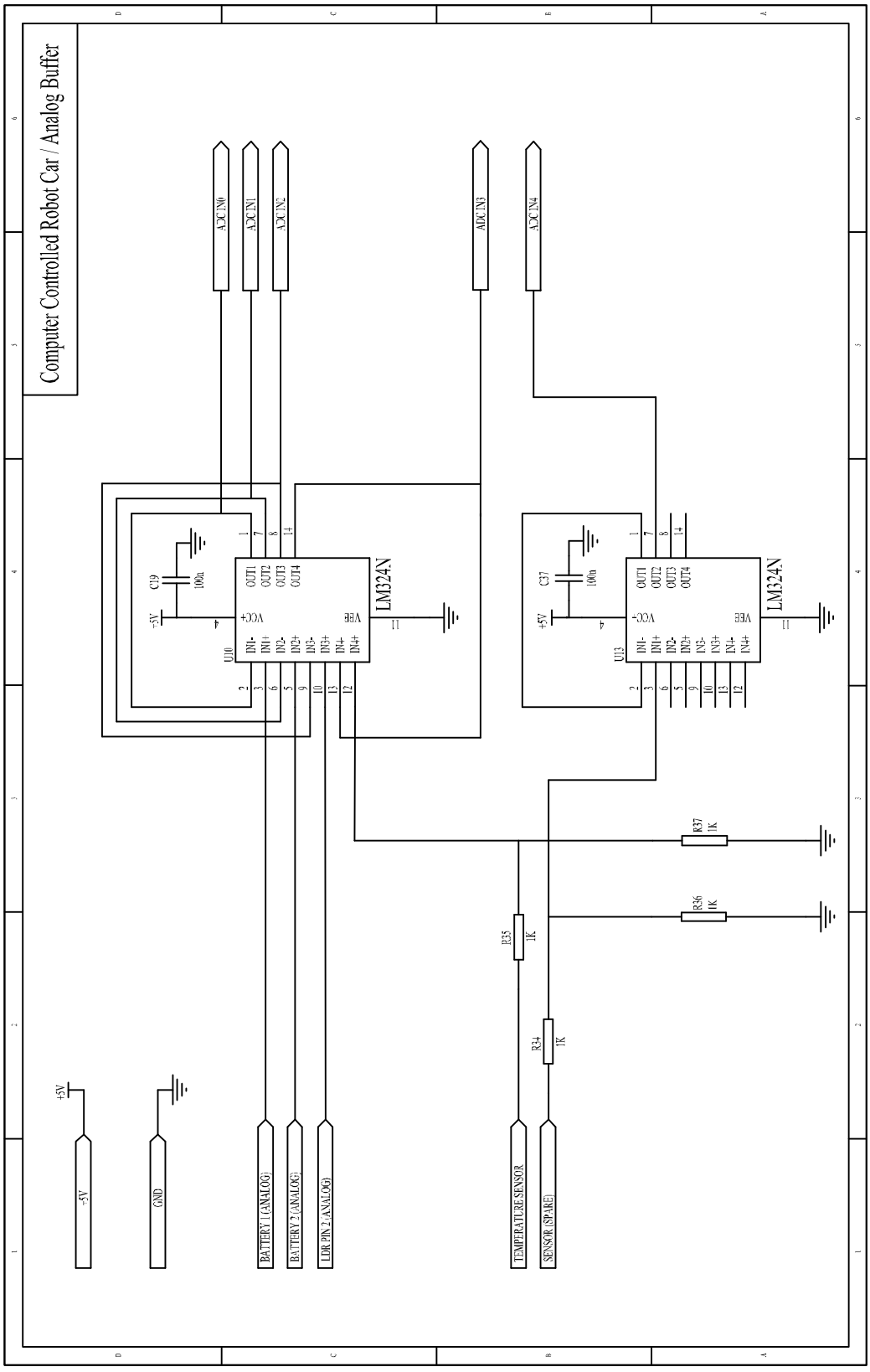


Figure 3.12. Analog Buffer

3.7. Lamps Control Unit

Eight lamps have been installed on the various point of cover of the robot car. Two lamps are used for fars, left signal lamps, right signal lamps and back lamps, respectively. Whenever robot car inside tunnel or dark area, far lamps will be activated for that reason LDR is used under the mobile platform. When robot car goes reverse, back lamps will be activated continuously, in addition to this, buzzer will be activated in an intermittent way as a warning sound. Whenever robot car turn left or right, related lamps will be activated like signal lamps on real car in real-life. MCU unit is sending signals to ULN2003A in order to execute the duties explained above related to the lamps and buzzer activation. As it seen in Figure 3.13, ULN 2003A has been used to transfer the signals coming from the MCU Unit to lamps and buzzer, respectively. As explained before, when MCU Unit send positive voltage (+5V) to any input pins of ULN2003A, related out pins will be grounded for that reason the common wire of lamps are selected positive in order to obtain lights from lamps. The following figure is showing how the control of lamps executed.

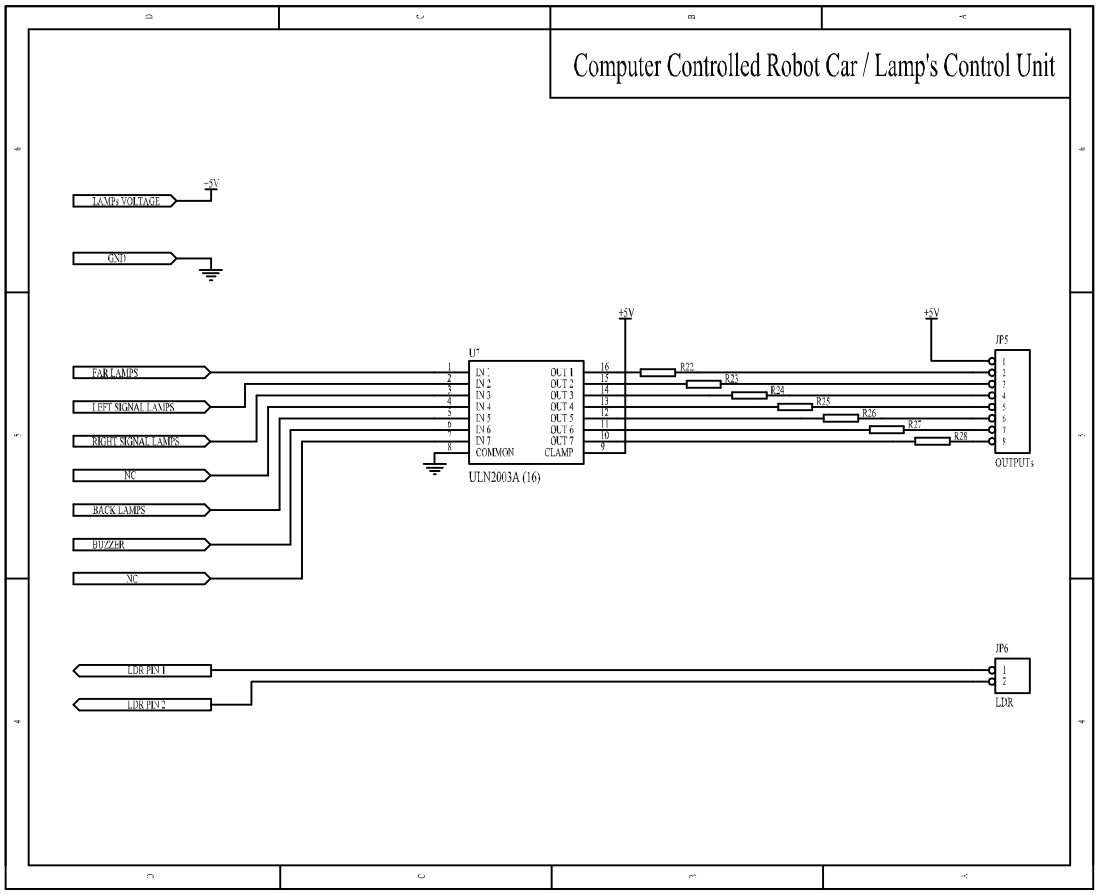


Figure 3.13. Lamps control unit

3.8. Temperature Sensor

A temperature sensor called LM35 has been used in this study in order to give the user in front of the personal computer temperature information related to the location where robot car is. As it seen in Figure 3.11 on page 20 , temperature sensor is connected to the JP8 terminal on device. Third pin of JP8 terminal in Figure 3.11 is used as an output signal of LM35. That output signal has been connected to analog buffer and then MCU unit as an analog signal.

3.9. Line Follow

A kit called Linbot has been purchased from Microrobot Company in order to follow line and detect obstacle in front of the robot car. Linbot is an advanced line tracer robot kit. It has one pair of infrared emitters and sensors directed forward, as well as three pairs of infrared emitters and sensors directed downward. Linbot can follow a black line placed on a white floor as well as a white line on a black floor. When it meets with an obstacle while following the line, robot car has an ability to stop. When remove the obstacle in front of robot car, it continues to follow line again. In this study, just circuit of whole kit Linbot has been used and mounted at the bottom of robot car. The following figure is showing the sensor leds and mounting components.

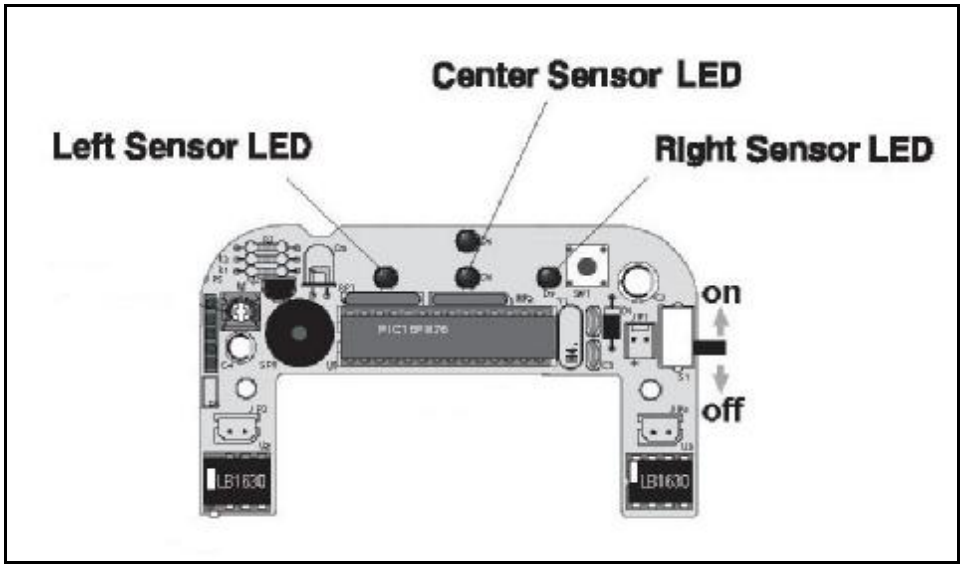


Figure 3.14. Top view of line follow circuit (Line Tracer Linbot User Manual, 2004)

Three couple of infrared emitters and sensors have been mounted on the back surface of plate as it shows in the following Figure 3.15. When the infrared led meets a black line, related led seen in Figure 3.14 lights up.

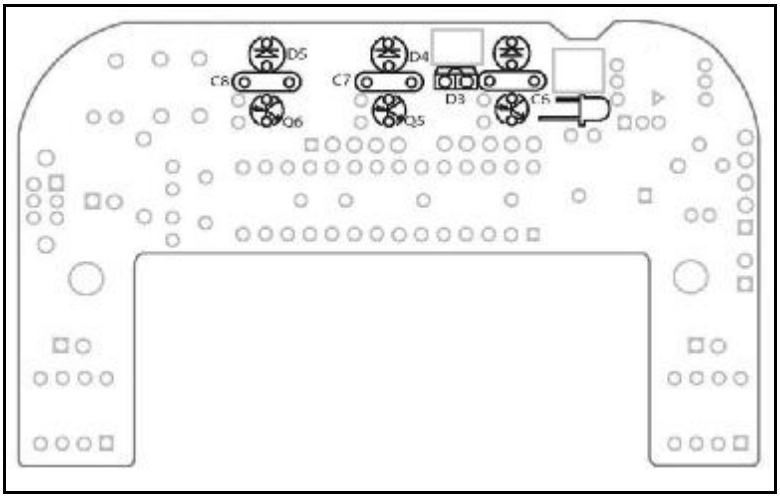


Figure 3.15. Bottom view of line follow circuit
(Line Tracer Linbot User Manual, 2004)

As it seen in Figure 3.15, ceramic capacitors (C6, C7, C8) have been used between infrared emitter and sensor in order to prevent light from the infrared led interfering with the sensors. (Line Tracer Linbot User Manual, 2004)

The following figure is showing the mounting type of ceramic capacitors indicated at the right side on Figure 3.16

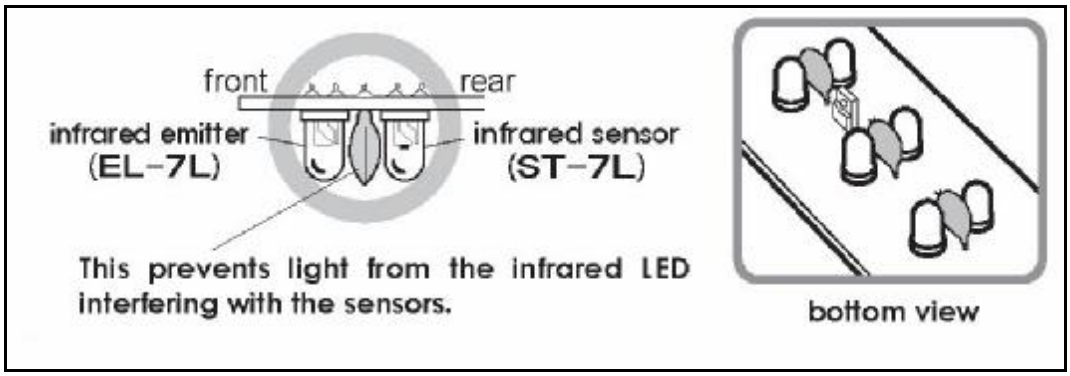


Figure 3.16. Mounting type of ceramic capacitor
(Line Tracer Linbot User Manual, 2004)

The following figures are showing how robot car changes the direction according to line and obstacle.

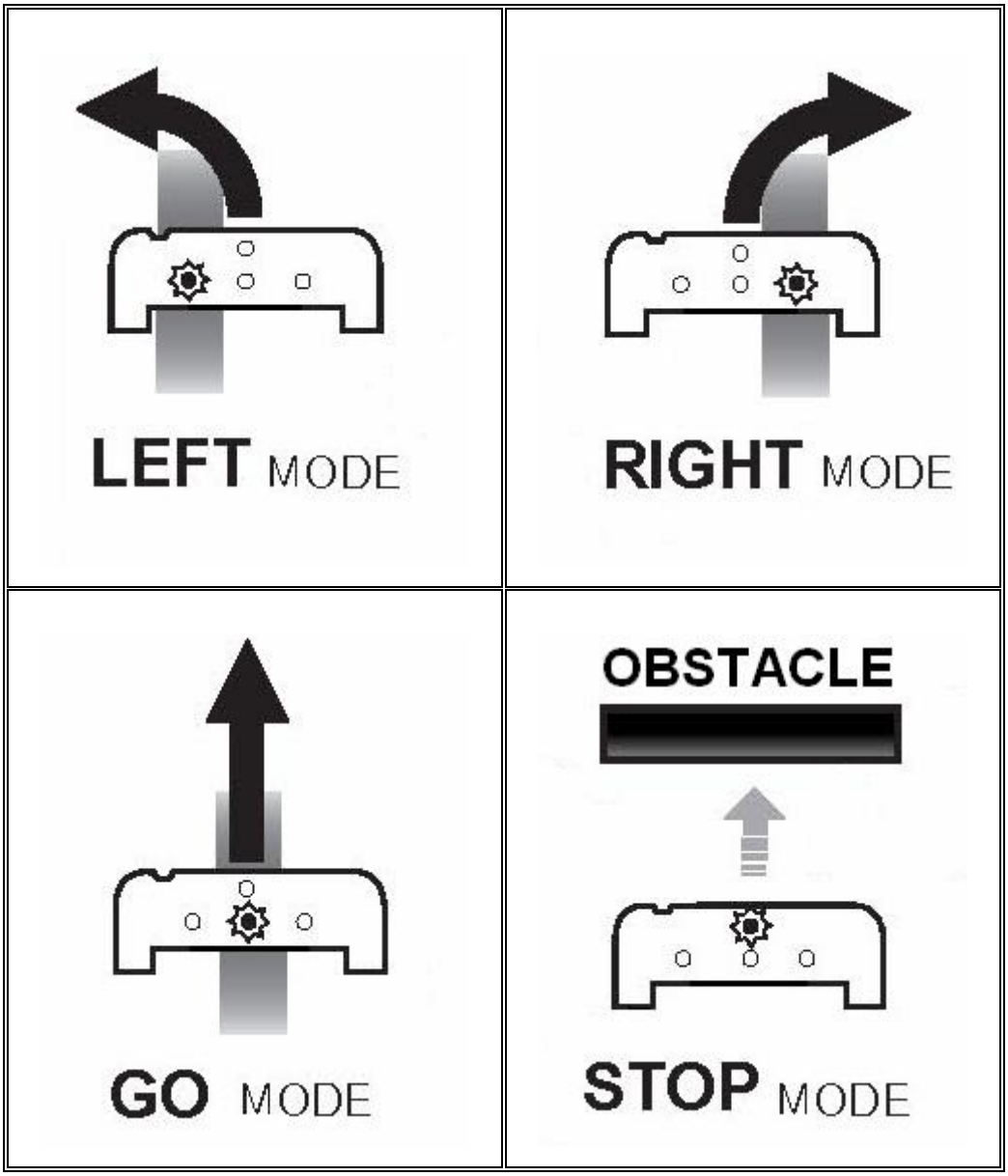


Figure 3.17. Direction of robot car according to line and obstacle
(Line Tracer Linbot User Manual, 2004)

As mentioned before, leds mounted on top of circuit are lighting up when infrared sensor meets a black line. In this study, leds voltage has been used as an input signal to main MCU Unit. That is, main MCU unit (PIC 18F452) is taking digital signals

from the related pin (15,16,18) of PIC16F876 mounted on line follow circuit. As soon as MCU Unit receives these signals, it has direction motor moved to related direction. The following figure is showing whole connection diagram of circuit.

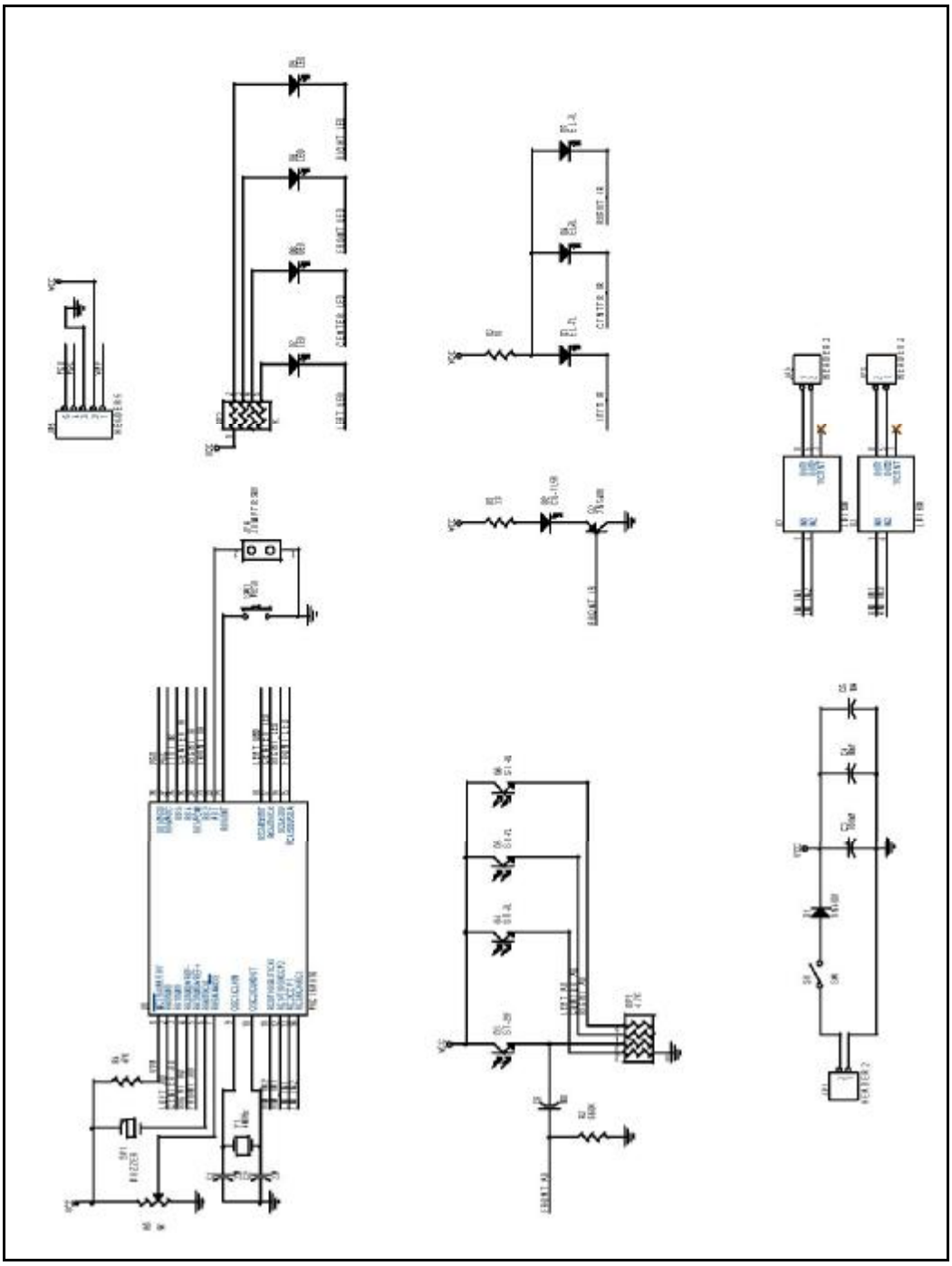


Figure 3.18. Line follow circuit connection diagram
(Robotics and Electronics Technology, 2003)

4. WIRELESS COMMUNICATION

A couple of modem module marked UDEA type UFM-A12 WPA for wireless communication between computer and mobile platform has been used.

General features of the module are as follows,

1) 868 MHz or 915MHz UHF band. Compatible with European EN300 220 standard.

2) High frequency stability

3) Ideal for long-range application with main power.

(UFM-A12 WPA Modem Module Operation Guide, 2005)

General view of modem unit is as follows.

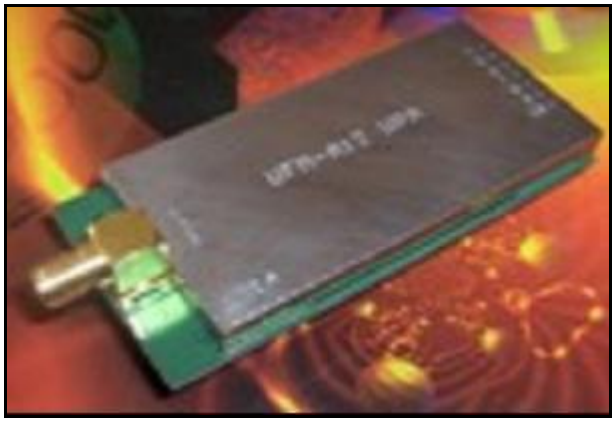


Figure 4.1. General view of RF Modem
(UFM-A12 WPA Modem Module Operation Guide, 2005)

The UFM-A12 WPA UHF FSK data transceiver modem module is developed to cover a band plan ERC Recommendation on Short Range Device (SRD) in the range of 868MHz ISM band. The UFM-A12 WPA is designed for PCB mounting. A simple wire can be soldered to the antenna input or external antenna can be used.

4.1. Supply Voltage

UFM-A12WPA has a voltage regulator; user must guarantee stable voltage in the given range. Supply voltage must be used within specified voltage. The module shows unstable function with the voltage lower than specified. If the voltage which connected to the Vcc (+) and Ground (-) terminal is beyond the maximum voltage given in the technical specification or reversed, the module will be permanently damaged. To enable a low minimum voltage, no internal circuit is used to prevent damage by incorrect polarity. If a higher supply voltage is available then a simple diode can be inserted in connection line to the Vcc terminal to prevent damage by incorrect polarity. Any more then ±100 mV change in voltage supply of circuit will cause unstable function. (UFM-A12 WPA Modem Module Operation Guide, 2005)

4.2. Connecting to Microcontroller

The microcontroller use a UART pins can be used for data to be transmitted and data received. Microcontroller interface for modem is shown below.

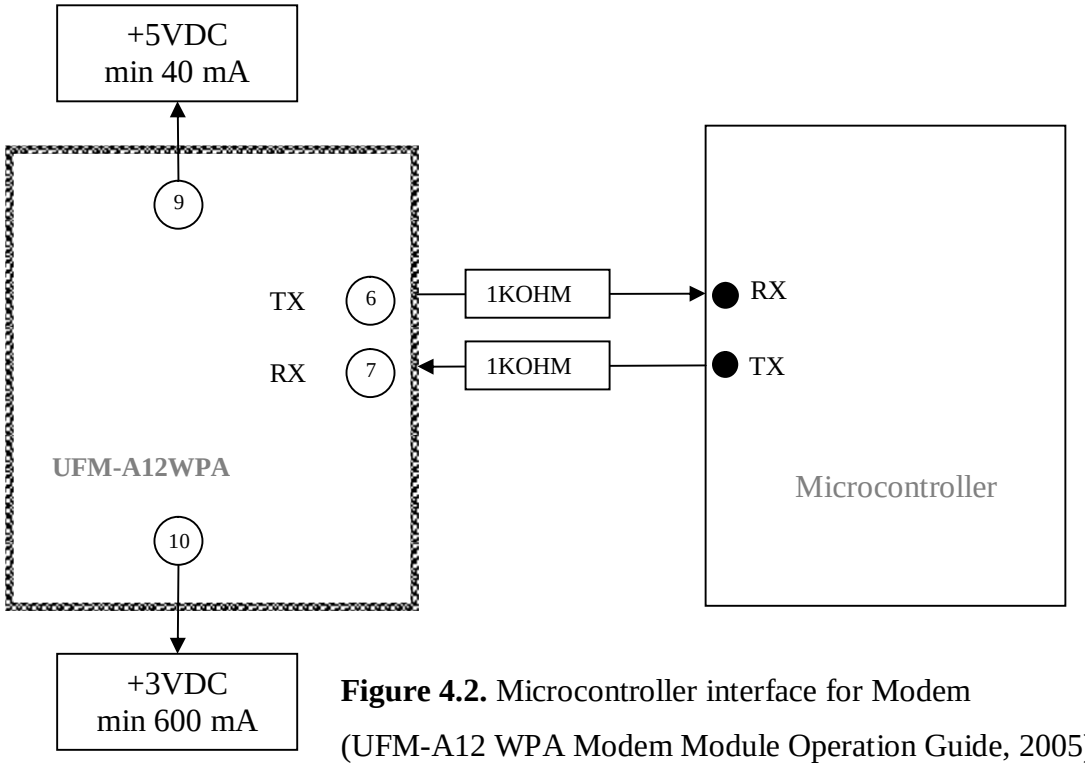


Figure 4.2. Microcontroller interface for Modem (UFM-A12 WPA Modem Module Operation Guide, 2005)

4.3. Data Communication

4.3.1. Physical Characteristics

- Connection : Module Pins
- Transmission : Serial asynchronous (UART)
- Baud Rate : 2.4 Kbitps
- Link : TTL 5 V - RS 232

(UFM-A12 WPA Modem Module Operation Guide, 2005)

4.3.2. Data Format

- 8 data bits, no parity, 1 stop bit(8N1)
- CTS and RTS are not used

(UFM-A12 WPA Modem Module Operation Guide, 2005)

4.3.3. General Data Format

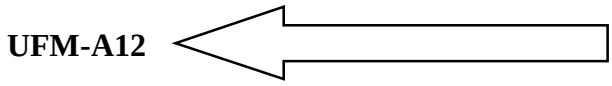
$	R	F	DATA	E	N	D	CR	LF
24h	52h	46h	BYTE (Max 72 BYTE)	45h	4Eh	44h	0Dh	0Ah

Start of Frame **Data** **End of Frame**

Figure 4.3. Superframe structure of RF Data
(UFM-A12 WPA Modem Module Operation Guide, 2005)

4.3.4. Data Input UFM-A12

The data should be given as shown in Figure 4.4. to module. First start of frame (3 byte), then data(max 72 byte) and then end of frame (5 byte). The MAC layer of the module add necessary payload (preamble, synchronization header, CRC) to given data and then send it to RF. (UFM-A12 WPA Modem Module Operation Guide, 2005)

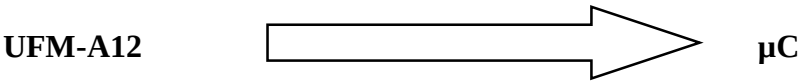


$	R	F	DATA	E	N	D	CR	LF
24h	52h	46h	(Max 72 BYTE)	45h	4Eh	44h	0Dh	0Ah

Figure 4.4. Frame structure of Input
(UFM-A12 WPA Modem Module Operation Guide, 2005)

4.3.5. Data Output UFM-A12

The received RF data is given as shown in Figure 4.5. to output. First start of frame (3 byte), then received data(max 72 byte) and then end of frame (5 byte). The MAC layer of the module detach the RF payload (preamble, synchronization header, CRC) and give the data to output port. (UFM-A12 WPA Modem Module Operation Guide, 2005)



$	R	F	DATA	E	N	D	CR	LF
24h	52h	46h	(Max 72 BYTE)	45h	4Eh	44h	0Dh	0Ah

Figure 4.5. Frame structure of output
(UFM-A12 WPA Modem Module Operation Guide, 2005)

4.4. Antenna

Most important for effective data transmission is selection of a good antenna, and RF grounding, both for the transmitter and receiver. Without an antenna it is impossible to transmit data over a long distance range. The UFM-A12 has a simple antenna input pin. Any suitable 868MHz UHF antenna can be connected to it. If the receiving antenna is installed away from the module, a 50-Ohm Coax antenna wire

can be used. The shielding of the antenna wire should be soldered to the case near the antenna input of the module. (Udea Wireless Technologies, 1999)

In most cases the following basic rules will help for implementation:

- Connect an antenna with 50-Ohm impedance.
- Lambda/4 whip antenna length is approximately 8.6cm for 868MHz.
- Place the antenna vertically, straight up or down from the transmitter and receiver module.
- Do not cover the antenna with metal parts.
- The human body can have similar effects like metal objects. Pocket transmitters should be taken in the hand and put in a position away from the body and pointed in the direction of the receiver.
- Best range is achieved if the transmitter and receiver antenna have a direct visual connection. Any object in between the transmitter and receiver antenna, and metallic objects in particular, will decrease the range.
- The transmission is influenced by possibility to have data error by overlaying the direct and reflected signal (UDEA Wireless Technologies, 1999)

4.5. Modem Device on Robot Car

One of the couple of modems has been mounted on main PCB in robot car and voltage requirements of modem as explained in section 4.1. were provided by voltage regulators that generate stable +5V and +3V. Although modem allow us to send 72 byte explained in section 4.3.3 and 4.3.4, 10 byte data is being sent for communications of modems in this study. Configuration of 10 byte data (Receive and Transmit) has been explained below.

Rxcommands[10]={0,0,0,0,0,0,0,0,0,0}

[0]: low nibble----Camera Position (3 bits)

high nibble----- Motion Motor PWM (4,5,6) and 7^{th} bit is forward or reverse

[1]: Low nibble: Direction Motor left or right (0,1 bits)

Low nibble: Automatically Line Follow Mode (2^{nd} bits)

The balance bytes of Rxcommands are not assigned.

Txcommands[10]={0,0,0,0,0,0,0,0,0,0}

[0]: Error

[1]: Warning Message

[2]: Start up indicator

[3]: Battery 1 (Analog 8 bit)

[4]: Battery 2 (Analog 8 bit)

[5]: Temperature Sensor (Analog 8 bit)

The balance bytes of Txcommands are not assigned.

The following figure is showing the connection of modem to microcontroller and voltage suppliers.

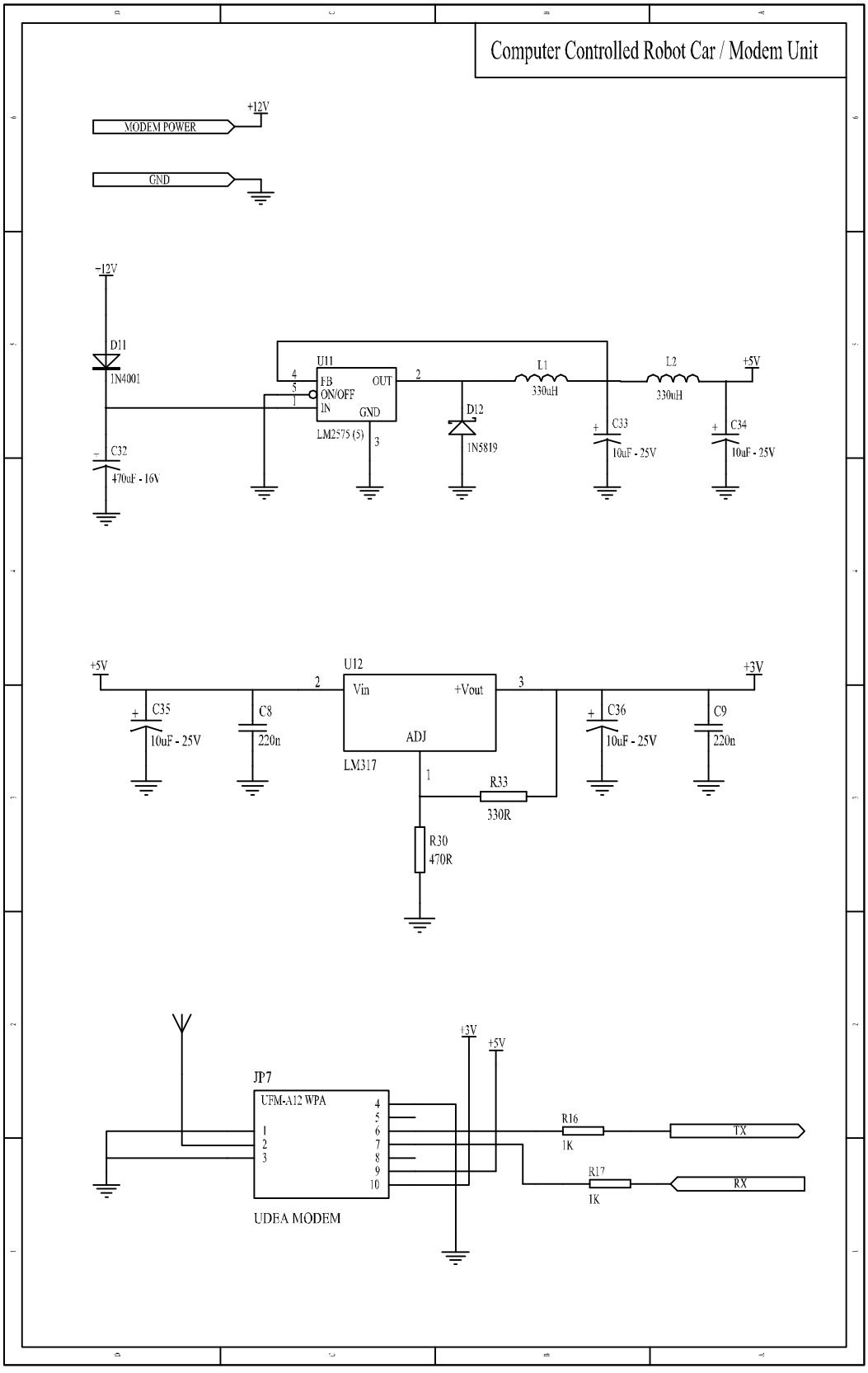
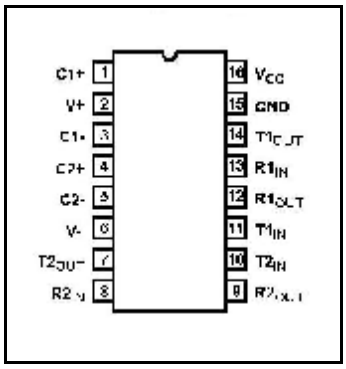


Figure 4.6. Modem Unit

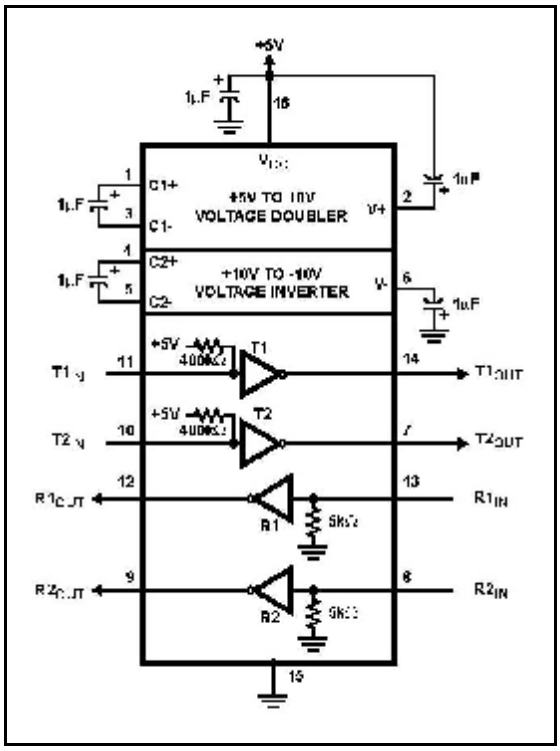
4.6. Modem Device for Remote Control

Modem device has been used to receive signals from computer and send received signals to other modem in robot car. DB9 connector was used for connection to computer's comport. As explained in section 4.1, voltage requirement of modem has been provided by using voltage regulators that gives stable +5V and +3V in circuit. As it seen in Figure 4.7, HIN232CP +5V Powered RS-232 Transmitters /Receivers has been used to obtain 10V for RS232 communication

It requires a single +5V power supply and feature onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply. The family of devices offer a wide variety of RS-232 transmitter/receiver combinations to accommodate applications. (Electronics Components Datasheet, 2005)



A



B

Figure 4.7. HIN232CP (a) Top view (b) Pin configuration
(Electronics Components Datasheet, 2005)

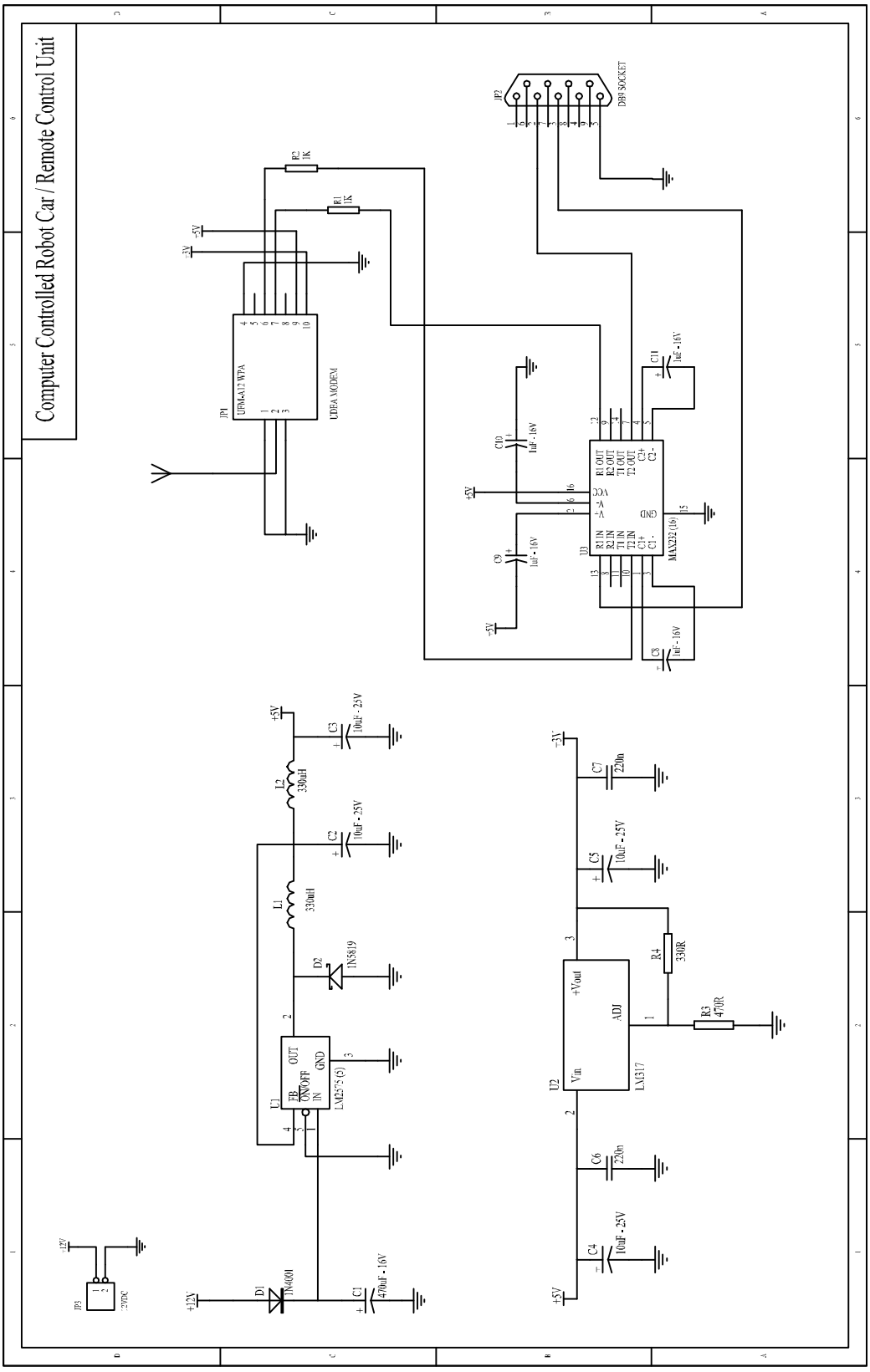


Figure 4.8. Remote Controller

5. WIRELESS VIDEO AND AUDIO CAMERA

A small wireless camera has been mounted on the top of the robot car and it sends views where robot car is to receiver unit to be seen by user in front of computer.

5.1. Transmitter and Receiver Unit

The camera mounted on the top of the robot car is including transmitter unit inside and views coming from the camera are being taken by receiver unit wirelessly. Voltage requirement of the camera has been provided by +9V DC battery mounted on the same platform with camera. Receiver unit has been connected to Digital Video Recorder Card installed on computer main board directly via BNC and video cable. The following figure is showing which camera and receiver unit are used in this study.

Figure 5.1. Camera and Receiver Unit
(JMK Shenzhen Jiameikang Science 1, 2003)

5.2. DVR Card

DVR Card (Digital Video Recorder Monitoring and Controlling Card) is a completion part of the camera and receiver unit so that views where the robot car is can be seen on computer's monitor. DVR card needs to be mounted on main board of computer and related software program needs to be installed to computer, respectively. As it seen in the technical specifications of DVR card, it has four channel to be connected four cameras separately at the same time.

Figure 5.2. DVR Card
(JMK Shenzhen Jiameikang Science 2, 2003)

Connection application is as follows,

- Computer→Video Capture Card→BNC→Video Data Cable→Receiver
  (JMK Shenzhen Jiameikang Science 2, 2003)

6. OBSERVATION OF COMPLETED MOBILE PLATFORM

6.1. General Specifications of Computer Controlled Robot Car

General specifications and abilities of robot car obtained at the end of this study have been explained as follows.

- Moving motor forward and reverse using keyboard or mouse.
- Lamps in front of car are ON when the robot inside the dark area.
- Signal lamps are flushing while robot rotate left or right.
- Buzzer and back signal lamps are ON when robot is going reverse
- Motor speed has been controlled with 4 speed values. (PWM Control)
- 360° rotation of camera on the top of the car by using keyboard
- Speed display on the User interface. (Tachometer not used)
- Battery Voltage value on the User interface.
- Temperature indicator on the User interface
- Warning message on the User interface when the robot face a problem.
- Teach the way to robot, then it will record the way, and then it will go as you teach
- Stop, when the robot sees an obstacle in front of it.

6.2. User Interface

The following screen has been created by Visual Basic to be controlled the robot car by user in front of computer. These buttons and displays were designed for user who operates the car remotely. User can see battery voltages, temperature wherever robot is, warning message, speed of car, gear, scroll bar for camera position (added an red array at right side to increase the visibility, that array rotating as the camera rotate at the same time), button (Kayıda Başla) for recording as teaching the way, button (Kayıt Stop) for stop recording, button (Kayıt Oku) for moving robot as taught, and last one is auto button to move robot in line follow mode and stop when it faces an obstacle.

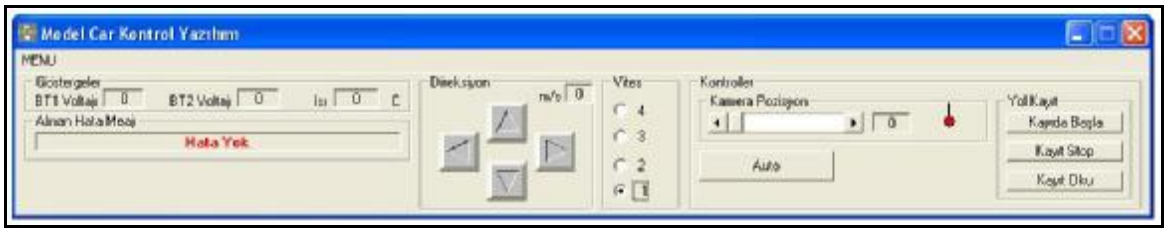


Figure 6.1. User Interface

The following flow chart is showing how the procedure is being executed in MCU Unit (PIC 18F452). As soon as robot car is energized by switching the button mounted under the car, camera will find the zero point, PC takes a signal from the modem which MCU is ready for getting commands and at the same time MCU read batteries and temperature values and then write these values to Txcommands arrays. The following functions mentioned in flow chart will be explained to make it more clear.

- **Start:** Energization of Robot car with both 12V dry batteries.
- **MCU Init:** Port arrangements as an input or output are executed.
- **Startup Settings:** MCU send signals to stepper motor for zero point.
  MCU sends a start up indicator bit to PC via Modem.
- **Measuring:** MCU takes analog signals from the batteries and temperature sensor and then converts it to digital signal (8bit) and then writes to Txcommands array to be sent to PC.
- **Send my parameters:** MCU sends the parameters means that battery voltages and temperature voltages and warning messages to PC via Modem.
- **Check for UART for Rx:** Check whether data coming from PC is ready.
- **Get Rx Commands:** MCU gets the 10byte signal from Modem and later interprets these bytes and bits.
- **Command Synthesis:** Check the related bytes and bits received from PC and decide what to do (Forward, Reverse, Left, Right, Rotate Camera, Change gear etc.)

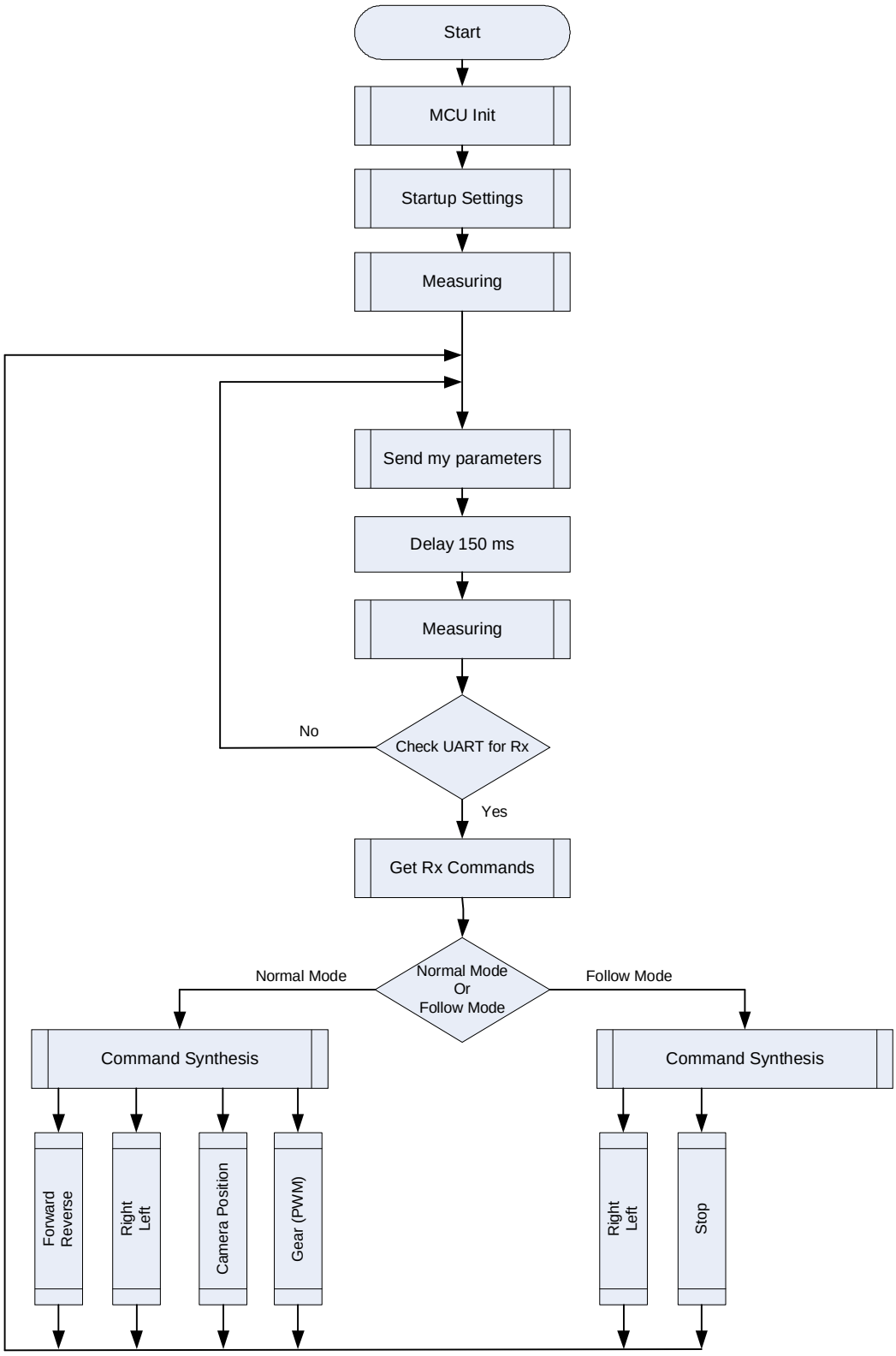


Figure 6.2. Flowchart for MCU on Mobile Platform

6.3. General Views of Completed Robot Car

Figure 6.3. General view of robot car

Figure 6.4. Camera mounted on the top of robot car

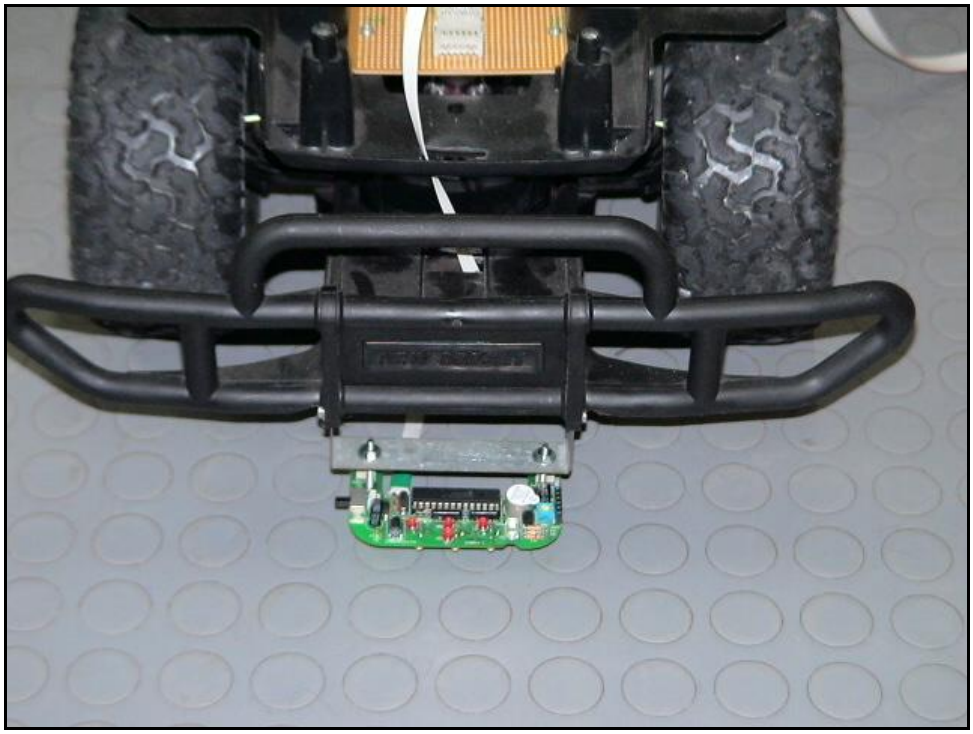


Figure 6.5. Line follow and obstacle detection circuit

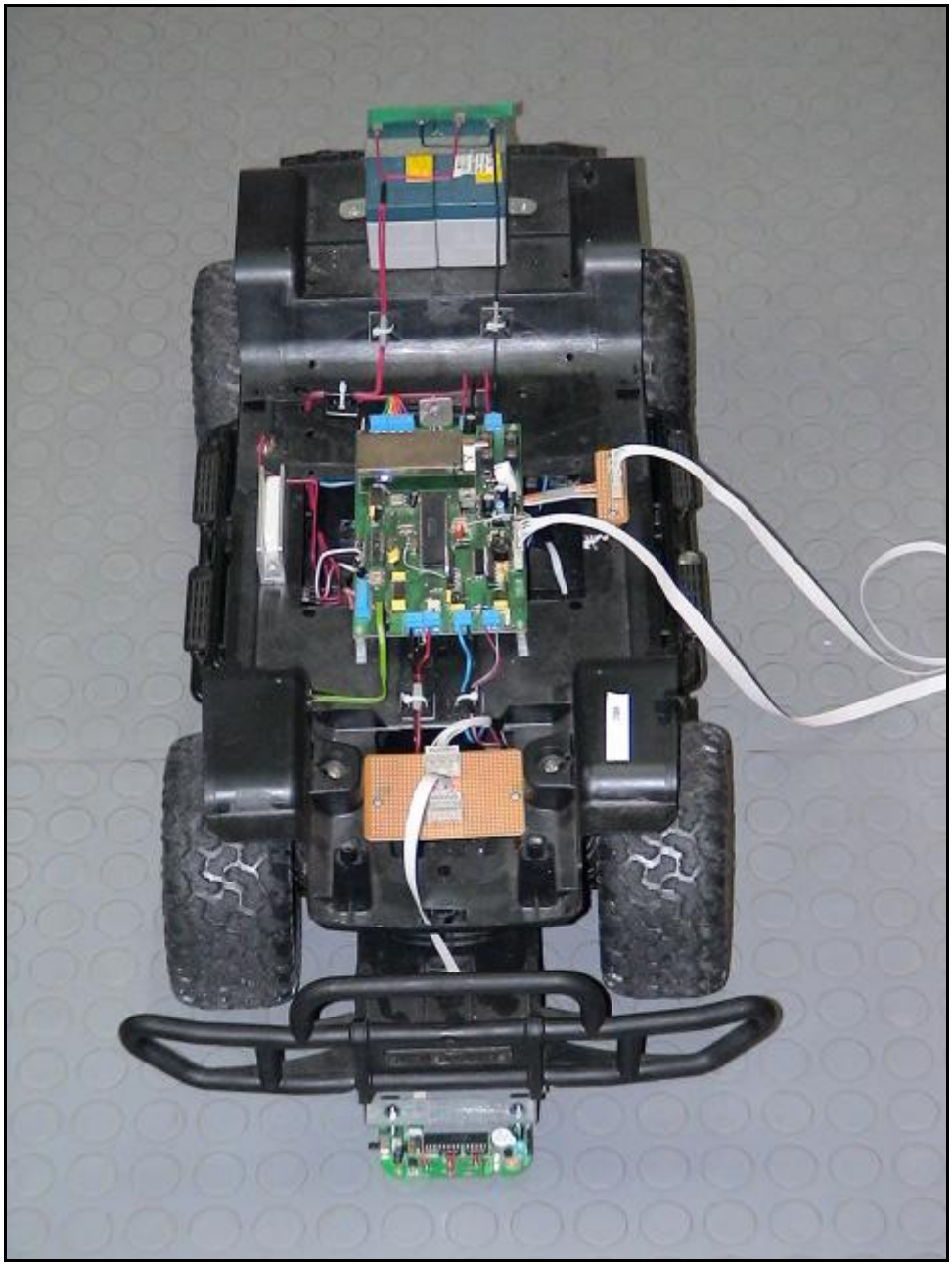


Figure 6.6. Completed electronic circuit in robot car

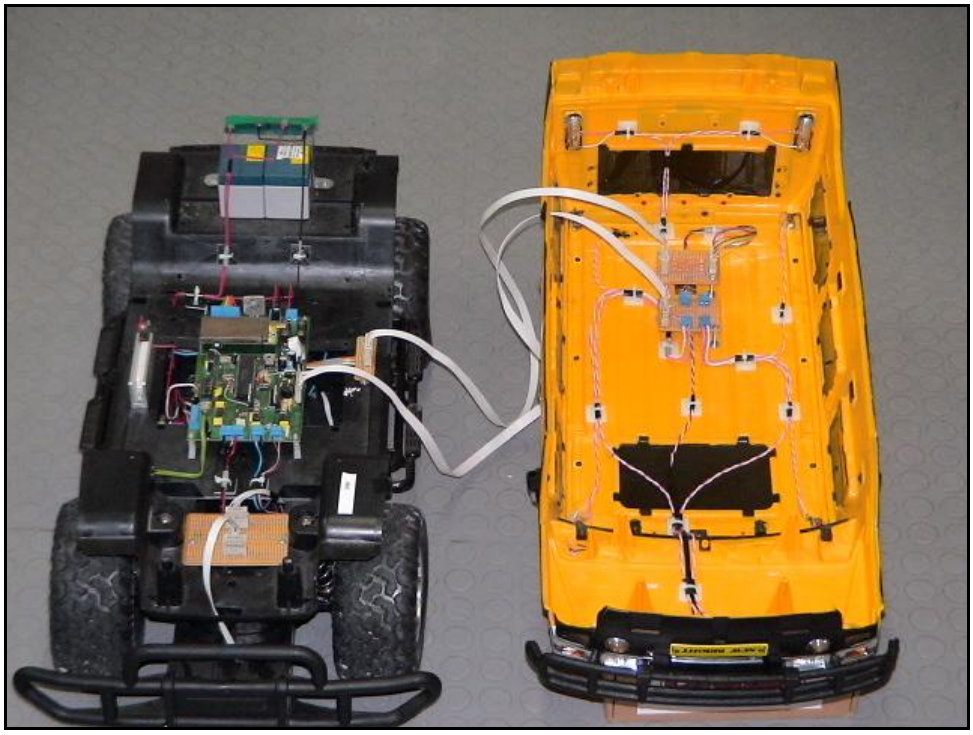


Figure 6.7. Both cover and mobile platform's view

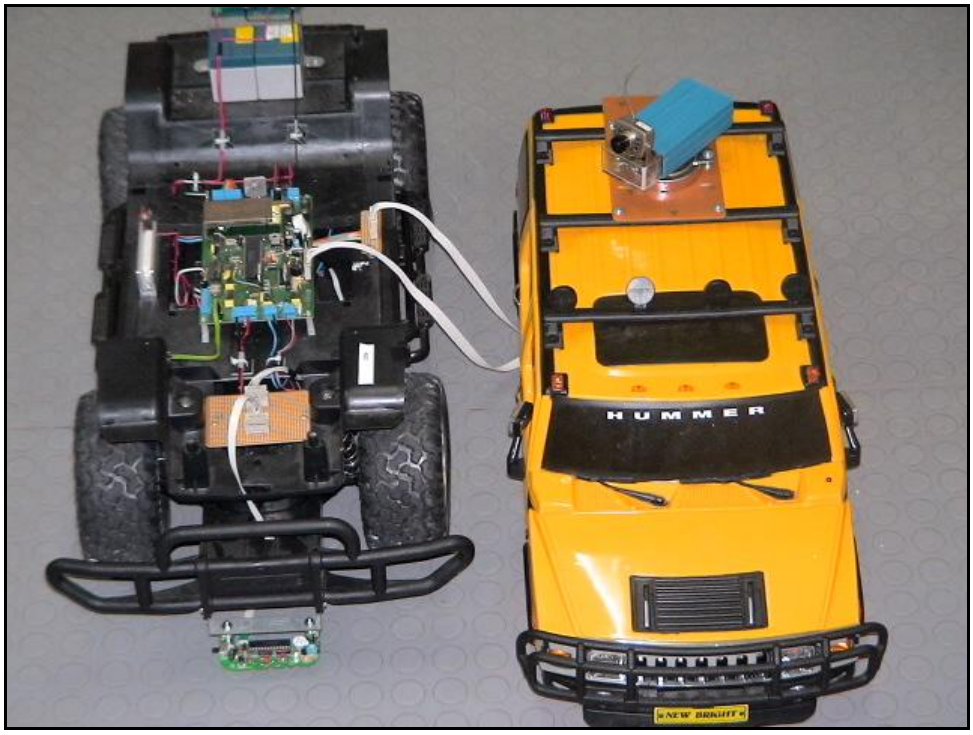


Figure 6.8. Both cameras on cover and mobile platform's view

7. RESULTS and FUTURE WORKS

A computer controlled car application has been carried out by using wireless modem to provide the communication between computer and robot microcontroller card. In addition to the specifications of robot car, a camera has been put on the top of car and views coming from camera have been seen on the computer screen. User in front of the computer can get the camera rotated 360 degree around itself to be able see environment where the robot car is. As soon as the car inside the tunnel, the lamps in front of car switch on automatically. When it goes backward, warning signal buzzer come up as the car in real life. It is going to be stopped, when it face to face an obstacle in front of it.

The PIC 18F452 microcontroller is a brain of robot car so that it controls the subsystems such as DC motor driver, Lamp Control Unit, stepper motor driver etc. First of all, microcontroller receives 10 byte data via modem device which has an ability to send / receive 72 byte, and then microcontroller proceeds these data seperately. After separate data come from computer, it sends the related signal to related subsystem's pin. On the other hand, it reads the analog digital converter time to time. Dry batteries and temperature sensor have been connected to microcontroller's Port A to be converted to 8 bit digital signal. Port B (4,5,6,7) pins connected to stepper motor on the top of the robot car cover to be able to rotate it 45 degree by 45 degree. In addition, a magnet was fixed under the camera box so that this magnet should find the magnetic contact mounted on steady plate for zero point. For this stepper motor application, ULN2003A IC was used because of the fact that digital output signal of microcontroller is not enough to feed stepper motor. Microcontroller is sending the digital signal to the related pin of ULN2003A and related output of ULN2003A sending the necessary voltage to stepper motor. The same situation is available for Lamps control unit as well. In line follow mode, leds mounted on top of circuit are lighting up when infrared sensor meets a black line on road. In this study, leds voltage has been used as an input signal to main MCU Unit. That is, main MCU unit is receiving digital signals from the related pin (15,16,18) of PIC16F876 mounted on circuit. As soon as MCU Unit receives these signals, it has

motion and direction motors moved to related direction. Microcontroller generates PWM signal to move robot car in 4 different speeds. L6203 ICs have been used to drive both motors in Forward, Reverse, left and right direction. PWM means that average voltage is being changed in a period by microcontroller, so motion motor's speed increase as the average voltage increased.

In future, the performance of the obstacle detection array can be greatly improved by simply replacing the infrared rangers with ultrasonic rangers that have a greater distance sensing range. A stepper motor and a ultrasonic sensor on stepper motor may be put on the top of robot car so that user can measure all distance away from robot car as user rotate the stepper motor under ultrasonic sensor. In addition, image processing can be applied by the views taken wireless from the camera on the top the car.

8. CONCLUSIONS

An application of computer controlled robot car has been completed successfully in this study. Three DC motors have been chosen to change rotation of the mobile robot as right and left, go on forward-backward and rotate the camera which put on the robot, cycled 360 degree around itself. The duties of the control device on robot are to communicate with the computer, to generate electrical signals for motion, to process the signals that will be taken from circuit including IR sensor, to convert the information which will be come from computer to electrical signals and besides, to send computer the signals that will be taken from the sensor. Wireless communication between robot and computer will be provided with transmitter and receiver units.

The information regarding the status of movement, rotation, speed, lighting and camera are being sent to control device with wireless transmitter by computer using high level programmed language. In addition to this, the information consist of the distance of robot to object, daylight-dark, temperature have been sent to computer by the control device to be viewed by operator of computer. A surface which provided an interaction between operator and hardware has been created. In this study, the specification of the unknown area can be learned without any human activities and tried to process about the status of that area.

REFERENCES

ARD, A. and SKIPPER, J. December 2004, Preliminary Robotics Design Document 'Robotics Capstone Design'

Active Robots, Robotics and Electronics Technology www.active-robots.com/products/robots/line/linbot-manual.pdf, 2003

Catalog of Electronic Components www.chipcatalog.com/Allegro/ULN2003A.htm, 2004

Electronic Components Datasheet www.datasheet4u.com/download.php?id=75403, 2005

GOTT, O., D., May 2003, The Smart Car Project: A Case Study in Computer-Mediated Control mechatronics.eng.buffalo.edu/research/smartcar/Publications/Daniel_Project_June03.pdf

Imperial College London, Faculty of Engineering, Department of Computing doc.ic.ac.uk/~ih/doc/stepper/others/, 1998

Imperial College London, Faculty of Engineering, Department of Computing www.doc.ic.ac.uk/~ih/doc/stepper/control2/sequence.html, 1997

JEONG, Y., 2006, Operational Amplifiers Lab. Notes www.ag.arizona.edu/~jyyoon/lab6.pdf, 2006

JMK Shenzhen Jiameikang Science Co., Ltd 1 www.jmk.com.cn/ey/productShow.asp?id=741, 2003

JMK Shenzhen Jiameikang Science Co., Ltd 2 www.jmk.com.cn/ey/productShow.asp?id=541, 2003

Line Tracer Linbot User Manual, 2004

MANSEUR, R., 2006, Robot Modelling & Kinematics, 367pp

Microchip Technology Inc., 2006, PIC18FXX2 Datasheet

National Semiconductors www.cache.national.com/ds/LM/LM35.pdf, 2006

Netrino Technical Library www.netrino.com/Publications/Glossary/PWM.html, 2001

ÖZEN, S., YILDIZ, E. and UZUN, T., 2000, Bilgisayar Kontrollü Gezgin Robot Uygulaması, ELECO'2000

Peter H. Anderson, 1997, Pulse Width Modulation
www.phanderson.com/PIC/16C84/pwm.html, 1997

STMicroelectronics 1, 2006, L7800 Series Positive Voltage Regulator ICs Datasheet
www.st.com/stonline/products/literature/ds/2143/l7805c.pdf, 2006

STMicroelectronics 2, 2006, ULN200XA Seven Darlington Array ICs Datasheet
www.st.com/stonline/books/pdf/docs/5279.pdf, 2006

STMicroelectronics 3, 2006, L6201-L6202-L6203 ICs Datasheet
www.st.com/stonline/products/literature/ds/1373.pdf, 2006

STMicroelectronics 4, 2006, LM124-LM224-LM324 ICs Datasheet
www.st.com/stonline/products/literature/ds/2156.pdf, 2006

UDEA Wireless Technologies
www.udea.com.tr, 1999

UFM-A12 WPA Modem Module Operation Guide, 2005

WARD, S. and STOOR, B., 1999, Computer Controlled Vehicle
courses.ece.uiuc.edu/ece390/archive/archive-sum99/prizes/rc_vehicle/public/index.htm

YILDIZ, N., and UZUN, T., 2000, Araba Benzeri Bir Gezgin Robotun Donanımı ile Yazılımının Tasarlanmasi Ve Gerçekleştirilmesi, ELECO'2000

BIOGRAPHY

Ş.Şamil ASLAN was born on December 20, 1977, in Adana, Turkey, as the second of four children of Ali ASLAN and Ayşe ASLAN. He was educated at the Electronics department of Technical and Industrial Vocational High School in Osmaniye. He received his B.Sc. degree from Kocaeli University in the field of Electronics and Communication Engineering in 2001.

After he graduated the university, he started as a master student at the Electrical and Electronics Engineering Department of Çukurova University. Meanwhile he has been working in Baku-Tbilisi-Ceyhan Crude Oil Pipeline Project as a site representative engineer for Tekfen Construction and Installation Co. Inc., for 3 years.

APPENDIX A

The following program main.c has been loaded to PIC18F452 microcontroller by using IC-PROG software that converts a program written in C Programming Language to hex file.

- main.c file

```
// First Includes
#include "C:\COmputerControlledRobotCar\MCU_PIC18F452\main.h"
#include<stdlib.h>
#define     PIC18F452

enum{
   pas=0,
   act,
   thereIsNewCommands,
   noCommand,
   begin,
   end,
   clrWarnMsgs,
   rxCommandsVar,
   txCommandsVar,
   right,
   left,
   zero=0,
   normalMode,
   followMode
};

//-------------------------- Interface ---------------------------------------------
#define     UartAnswerWaitingTime   50000
#define     mainProcRunningFreq     1
#define     ADCinitTime             10
#define     lightSenseThreshold     1

//-------------------------- Step Motor Constants ----------------------------------
#define     stepMotorTurningSpeed   25

//-------------------------- Global Definitions ------------------------------------
// BITs ;
#define  stepMotorPort  pb
#define  followBits     pc

// VARIABLEs;
unsigned long  globalPositionKeeper=0;
char  lastDirection=0;
char  modeIndicator=normalMode;
char  globalUARTsenser=0;

//-------------------------- String Arrays -----------------------------------------
char rxCommands[10]={0,0,0,0,0,0,0,0,0,0};   // Used for command buffer from PC UART
/*
     Byte Assignments for rxCommands ;

     rxCommands[0]=  Low nibble      : Camera position (4 bit)
                     High nibble     : Motion motor pwm managing (3 bit)
                     7th Bit         : Forward or back control bit
     rxCommands[1]=  First two bits  : Direction motor managing bits (0th and 1st)
                     Second bit      : Automatical line follow mode bit (2nd)
     rxCommands[2]=  Not Assigned
     rxCommands[3]=  Not Assigned
     rxCommands[4]=  Not Assigned
     rxCommands[5]=  Not Assigned
     rxCommands[6]=  Not Assigned
     rxCommands[7]=  Not Assigned
     rxCommands[8]=  Not Assigned
     rxCommands[9]=  Not Assigned
*/

char txCommands[10]={0,0,1,0,0,0,0,0,0,0};   // Parameters to send to PC
/*
     Byte Assignments for txCommands ;

     txCommands[0]=  Reporting error byte to PC
     txCommands[1]=  Warning messages
```

```
     txCommands[2]=  FirstBit: StartUp indicator
     txCommands[3]=  Battery 1 Analog
     txCommands[4]=  Battery 2 Analog
     txCommands[5]=  Temperature Sensor
     txCommands[6]=  Not Assigned
     txCommands[7]=  Not Assigned
     txCommands[8]=  Not Assigned
     txCommands[9]=  Not Assigned
*/
//-------------------------- Message Definitons ----------------------------------

//-------------------------- Errors ----------------------------------------------
#define        DoesNotFindZeroSwitch         1      // Camera zero switch not found
#define        DirectionEncoderDosnotAnswer  2
#define        tooWideLineInFollowMode       3
#define        couldnotFindLineInFollowMode  4

//-------------------------- Warnings --------------------------------------------
#define        CameraRotationCommandUndefined      1

//-------------------------- Command Set -----------------------------------------
#define     modemConstsBegin        sendModemConsts(begin)
#define     modemConstsEnd          sendModemConsts(end)
#define     zeroSwitch              pd0(0)
#define     barrierSenser           pb0(0)
#define     motionMotorForward(x)   pb1(x)
#define     motionMotorBack(x)      pb2(x)
#define     directionMotorEn(x)     pb3(x)
#define     directionMotorLeft(x)   pe0(x)
#define     directionMotorRight(x)  pe1(x)
#define     directionZero           pc5(0)

//-------------------------- Lamps Commands --------------------------------------
#define     setFarLamp(x)           pd1(x)
#define     leftSignalLamp(x)       pd2(x)
#define     rightSignalLamp(x)      pd3(x)
#define     setBuzzer(x)            pd6(x)
#define     setBackLamp(x)          pd5(x)

// SECOND INCLUDEs
#include<picos.c>

//-------------------------- Prototypes ------------------------------------------
void  mcuInit(void);                 // MCU Initial
void  startupSettings(void);         // Startup Settings
void  commandSynthesis(void);        // Command synthesis for managing device
void  interpretCommand0(char);       // Command0 interpreting
void  interpretCommand1(char);       // Command1 interpreting
void  interpretCommand2(char);       // Command2 interpreting
void  interpretCommand3(char);       // Command3 interpreting
void  interpretCommand4(char);       // Command4 interpreting
void  interpretCommand5(char);       // Command5 interpreting
void  interpretCommand6(char);       // Command6 interpreting
void  interpretCommand7(char);       // Command7 interpreting
void  interpretCommand8(char);       // Command8 interpreting
void  interpretCommand9(char);       // Command9 interpreting
void  setErr(char);                  // Error causes report. via txCommands[0]
void  setWarning(char);              // Warning messages report via txCommands[1]
void  stopAllHardware(void);         // All hardware stopping for error status
void  measuring(void);               // Analog measuring procedures
void  lineFollowMode(void);          // Line follow mode
char  smartDelay(char/*entryVal*/);  // Expanded delay routine with UART dedection
void  lampsManaging(void);           // Far Lamps Managing
void  roadBusy(void);                // Car road busy

//-------------------------- Modem Connection Procedures-----------------------------
void  checkUartForRx(void);          // UART Buffer checking & command receiving
void  sendMyParameters(void);        // Send my parameters
void  sendModemConsts(char);         // Modem Constants sending
void  getRxCommands(void);           // RX Commands getting

//-------------------------- Step Motor Control Routines for Camera ----------------
void  stepForward(void);             // Step motor forward turning
void  stepBack(void);                // Step motor back turning
void  cameraPosition(unsigned long); // Camera position setting as degree
void  findZero(void);                // Camera step motor zero fixing



//-------------------------- Direction Motor Managing ------------------------------
```

```
void  comeZero(void);               // Direction motor zero point
void  turnRight(void);              // Turn right
void  turnLeft(void);               // Turn left
void  turnDirection(char   /*direction*/,char /*turnTimeLength*/);



void  main(void)  {
   char  i;
   mcuInit();
   startupSettings();
   measuring();               // Temperature and batery voltages
   sendMyParameters();
   delay_ms(150);
   while(1) {
      measuring();            // Temperature and batery voltages
      if(modeIndicator==normalMode) {
         checkUartForRx();    // Continously loops until uart startbit detecting
         getRxCommands();
      }
      sendMyParameters();
      commandSynthesis();
      if(modeIndicator==normalMode) delay_ms(mainProcRunningFreq);
   }
}

void  roadBusy(void) {
   long  li;
   motionMotorForward(0);
   while(!barrierSenser) {
      if(++li>=10000) {
         li=0;setBuzzer(1);
         delay_ms(500);
         setBuzzer(0);
         delay_ms(500);
         setFarLamp(1);
         delay_ms(400);
         setFarLamp(0);
         delay_ms(600);
         setFarLamp(1);
         delay_ms(400);
         setFarLamp(0);
      }
      delay_us(400);
   }
   motionMotorForward(1);
}

void  lampsManaging(void) {
   static unsigned long lt;            // Far Lamps Acitivation Time Managing
   if(lt>65500) {
      if(rxCommands[1]&2) pd^=4;       // Left signals setting
      else pd&=0xFB;
      if(rxCommands[1]&1) pd^=8;       // Right signals setting
      else pd&=0xF7;
      set_ADC_Channel(2);
      delay_us(ADCinitTime);
      if(read_ADC()<lightSenseThreshold){setFarLamp(1);setBackLamp(1);}
      else {setFarLamp(0);setBackLamp(0);}
      if(rxCommands[0]&112) {          // Back Lamp Managing
         if(rxCommands[0]&128);        // Back Lamp and buzzer Managing
         else {setBackLamp(1);pd^=64;}  // Back Lamp and buzzer Managing
      }
      if(0==(rxCommands[0]&112))pd&=191;
      lt=0;
   }
   else ++lt;
}

char  smartDelay(char   entryVal) {
   char  i,ii;
   for(i=0;i<entryVal;++i) {
      for(ii=0;ii<20;++ii) {
         if(kbHit())return 1;
         delay_us(45);
      }
   }
   return   0;
}
void  lineFollowMode(void) {
```

```
   char  i;
   set_ADC_Channel(2);
   delay_us(ADCinitTime);
   while(1) {
      motionMotorForward(1);
      if(read_ADC()<lightSenseThreshold){setFarLamp(1);setBackLamp(1);} // Far Lamp
      else {setFarLamp(0);setBackLamp(0);}
      set_pwm1_duty(500);
      if(kbHit()){getRxCommands();modeIndicator=followMode;return;}

      switch(followBits&24) {
         case 0: setErr(tooWideLineInFollowMode);break;
         case 8: if(lastDirection!=left){lastDirection=left;turnDirection(left,20);}
                 for(i=0;i<250;++i) {
                    if(followBits&16)break;
                    if(smartDelay(10)) {
                       getRxCommands();
                       modeIndicator=followMode;
                       return;
                    }
                 }
                 if(i>=250){
                    setErr(couldnotFindLineInFollowMode);
                    modeIndicator=followMode;
                    return;
                 }
                 else{
                    comeZero();
                    lastDirection=zero;
                    if(globalUARTsenser) {
                       getRxCommands();
                       modeIndicator=followMode;
                       globalUARTsenser=0;
                       return;
                    }
                 }
                 break;
         case 16: if(lastDirection!=right) {
                     lastDirection=right;
                     turnDirection(right,20);
                  }
                  for(i=0;i<250;++i) {
                     if(followBits&8)break;
                     if(smartDelay(10)) {
                        getRxCommands();
                        modeIndicator=followMode;
                        return;
                     }
                  }
                  if(i>=250) {
                     setErr(couldnotFindLineInFollowMode);
                     modeIndicator=followMode;
                     return;
                  }
                  else {
                     comeZero();
                     lastDirection=zero;
                     if(globalUARTsenser) {
                        getRxCommands();
                        modeIndicator=followMode;
                        globalUARTsenser=0;
                        return;
                     }
                  }
                  break;
         case  24:   break;
      }
      if(!barrierSenser) roadBusy();
   }
}

void  measuring(void) {
   set_ADC_Channel(0);
   delay_us(ADCinitTime);
   txCommands[3]=read_ADC();
   set_ADC_channel(1);
   delay_us(ADCinitTime);
   txCommands[4]=read_ADC();
   set_ADC_channel(3);
   delay_us(ADCinitTime);
```

```
   txCommands[5]=read_ADC();
   delay_ms(1);
}

void  stopAllHardware(void) {
   motionMotorForward(0);
   motionMotorBack(0);
}

void  setWarning(char   warnID) {
   if(warnID!=clrWarnMsgs)txCommands[1]=warnID;
   else txCommands[1]=0;
}

void  setErr(char errID) {
   txCommands[0]=errID;
}

void  interpretCommand0(char  command) {
   // Camera bits interpreting
   switch(command&0x07) {
      case  0: cameraPosition(0);break;      // 0°
      case  1: cameraPosition(45);break;     // 45°
      case  2: cameraPosition(90);break;     // 90°
      case  3: cameraPosition(135);break;    // 135°
      case  4: cameraPosition(180);break;    // 180°
      case  5: cameraPosition(225);break;    // 225°
      case  6: cameraPosition(270);break;    // 270°
      case  7: cameraPosition(315);break;    // 315°
   }

   // Motion motor bits interpreting
   if(!(rxCommands[0]&112)){motionMotorForward(0);motionMotorBack(0);} // Stop
   else {
      if(rxCommands[0]&128){motionMotorBack(0);motionMotorForward(1);} // Forward
      else{motionMotorForward(0);motionMotorBack(1);}                  // Back

      switch(rxCommands[0]&112) {                          // Motion motor speed (PWM)
         case  16:   set_pwm1_duty(500);     break;
         case  32:   set_pwm1_duty(700);     break;
         case  48:   set_pwm1_duty(900);     break;
         case  64:   set_pwm1_duty(1023);    break;
      }
   }
}

//------------------------- Direction Motor Managing -------------------------------
void  comeZero(void) {
   char  i;
   switch(lastDirection) {
      case  zero: return;
      case  right:for(i=0;i<80;++i) {
                     turnLeft();
                     if(globalUARTsenser)return;
                     if(directionZero)break;
                  }
                  if(i>=80)setErr(DirectionEncoderDosnotAnswer);
                  break;
      case  left: for(i=0;i<80;++i) {
                     turnRight();
                     if(directionZero)break;
                  }
                  if(i>=80)setErr(DirectionEncoderDosnotAnswer);
                  break;
   }
}

void  turnLeft(void) {
   directionMotorLeft(1);
   directionMotorEn(1);
   if(smartDelay(10))globalUARTsenser=1;
   directionMotorEn(0);
   directionMotorLeft(0);
}

void  turnRight(void) {
   directionMotorRight(1);
   directionMotorEn(1);
   if(smartDelay(10))globalUARTsenser=1;
   directionMotorEn(0);
```

```
   directionMotorRight(0);
}

void  turnDirection(char   direction,char turnTimeLength) {
   char  i;
   if(direction==right) {
      for(i=0;i<turnTimeLength;++i) {
         turnRight();
      }
   }
   else {
      for(i=0;i<turnTimeLength;++i) {
         turnLeft();
      }
   }
}

void  interpretCommand1(char  command) {
   switch(command&0x03) {
      case  0: comeZero();lastDirection=zero;break;
      case  1: lastDirection=right;turnDirection(right,30);break;
      case  2: lastDirection=left;turnDirection(left,20);break;
      default:break;
   }

   if(command&4) {
      lineFollowMode();
   }
   else {
      if(modeIndicator==followMode) {
         motionMotorBack(0);
         motionMotorForward(0);
         comeZero();
         modeIndicator=normalMode;
      }
   }
   return;
}

void  interpretCommand2(char  command) {

}

void  interpretCommand3(char  command) {

}

void  interpretCommand4(char  command) {

}

void  interpretCommand5(char  command) {

}

void  interpretCommand6(char  command) {

}

void  interpretCommand7(char  command) {

}

void  interpretCommand8(char  command) {

}

void  interpretCommand9(char  command) {

}

void  commandSynthesis(void) {
   char  i;
   for(i=0;i<=9;++i) {
      switch(i) {
         case  0: interpretCommand0(rxCommands[i]);break;
         case  1: interpretCommand1(rxCommands[i]);break;
         case  2: interpretCommand2(rxCommands[i]);break;
         case  3: interpretCommand3(rxCommands[i]);break;
         case  4: interpretCommand4(rxCommands[i]);break;
```

```
         case  5: interpretCommand5(rxCommands[i]);break;
         case  6: interpretCommand6(rxCommands[i]);break;
         case  7: interpretCommand7(rxCommands[i]);break;
         case  8: interpretCommand8(rxCommands[i]);break;
         case  9: interpretCommand9(rxCommands[i]);break;
         default: break;
      }
   }
}


//------------------------- Step Motor Procedures ---------------------------------
#define  ForwardConst      16
#define  BackConst         128

void  stepForward(void) {
   char  i;
   char  virtualPort=ForwardConst;
   for(i=0;i<4;++i) {
      stepMotorPort|=virtualPort;
      virtualPort*=2;
      delay_ms(stepMotorTurningSpeed);
      stepMotorPort&=0x0F;
      delay_us(100);
   }
}

void  stepBack(void) {
   char  i;
   char  virtualPort=BackConst;
   for(i=0;i<4;++i) {
      stepMotorPort|=virtualPort;
      virtualPort/=2;
      delay_ms(stepMotorTurningSpeed);
      stepMotorPort&=0x0F;
      delay_us(100);
   }
}

void  cameraPosition(unsigned long  localPosition) {
   char  i;
   if(localPosition!=globalPositionKeeper)globalPositionKeeper=localPosition;
   else return;
   switch(localPosition) {
      case  0:    findZero();
                  break;
      case  45:   findZero();
                  for(i=0;i<7;++i)stepForward();
                  break;
      case  90:   findZero();
                  for(i=0;i<14;++i)stepForward();
                  break;
      case  135:  findZero();
                  for(i=0;i<21;++i)stepForward();
                  break;
      case  180:  findZero();
                  for(i=0;i<26;++i)stepForward();
                  break;
      case  225:  findZero();
                  for(i=0;i<32;++i)stepForward();
                  break;
      case  270:  findZero();
                  for(i=0;i<38;++i)stepForward();
                  break;
      case  315:  findZero();
                  for(i=0;i<46;++i)stepForward();
                  break;
      default: break;
   }
}

void  findZero(void) {
   char  i;
   for(i=0;i<90;++i) {
      stepBack();
      if(!zeroSwitch)break;
   }
   if(i>=90)setErr(DoesNotFindZeroSwitch);
}
//------------------------- Modem Connection Procedures ----------------------------
```

```
void  getRxCommands(void) {
   char  i;
   char  chrBuf;
   while(1) {
      while(!kbHit());
      chrBuf=getc();
      if(chrBuf=='F') { // End of array that '$RF' at the beginning of rxCommands
         for(i=0;i<=9;++i) {
            while(!kbhit());
            rxCommands[i]=getc();
         }
         delay_ms(80);  // wait for characters that 'END'+13+10 (received from modem)
         return;
      }
   }
}

void  sendModemConsts(char constID) {
   switch(constID) {
      case  begin:
                     delay_ms(1);
                     putc('$');
                     delay_ms(1);
                     putc('R');
                     delay_ms(1);
                     putc('F');
                     delay_ms(1);
                     break;
      case  end:
                     delay_ms(1);
                     putc('E');
                     delay_ms(1);
                     putc('N');
                     delay_ms(1);
                     putc('D');
                     delay_ms(1);
                     putc(13);
                     delay_ms(1);
                     putc(10);
                     delay_ms(1);
                     break;
   }
}

void  sendMyParameters(void) {
   char  i;
   modemConstsBegin;
   for(i=0;i<=9;++i) {
      putc(txCommands[i]);
   }
   modemConstsEnd;
   txCommands[2]=(0xFE&txCommands[2]); // Clear startup indicator bit
   setWarning(clrWarnMsgs);
   if(txCommands[0]) {                 // Check for all devices to stop (Error)
      stopAllHardware();
      while(1) {                       // Continiously runs for error until MCU reset
         for(i=0;i<3;++i) {
            setBuzzer(1);
            delay_ms(300);
            setBuzzer(0);
            delay_ms(300);
         }
         setBuzzer(1);
         delay_ms(1000);
         setBuzzer(0);
         delay_ms(500);
      }
   }
}

void  checkUartForRx(void) {
   while(!kbHit())
      lampsManaging();
}

void  startupSettings(void) {
   delay_ms(500);
   putc('A');
   findZero();              // Fixing camera position
   pc=64;
```

```
}

void  mcuInit(void) {
   setup_adc_ports(A_ANALOG);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DIV_BY_16,255,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_ccp1(CCP_PWM);
   pa=0;
   pb=0;
   pc=0;
   pd=0;
   pe=4;                      // Voltage to LDR's pin
   set_tris_a(0b11111111);
   set_tris_b(0b00000001);
   set_tris_c(0b00111010);
   set_tris_d(0b00000001);
   set_tris_e(0b11111000);
   delay_ms(100);
}
```

- main.h file (Saved in C:\COmputerControlledRobotCar\MCU_PIC18F452\main.h)

```
#include <18F452.h>
#device adc=8
#use delay(clock=40000000)
#fuses H4,NOWDT,NOLVP,NOBROWNOUT,NOPUT,NOWRT,NOCPD,PROTECT
#use rs232(baud=2400,parity=N,xmit=PIN_C0,rcv=PIN_C1,bits=8)
```

- picos.c file (In Second includes)

```
#ifdef PIC18F452
   #byte    pa=0xF80
   #byte    pb=0xF81
   #byte    pc=0xF82
   #byte    pd=0xF83
   #byte    pe=0xF84

//-------------------------- Prototypes --------------------------------------------
char  pa0(char    entryVar);
char  pa1(char    entryVar);
char  pa2(char    entryVar);
char  pa3(char    entryVar);
char  pa4(char    entryVar);
char  pa5(char    entryVar);
char  pb0(char    entryVar);
char  pb1(char    entryVar);
char  pb2(char    entryVar);
char  pb3(char    entryVar);
char  pb4(char    entryVar);
char  pb5(char    entryVar);
char  pb6(char    entryVar);
char  pb7(char    entryVar);
char  pc0(char    entryVar);
char  pc1(char    entryVar);
char  pc2(char    entryVar);
char  pc3(char    entryVar);
char  pc4(char    entryVar);
char  pc5(char    entryVar);
char  pc6(char    entryVar);
char  pc7(char    entryVar);
char  pd0(char    entryVar);
char  pd1(char    entryVar);
char  pd2(char    entryVar);
char  pd3(char    entryVar);
char  pd4(char    entryVar);
char  pd5(char    entryVar);
char  pd6(char    entryVar);
char  pd7(char    entryVar);
char  pe0(char    entryVar);
char  pe1(char    entryVar);
char  pe2(char    entryVar);

char  pa0(char    entryVar) {
```

```
   if(entryVar)pa|=1;
   else pa&=254;
   return   (pa&1);
}

char  pa1(char    entryVar) {
   if(entryVar)pa|=2;
   else pa&=253;
   return   (pa&2);
}

char  pa2(char    entryVar) {
   if(entryVar)pa|=4;
   else pa&=251;
   return   (pa&4);
}

char  pa3(char    entryVar) {
   if(entryVar)pa|=8;
   else pa&=247;
   return   (pa&8);
}

char  pa4(char    entryVar) {
   if(entryVar)pa|=16;
   else pa&=239;
   return   (pa&16);
}

char  pa5(char    entryVar) {
   if(entryVar)pa|=32;
   else pa&=223;
   return   (pa&32);
}

char  pb0(char    entryVar) {
   if(entryVar)pb|=1;
   else pb&=254;
   return   (pb&1);
}

char  pb1(char    entryVar) {
   if(entryVar)pb|=2;
   else pb&=253;
   return   (pb&2);
}

char  pb2(char    entryVar) {
   if(entryVar)pb|=4;
   else pb&=251;
   return   (pb&4);
}

char  pb3(char    entryVar) {
   if(entryVar)pb|=8;
   else pb&=247;
   return   (pb&8);
}

char  pb4(char    entryVar) {
   if(entryVar)pb|=16;
   else pb&=239;
   return   (pb&16);
}

char  pb5(char    entryVar) {
   if(entryVar)pb|=32;
   else pb&=223;
   return   (pb&32);
}

char  pb6(char    entryVar) {
   if(entryVar)pb|=64;
   else pb&=191;
   return   (pb&64);
}

char  pb7(char    entryVar) {
   if(entryVar)pb|=128;
   else pb&=127;
```

```
   return   (pb&128);
}

char  pc0(char    entryVar) {
   if(entryVar)pc|=1;
   else pc&=254;
   return   (pc&1);
}

char  pc1(char    entryVar) {
   if(entryVar)pc|=2;
   else pc&=253;
   return   (pc&2);
}

char  pc2(char    entryVar) {
   if(entryVar)pc|=4;
   else pc&=251;
   return   (pc&4);
}

char  pc3(char    entryVar) {
   if(entryVar)pc|=8;
   else pc&=247;
   return   (pc&8);
}

char  pc4(char    entryVar) {
   if(entryVar)pc|=16;
   else pc&=239;
   return   (pc&16);
}

char  pc5(char    entryVar) {
   if(entryVar)pc|=32;
   else pc&=223;
   return   (pc&32);
}

char  pc6(char    entryVar) {
   if(entryVar)pc|=64;
   else pc&=191;
   return   (pc&64);
}

char  pc7(char    entryVar) {
   if(entryVar)pc|=128;
   else pc&=127;
   return   (pc&128);
}

char  pd0(char    entryVar) {
   if(entryVar)pd|=1;
   else pd&=254;
   return   (pd&1);
}

char  pd1(char    entryVar) {
   if(entryVar)pd|=2;
   else pd&=253;
   return   (pd&2);
}

char  pd2(char    entryVar) {
   if(entryVar)pd|=4;
   else pd&=251;
   return   (pd&4);
}

char  pd3(char    entryVar) {
   if(entryVar)pd|=8;
   else pd&=247;
   return   (pd&8);
}

char  pd4(char    entryVar) {
   if(entryVar)pd|=16;
   else pd&=239;
   return   (pd&16);
}
```

```
char  pd5(char    entryVar) {
   if(entryVar)pd|=32;
   else pd&=223;
   return   (pd&32);
}

char  pd6(char    entryVar) {
   if(entryVar)pd|=64;
   else pd&=191;
   return   (pd&64);
}

char  pd7(char    entryVar) {
   if(entryVar)pd|=128;
   else pd&=127;
   return   (pd&128);
}

char  pe0(char    entryVar) {
   if(entryVar)pe|=1;
   else pe&=254;
   return   (pe&1);
}

char  pe1(char    entryVar) {
   if(entryVar)pe|=2;
   else pe&=253;
   return   (pe&2);
}

char  pe2(char    entryVar) {
   if(entryVar)pe|=4;
   else pe&=251;
   return   (pe&4);
}

#endif
```

APPENDIX B

The following Visual Basic programs have been installed to the computer in order to create a user interface and communication with robot car.

- MainForm.frm

```
Dim mainFormXor As Variant

Private Sub forwardButton_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    If (globalRecordVar = 1) Then
        timeVal = 0
        recTimer.Enabled = True
    End If
    SpeedValue = 1
    speedTimer.Enabled = True
    If ((txCommand(3) And 112) = 0) Then txCommand(3) = (txCommand(3) Or 16)
    txCommand(3) = (txCommand(3) Or 128)     ' Up key
    mainForm.carSenseTimer.Enabled = True
    sendStream
End Sub

Private Sub forwardButton_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then saveRoad "adv", timeVal
    speedLabel.Caption = "0"
    If (startupVar = 1) Then
        txCommand(3) = (txCommand(3) And 143)
        txCommand(4) = (txCommand(4) And 252)
        If (CameraAdjusting = 1) Then
            CameraAdjusting = 0
        Else
            sendStream
            forwardButton.Enabled = True
            backButton.Enabled = True
            rightButton.Enabled = True
            leftButton.Enabled = True
        End If
    End If
    sendStream
End Sub

Private Sub autoButton_Click()
    mainFormXor = (mainFormXor Xor 1)
    If (mainFormXor = 1) Then
        txCommand(4) = (txCommand(4) Or 4)
        forwardButton.Enabled = False
        backButton.Enabled = False
        leftButton.Enabled = False
        rightButton.Enabled = False
        Option1.Enabled = False
        Option2.Enabled = False
        Option3.Enabled = False
        Option4.Enabled = False
    Else
        txCommand(4) = (txCommand(4) And 251)
        forwardButton.Enabled = True
        backButton.Enabled = True
        leftButton.Enabled = True
        rightButton.Enabled = True
        Option1.Enabled = True
        Option2.Enabled = True
        Option3.Enabled = True
        Option4.Enabled = True
    End If
    sendStream
End Sub

Private Sub backButton_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then
        timeVal = 0
        recTimer.Enabled = True
    End If
```

```
    SpeedValue = 1
    speedTimer.Enabled = True
    If ((txCommand(3) And 112) = 0) Then txCommand(3) = (txCommand(3) Or 16)
    txCommand(3) = (txCommand(3) And 127)    ' this is down key
    mainForm.carSenseTimer.Enabled = True
    sendStream
End Sub

Private Sub backButton_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then saveRoad "back", timeVal
    speedLabel.Caption = "0"
    If (startupVar = 1) Then
        txCommand(3) = (txCommand(3) And 143)
        txCommand(4) = (txCommand(4) And 252)
        If (CameraAdjusting = 1) Then
            CameraAdjusting = 0
        Else
            sendStream
            forwardButton.Enabled = True
            backButton.Enabled = True
            rightButton.Enabled = True
            leftButton.Enabled = True
        End If
    End If
    sendStream
End Sub

Private Sub Command1_Click()

End Sub

Private Sub cameraPosTimer_Timer()
    txCommand(3) = (txCommand(3) And 248)
    txCommand(3) = (txCommand(3) Or HScroll1.Value)
    sendStream
    cameraPosTimer.Enabled = False
End Sub

Private Sub carSenseTimer_Timer()
    m = MsgBox("Model araç yanıt vermiyor..! Araç kapalı olabilir.", vbExclamation,
"Uyarı..!")
    carSenseTimer.Enabled = False
End Sub

Private Sub Command2_Click()
    globalRecordVar = 1
    saveRoad "StartRecord", 0
    autoButton.Enabled = False
    m = MsgBox("Yol kayıt modunda sadece mouse kullanılması gerekir..! Klavye kullanarak
kayıt yapamazsınız..! Ayrıca kayıt tamamlandıktan sonra mutlaka kayıt stop butonuna
basmanız gerekir..!", vbInformation, "HATIRLATMA")
End Sub

Private Sub Command3_Click()
    globalRecordVar = 0
    autoButton.Enabled = True
    saveRoad "StopRecord", 0
End Sub

Private Sub Command4_Click()
    Command2.Enabled = False
    Command3.Enabled = False
    Command4.Enabled = False
    autoButton.Enabled = False
    HScroll1.Enabled = False
    Option1.Enabled = False
    Option2.Enabled = False
    Option3.Enabled = False
    Option4.Enabled = False
    readTimer.Enabled = True
    readNextRecord "StartRead"
End Sub

Private Sub errFlashTimer_Timer()
    Static localXorVar As Variant
    Static localStrBuf As Variant
    If (errLabel.Caption = "---") Then Exit Sub
    If Not (errLabel.Caption = "Hata Yok") Then
        localXorVar = (localXorVar Xor 1)
```

```
        If ((localXorVar And 1) = 1) Then
            localStrBuf = errLabel.Caption
            errLabel.Caption = ""
        Else
            errLabel.Caption = localStrBuf
        End If
    End If
End Sub

Private Sub fastUsingDisTimer_Timer()

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    Dim keyRefVar As Variant
    If (mainFormXor = 1) Then
        itIsNotRealKey = 1
        Exit Sub
    End If
    keyRefVar = 0
    Select Case KeyCode
        Case 100: keyRefVar = 1   ' Left key
        Case 102: keyRefVar = 1   ' Right key
        Case 98: keyRefVar = 1    ' Forward Key on keyboard
        Case 104: keyRefVar = 1   ' Back key on keyboard
        Case 81: keyRefVar = 1    ' This is 'Q' key for changing camera position
        Case 87: keyRefVar = 1    ' Tihs is 'W' key for changing camera position
    End Select
    If (keyRefVar = 0) Then
        itIsNotRealKey = 1
    Else
        itIsNotRealKey = 0
    End If
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
    If (itIsNotRealKey = 1) Then Exit Sub
    If (globalPressVar = 0) Then
        globalPressVar = 1
        Option1.SetFocus
        If (startupVar = 1) Then
            Select Case KeyAscii
                Case 52:    SpeedValue = 1
                            speedTimer.Enabled = True
                            txCommand(4) = (txCommand(4) Or 2)  ' Left key
                            txCommand(3) = (txCommand(3) Or 128)
                            txCommand(3) = (txCommand(3) Or 16) ' Speed control bits
                            rightButton.Enabled = True
                            leftButton.Enabled = False
                            Select Case (txCommand(3) And 112)
                                Case 16:    Option1.Value = True
                                Case 32:    Option2.Value = True
                                Case 48:    Option3.Value = True
                                Case 64:    Option4.Value = True
                            End Select
                            sendStream
                            Exit Sub
                Case 54:    SpeedValue = 1
                            speedTimer.Enabled = True
                            txCommand(4) = (txCommand(4) Or 1)  ' Right key
                            txCommand(3) = (txCommand(3) Or 128)
                            txCommand(3) = (txCommand(3) Or 16) ' Speed control bits
                            rightButton.Enabled = False
                            leftButton.Enabled = True
                            Select Case (txCommand(3) And 112)
                                Case 16:    Option1.Value = True
                                Case 32:    Option2.Value = True
                                Case 48:    Option3.Value = True
                                Case 64:    Option4.Value = True
                            End Select
                            sendStream
                            Exit Sub

                Case 50:    SpeedValue = 1
                            speedTimer.Enabled = True
                            If ((txCommand(3) And 112) = 0) Then txCommand(3) =
(txCommand(3) Or 16)
                            txCommand(3) = (txCommand(3) And 127)    ' Down key
                            mainForm.carSenseTimer.Enabled = True
```

```
                Case 56:    SpeedValue = 1
                            speedTimer.Enabled = True
                            If ((txCommand(3) And 112) = 0) Then txCommand(3) =
(txCommand(3) Or 16)
                            txCommand(3) = (txCommand(3) Or 128)     ' Up key
                            mainForm.carSenseTimer.Enabled = True

                Case 81:    If Not (HScroll1.Value = 7) Then HScroll1.Value =
(HScroll1.Value + 1)
                            ' Q Key for camera position managing
                            CameraAdjusting = 1
                            Exit Sub
                Case 113:   If Not (HScroll1.Value = 7) Then HScroll1.Value =
(HScroll1.Value + 1)
                            ' q Key for camera position managing
                            CameraAdjusting = 1
                            Exit Sub

                Case 119:   If Not (HScroll1.Value = 0) Then HScroll1.Value =
(HScroll1.Value - 1)
                            ' w Key for camera position managing
                            CameraAdjusting = 1
                            Exit Sub

                Case 87:    If Not (HScroll1.Value = 0) Then HScroll1.Value =
(HScroll1.Value - 1)
                            ' W Key for camera position managing
                            CameraAdjusting = 1
                            Exit Sub

            End Select
        End If
        ' Automatical Speed Ranging
        With mainForm
            Select Case (txCommand(3) And 112)
                Case 16:    Option1.Value = True
                Case 32:    Option2.Value = True
                Case 48:    Option3.Value = True
                Case 64:    Option4.Value = True
            End Select
        End With
        ' Automatical forward and back indicating on direction buttons
        With mainForm
            If ((txCommand(3) And 128) = 0) Then
                forwardButton.Enabled = True
                backButton.Enabled = False
            Else
                forwardButton.Enabled = False
                backButton.Enabled = True
            End If
        End With
        sendStream
    End If
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    speedLabel.Caption = "0"
    If (itIsNotRealKey = 1) Then Exit Sub
    If (globalPressVar = 1) Then
        globalPressVar = 0
        If (startupVar = 1) Then
            txCommand(3) = (txCommand(3) And 143)
            txCommand(4) = (txCommand(4) And 252)
            If (CameraAdjusting = 1) Then
                CameraAdjusting = 0
            Else
                sendStream
                forwardButton.Enabled = True
                backButton.Enabled = True
                rightButton.Enabled = True
                leftButton.Enabled = True
            End If
        End If
    End If
End Sub

Private Sub Form_Load()
    init
    mainFormXor = 0
End Sub
```

```
Private Sub HScroll1_Change()
    If (startupVar = 1) Then
        cameraLabel.Caption = (HScroll1.Value * 45)
        animasyonPosition (HScroll1.Value)
        cameraPosTimer.Enabled = True
    End If
End Sub

Private Sub leftButton_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then
        timeVal = 0
        recTimer.Enabled = True
    End If
    SpeedValue = 1
    speedTimer.Enabled = True
    txCommand(4) = (txCommand(4) Or 2)   ' Left key
    txCommand(3) = (txCommand(3) Or 128)
    txCommand(3) = (txCommand(3) Or 16)  ' Speed control bits
    Select Case (txCommand(3) And 112)
        Case 16:    Option1.Value = True
        Case 32:    Option2.Value = True
        Case 48:    Option3.Value = True
        Case 64:    Option4.Value = True
    End Select
    sendStream
End Sub

Private Sub leftButton_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then saveRoad "left", timeVal
    speedLabel.Caption = "0"
    If (startupVar = 1) Then
        txCommand(3) = (txCommand(3) And 143)
        txCommand(4) = (txCommand(4) And 252)
        If (CameraAdjusting = 1) Then
            CameraAdjusting = 0
        Else
            sendStream
            forwardButton.Enabled = True
            backButton.Enabled = True
            rightButton.Enabled = True
            leftButton.Enabled = True
        End If
    End If
    sendStream
End Sub

Private Sub menuQUITitem_Click()
    End
End Sub

Private Sub MSComm1_OnComm()
    uartTimer.Enabled = True
End Sub

Private Sub Option1_Click()
    If Not ((txCommand(4) And 3) = 0) Then Option1.Value = True: Exit Sub
    If (startupVar = 1) Then
        SpeedValue = 1
        speedTimer.Enabled = True
        If ((txCommand(3) And 112) = 0) Then Exit Sub
        If Not ((txCommand(3) And 112) = 0) Then
            willBeSentReg = 1
        Else
            willBeSentReg = 0
        End If
        txCommand(3) = (txCommand(3) And 143)
        txCommand(3) = (txCommand(3) Or 16)
        If (willBeSentReg = 1) Then sendStream
    End If
End Sub

Private Sub Option2_Click()
    If Not ((txCommand(4) And 3) = 0) Then Option1.Value = True: Exit Sub
    If (startupVar = 1) Then
        SpeedValue = 2
        speedTimer.Enabled = True
        If ((txCommand(3) And 112) = 0) Then Exit Sub
```

```
        If Not ((txCommand(3) And 112) = 0) Then
            willBeSentReg = 1
        Else
            willBeSentReg = 0
        End If
        txCommand(3) = (txCommand(3) And 143)
        txCommand(3) = (txCommand(3) Or 32)
        If (willBeSentReg = 1) Then sendStream
    End If
End Sub

Private Sub Option3_Click()
    If Not ((txCommand(4) And 3) = 0) Then Option1.Value = True: Exit Sub
    If (startupVar = 1) Then
        SpeedValue = 3
        speedTimer.Enabled = True
        If ((txCommand(3) And 112) = 0) Then Exit Sub
        If Not ((txCommand(3) And 112) = 0) Then
            willBeSentReg = 1
        Else
            willBeSentReg = 0
        End If
        txCommand(3) = (txCommand(3) And 143)
        txCommand(3) = (txCommand(3) Or 48)
        If (willBeSentReg = 1) Then sendStream
    End If
End Sub

Private Sub Option4_Click()
    If Not ((txCommand(4) And 3) = 0) Then Option1.Value = True: Exit Sub
    If (startupVar = 1) Then
        SpeedValue = 4
        speedTimer.Enabled = True
        If ((txCommand(3) And 112) = 0) Then Exit Sub
        If Not ((txCommand(3) And 112) = 0) Then
            willBeSentReg = 1
        Else
            willBeSentReg = 0
        End If
        txCommand(3) = (txCommand(3) And 143)
        txCommand(3) = (txCommand(3) Or 64)
        If (willBeSentReg = 1) Then sendStream
    End If
End Sub

Private Sub Timer1_Timer()

End Sub

Private Sub readTimer_Timer()
    readNextRecord "noMessage"
    If (glbCommand = "EndOfCommand") Then
        readTimer.Enabled = False
        Command2.Enabled = True
        Command3.Enabled = True
        Command4.Enabled = True
        autoButton.Enabled = True
        HScroll1.Enabled = True
        forwardButton.Enabled = True
        backButton.Enabled = True
        leftButton.Enabled = True
        rightButton.Enabled = True
        Option1.Enabled = True
        Option2.Enabled = True
        Option3.Enabled = True
        Option4.Enabled = True
        m = MsgBox("Kayıtdan komut yürütülmesi başarıyla tamamlandı..", vbInformation,
"Kayıt Okuma Tamamlandı")
    Else
        glb_i = 0
        walkingTimer.Enabled = True
        readTimer.Enabled = False
    End If

End Sub

Private Sub recTimer_Timer()
    timeVal = (timeVal + 1)
End Sub
```

```
Private Sub rightButton_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then
        timeVal = 0
        recTimer.Enabled = True
    End If
    SpeedValue = 1
    speedTimer.Enabled = True
    txCommand(4) = (txCommand(4) Or 1)   ' Right key
    txCommand(3) = (txCommand(3) Or 128)
    txCommand(3) = (txCommand(3) Or 16)  ' Speed control bits
    Select Case (txCommand(3) And 112)
        Case 16:    Option1.Value = True
        Case 32:    Option2.Value = True
        Case 48:    Option3.Value = True
        Case 64:    Option4.Value = True
    End Select
    sendStream
End Sub

Private Sub rightButton_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    If (globalRecordVar = 1) Then saveRoad "right", timeVal
    speedLabel.Caption = "0"
    If (startupVar = 1) Then
        txCommand(3) = (txCommand(3) And 143)
        txCommand(4) = (txCommand(4) And 252)
        If (CameraAdjusting = 1) Then
            CameraAdjusting = 0
        Else
            sendStream
            forwardButton.Enabled = True
            backButton.Enabled = True
            rightButton.Enabled = True
            leftButton.Enabled = True
        End If
    End If
    sendStream
End Sub

Private Sub speedTimer_Timer()
    Select Case (SpeedValue)
        Case 1:    speedLabel.Caption = "1.3"
        Case 2:    speedLabel.Caption = "3.2"
        Case 3:    speedLabel.Caption = "4.1"
        Case 4:    speedLabel.Caption = "5.1"
    End Select
    speedTimer.Enabled = False
End Sub

Private Sub startupTimer_Timer()
    startupVar = 1
End Sub

Private Sub uartTimer_Timer()
    Dim strBuf As Variant
    Dim strBuf2 As Variant
    strBuf = MSComm1.Input
    If (Len(strBuf) < 2) Then Exit Sub
    If (strBuf = "") Then Exit Sub
    carSenseTimer.Enabled = False
    If (strBuf2 = 1) Then carHasStartup  ' Car startup detecting
    showBT2 (110 + (Int(Rnd * 3))) ' Battery 2 voltage
    showBT1 (110 + (Int(Rnd * 2))) ' Battery 1 voltage
    showTemp (24 + (Int(Rnd * 3))) ' Temperature sensor
    If Not (errLabel.Caption = "Hata Yok") Then
        m = MsgBox("Bir hata mesajı alındı..! Hata mesajları, model aracın restart
yapılmasını gerektirir..!", vbCritical, "UYARI..!")
    End If
    uartTimer.Enabled = False
End Sub

Private Sub walkingTimer_Timer()
    If (glb_i = 0) Then
        Select Case glbCommand
            Case "adv": SpeedValue = 1
                        speedTimer.Enabled = True
                        If ((txCommand(3) And 112) = 0) Then txCommand(3) =
(txCommand(3) Or 16)
                        txCommand(3) = (txCommand(3) Or 128)     ' Up key
```

```
                        forwardButton.Enabled = False
                        sendStream

            Case "back": SpeedValue = 1
                         speedTimer.Enabled = True
                         If ((txCommand(3) And 112) = 0) Then txCommand(3) =
(txCommand(3) Or 16)
                         txCommand(3) = (txCommand(3) And 127)    ' Down key
                         backButton.Enabled = False
                         sendStream

            Case "left": SpeedValue = 1
                         speedTimer.Enabled = True
                         txCommand(4) = (txCommand(4) Or 2)   ' Left key
                         txCommand(3) = (txCommand(3) Or 128)
                         txCommand(3) = (txCommand(3) Or 16)  ' Speed control bits
                         Select Case (txCommand(3) And 112)
                            Case 16:    Option1.Value = True
                            Case 32:    Option2.Value = True
                            Case 48:    Option3.Value = True
                            Case 64:    Option4.Value = True
                         End Select
                        leftButton.Enabled = False
                        sendStream

            Case "right": SpeedValue = 1
                          speedTimer.Enabled = True
                          txCommand(4) = (txCommand(4) Or 1)   ' Right key
                          txCommand(3) = (txCommand(3) Or 128)
                          txCommand(3) = (txCommand(3) Or 16)  ' Speed control bits
                          Select Case (txCommand(3) And 112)
                            Case 16:    Option1.Value = True
                            Case 32:    Option2.Value = True
                            Case 48:    Option3.Value = True
                            Case 64:    Option4.Value = True
                          End Select
                          rightButton.Enabled = False
                          sendStream
        End Select
    End If
    If (glb_i >= glbTime) Then
        If (globalRecordVar = 1) Then saveRoad "right", timeVal
        speedLabel.Caption = "0"
        If (startupVar = 1) Then
            txCommand(3) = (txCommand(3) And 143)
            txCommand(4) = (txCommand(4) And 252)
            If (CameraAdjusting = 1) Then
                CameraAdjusting = 0
            Else
                sendStream
                forwardButton.Enabled = True
                backButton.Enabled = True
                rightButton.Enabled = True
                leftButton.Enabled = True
            End If
        End If
        sendStream
        readTimer.Enabled = True
        walkingTimer.Enabled = False
        End If
    glb_i = (glb_i + 1)
End Sub
```

- Variables.bas

```
Public CameraAdjusting As Variant
Public globalPort As Variant
Public startupVar As Variant
Public globalPressVar As Variant
Public itIsNotRealKey As Variant
Public SpeedValue As Variant
Public timeVal As Variant
Public globalRecordVar As Variant
Public glbCommand As Variant
Public glbTime As Variant
Public glb_i As Variant
```

- Subs.bas

```
Public txCommand(18) As Byte

Sub init()
    ' Variables Init
    txCommand(0) = Asc("$")
    txCommand(1) = Asc("R")
    txCommand(2) = Asc("F")
    txCommand(3) = 0
    txCommand(4) = 0
    txCommand(5) = 0
    txCommand(6) = 0
    txCommand(7) = 0
    txCommand(8) = 0
    txCommand(9) = 0
    txCommand(10) = 0
    txCommand(11) = 0
    txCommand(12) = 0
    txCommand(13) = Asc("E")
    txCommand(14) = Asc("N")
    txCommand(15) = Asc("D")
    txCommand(16) = 13
    txCommand(17) = 10
    startupVar = 0
    globalRecordVar = 0
    CameraAdjusting = 0
    readParameters ' Parameters reading from text file
    mainForm.MSComm1.CommPort = globalPort
    mainForm.MSComm1.PortOpen = True
    mainForm.Option1.Value = True

End Sub

Sub readParameters()
    Open ("C:\Program Files\ComputerControlledRobotCarSoftware\mcParam.txt") For Input
As #1
        Input #1, globalPort
    Close #1
End Sub

Sub sendStream()
    Dim i As Byte
    For i = 0 To 17
        mainForm.MSComm1.Output = Chr(txCommand(i))
        Next i
End Sub

Sub animasyonPosition(positionData As Variant)
    With mainForm
        Select Case positionData
            Case 0: .L1.X1 = 3120
                    .L1.X2 = 3120
                    .L1.Y1 = 150
                    .L1.Y2 = 330
            Case 1: .L1.X1 = 3270
                    .L1.X2 = 3120
                    .L1.Y1 = 190
                    .L1.Y2 = 330
            Case 2: .L1.X1 = 3360
                    .L1.X2 = 3120
                    .L1.Y1 = 330
                    .L1.Y2 = 330
            Case 3: .L1.X1 = 3300
                    .L1.X2 = 3120
                    .L1.Y1 = 480
                    .L1.Y2 = 330
            Case 4: .L1.X1 = 3120
                    .L1.X2 = 3120
                    .L1.Y1 = 550
                    .L1.Y2 = 330
            Case 5: .L1.X1 = 3000
                    .L1.X2 = 3120
                    .L1.Y1 = 480
                    .L1.Y2 = 330
            Case 6: .L1.X1 = 2900
                    .L1.X2 = 3120
                    .L1.Y1 = 330
                    .L1.Y2 = 330
```

```
            Case 7: .L1.X1 = 2940
                    .L1.X2 = 3120
                    .L1.Y1 = 220
                    .L1.Y2 = 330
        End Select
    End With
End Sub

Function takeErrors(errID As Variant)
    Select Case errID
        Case 0: takeErrors = "Hata Yok"
                Exit Function
        Case 1: takeErrors = "Step Motor Sıfır Noktası Bulunamadı..!"
                Exit Function
        Case 2: takeErrors = "Direksiyon Motora ait Encoder Yanıt Vermiyor..!"
                Exit Function
        Case 3: takeErrors = "Çizgi izleme modu içerisinde çok geniş çizgi hatası..!"
                Exit Function
        Case 4: takeErrors = "Çizgi izleme modu içerisinde bir çizgi bulunamadı..!"
                Exit Function
    End Select
    takeErrors = "Geçersiz Haberleşme..!"
End Function

Function takeWarnings(warnID As Variant)
    Select Case warnID
        Case 0: takeWarnings = "Uyarı Yok"
    End Select
End Function

Sub carHasStartup()
    txCommand(3) = 0
    mainForm.HScroll1.Value = 0
    mainForm.cameraLabel.Caption = 0
    m = MsgBox("Model Araç şu anda çalıştırıldı..!", vbExclamation, "Uyarı")
End Sub

Sub showBT1(entryArg As Variant)
    mainForm.bt1Label.Caption = Mid((entryArg / 8.88), 1, 4)
End Sub

Sub showBT2(entryArg As Variant)
    mainForm.bt2Label.Caption = Mid((entryArg / 8.88), 1, 4)
End Sub

Sub showTemp(entryArg As Variant)
    mainForm.temperatureLabel.Caption = entryArg
End Sub

Sub saveRoad(keyID As Variant, timeVal As Variant)
    ' new file open
    If (keyID = "StartRecord") Then
        Kill ("c:\RobotCarRecord.txt")
        Open "c:\RobotCarRecord.txt" For Output As #1
    End If
        If (keyID = "StopRecord") Then
            Print #1, "EndOfRecord"
            Close #1
            Exit Sub
        End If
        If (keyID = "StartRecord") Then
            Print #1, "BeginOfRecord"
        Else
            Print #1, " " & keyID & "," & timeVal
        End If
End Sub

Sub readNextRecord(command As Variant)
    ' new file open
    If (command = "StartRead") Then
        Open "c:\RobotCarRecord.txt" For Input As #1
        Input #1, a
        Exit Sub
    End If
    Input #1, strBuf
    If (strBuf = "EndOfRecord") Then
        Close #1
        glbCommand = "EndOfCommand"
        Exit Sub
    End If
```

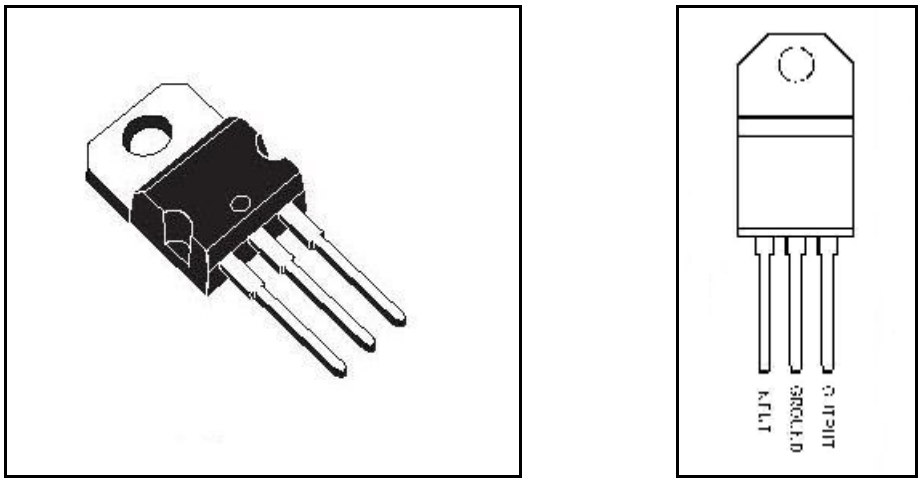
```
    seperateCommand strBuf
End Sub

Sub seperateCommand(stringArg As Variant)
    glbCommand = stringArg
    Input #1, glbTime
End Sub
```

APPENDIX C

C.1. L7800 Series Positive Voltage Regulator ICs

C.1.1 Description

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3 and D2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents. (STMicroelectronics 1, 2006)



A **B**

Figure C.1. L7800 series regulator ICs (a) Top view (b) Pin connection (STMicroelectronics 1, 2006)

C.1.2. Features

- Output current to 1.5A
- Output voltages of 5; 5.2; 6; 8; 8.5; 9; 10; 12;15; 18; 24V
- Thermal overload protection
- Short circuit protection
- Output transition SOA protection

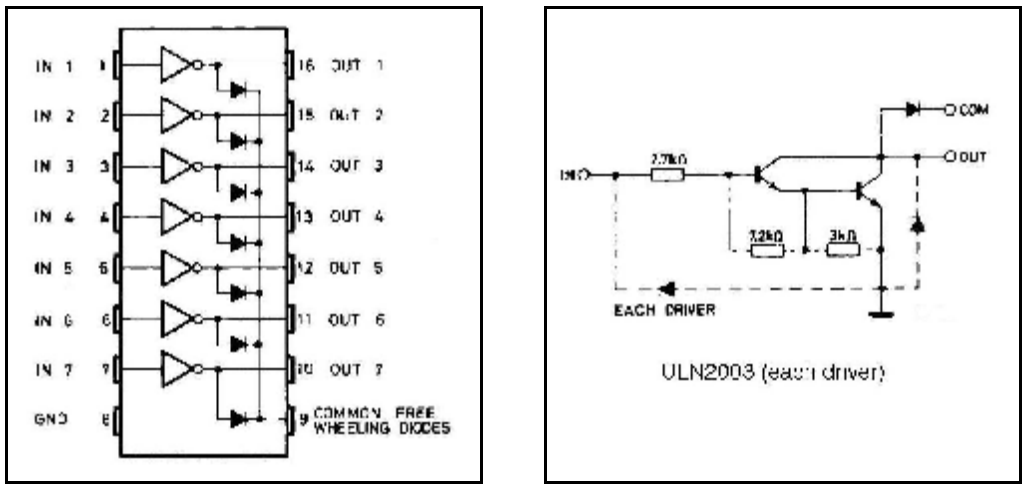
  (STMicroelectronics 1, 2006)

C.2. ULN 2003A Seven Darlington Array ICs

C.2.1 Description

The ULN2003 is high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout. These versatile devices are useful for driving a wide range of loads including solenoids, relays, DC motors, LED displays, filament lamps, thermal print heads and high power buffers. The ULN2003A is supplied in 16 pin plastic DIP packages with a copper lead frame to reduce thermal resistance. (STMicroelectronics 2, 2006)

The ULN2003A has series input resistors selected for operation directly with 5 V TTL or CMOS. These devices will handle numerous interface needs particularly those beyond the capabilities of standard logic buffers. The outputs will withstand at least 50 V in the OFF state. (Catalog of Electronic Components, 2004)



A **B**

Figure C.2. ULN2003A IC a) Top view b) Pin connection for each driver (STMicroelectronics 2, 2006)

C.2.2. Features

- Seven darlingtons per package
- Output current 500mA per driver (600mA peak)
- Output voltage 50V
- Integrated suppression diodes for inductive loads
- Outputs can be paralleled for higher current
- TTL/CMOS/PMOS/DTL Compatible inputs
- Inputs pinned opposite outputs to simplify layout

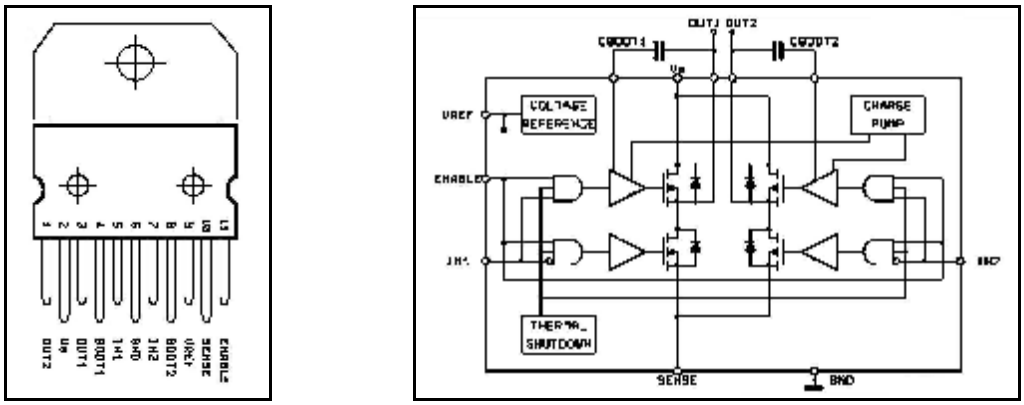
(STMicroelectronics 2, 2006)

C.3. L6203 Dmos Full Bridge Driver ICs

C.3.1 Description

The L6203 I.C. is a full bridge driver for motor control applications realized in Multi power-BCD technology which combines isolated DMOS power transistors with CMOS and Bipolar circuits on the same chip. By using mixed technology it has been possible to optimize the logic circuitry and the power stage to achieve the best

possible performance. The DMOS output transistors can operate at supply voltages up to 42V and efficiently at high switching speeds. All the logic inputs are TTL and CMOS compatible. Each channel (half-bridge) of the device is controlled by a separate logic input, while a common enable controls both channels.
(STMicroelectronics 3, 2006)



A **B**

Figure C.3. L6203 IC a) Top view b) Block diagram
(STMicroelectronics 3, 2006)

C.3.2. Features

- Supply voltage up to 48v
- 5A max peak current
- Total rms current up to 4A
- Rds (on) 0.3 w (typical value at 25 °c)
- Cross conduction protection
- TTL compatible drive
- Operating frequency up to 100 khz
- Thermal shutdown
- Internal logic supply
- High efficiency

(STMicroelectronics 1, 2006)

C.4. LM324 Low Power Quad Operational Amplifiers ICs

C.4.1. Description

These circuits consist of four independent, high gain, internally frequency compensated operational amplifiers. They operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. (STMicroelectronics 4, 2006)

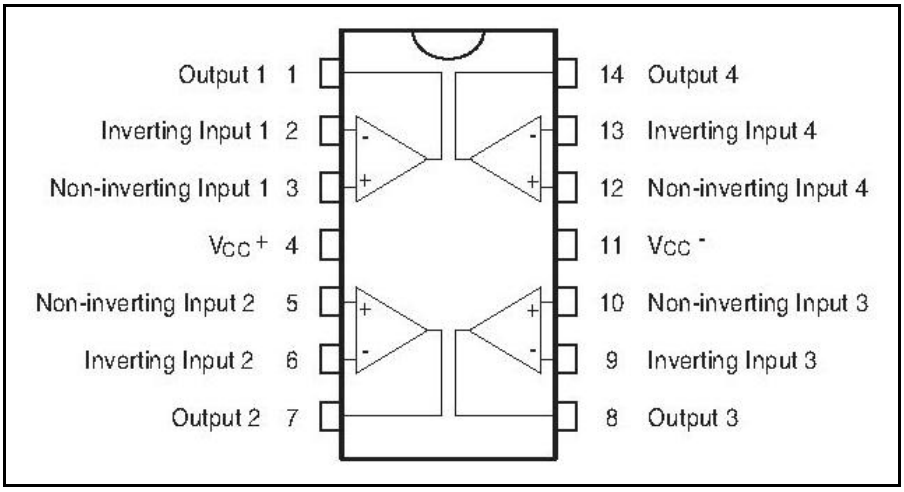


Figure C.4. LM 324 Pin configuration
(STMicroelectronics 4, 2006)

C.4.2. Features

- Wide gain bandwidth: 1.3 MHz
- Input common-mode voltage range includes ground
- Large voltage gain: 100 dB
- Very low supply current per amplifier: 375 µA
- Low input bias current: 20 nA
- Low input offset voltage: 5 mV max.
- Low input offset current: 2 nA
- Wide power supply range:

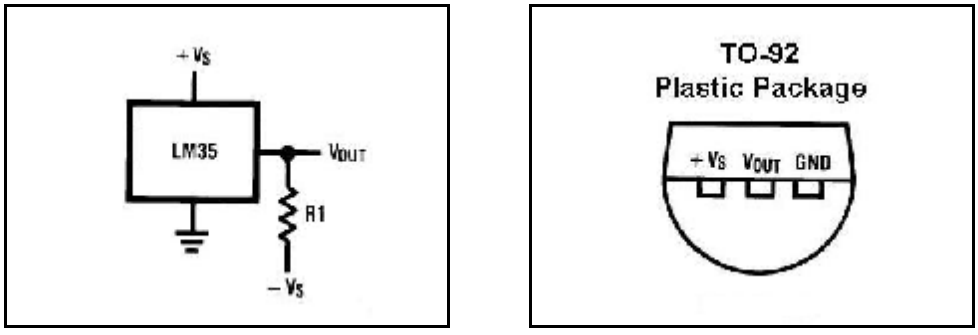
- Single supply: +3 V to +30 V
- Dual supplies: ±1.5 V to ±15 V

(STMicroelectronics 4, 2006)

C.5. LM35 Precision Centigrade Temperature Sensors

C.5.1. General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4°C at room temperature and ±3⁄4°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 Full-Range temperature sensor is rated to operate over a −55° to +150°C temperature range, and available in plastic TO-92 transistor packages. (National Semiconductors, 2006)



A **B**

Figure C.5. LM35 IC a) Centigrade Temperature Sensor b) Bottom view (National Semiconductors, 2006)

C.5.2. Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 20 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±1⁄4°C typical
- Low impedance output, 0.1 W for 1 mA load

(National Semiconductors, 2006)

C.5.3. Typical Applications

R1 resistor seen in Figure C.5 (a) is calculated as follows;

Choose $R_1 = -V_S / 50\ \mu A$ (National Semiconductors, 2006)

Vout = +1,500 mV at +150°C

= +250 mV at +25°C

= −550 mV at −55°C

C.6. UDEA Modem Module (UFM-A12 WPA)

C.6.1 Features

- 868 Mhz or 915 Mhz UHF band.
- High frequency stability
- Ideal for long range application with main power
- Easy use with embedded protocol

C.6.2. Applications

- Remote Control System
- Telemetry system
- Social alarm system
- Security alarm system
- Paging system

The following figure is showing the dimensions of UFM-A12 WPA modem module.

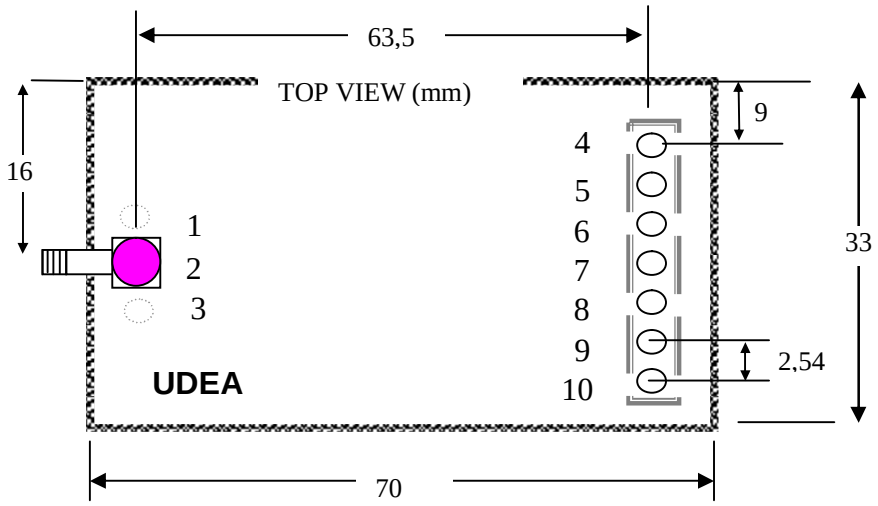


Figure C.6. Dimensions
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.3. General Descriptions

The UFM-A12 WPA UHF FSK data transceiver modem module is developed to cover a band plan ERC Recommendation on Short Range Device (SRD) in the range of 868 MHz ISM band.

The UFM-A12 WPA is designed for PCB mounting. A simple wire can be soldered to the antenna in put or external antenna can be used.

Pin assignments and technical features are as fallows.

Table C.1. Pin assignments of modem module
(UFM-A12 WPA Modem Module Operation Guide, 2005)

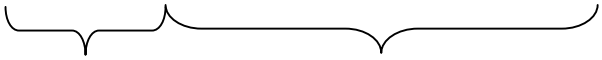
Pin No	Pin-Name	Input/Output	Description	
1,3,4	**GND**		**Connection to GND.**	
2	**ANT**		**Antenna connection.**	
5	**NC**			
6	**TX**	**O**	**TX - UART**	(3 VDC TTL)
7	**RX**	**I**	**RX - UART**	(3 VDC TTL)
8	**NC**	**-**		
9	**Vcc - 1**	**-**	**+3,3 – 5 VDC**	**use regulated voltage source. (**max 40mA)
10	**Vcc - 2**		**+3 VDC**	**use regulated voltage source.** (max 600mA)

Table C.2. Technical features of modem module
(UFM-A12 WPA Modem Module Operation Guide, 2005)

	Min.	Typ.	Max	Unit	Not
Voltage supply Vcc - 1	**3,3**		**5**	**Vdc1**	**use regulated voltage source. ±100 mV**
Voltage supply Vcc - 2		**3**		**Vdc2**	**use regulated voltage source. ±100 mV**
Supply current TX mod		**TBD**		**mA**	
Supply current RX mod		**TBD**		**mA**	
Logic "0" DI volt	**0**		**0.6**	**Vdc1**	
Logic "1" DI volt	**3,5**		**5**	**Vdc1**	
Logic "0" DO volt	**0**		**0.6**	**Vdc1**	
Logic "1" DO volt	**3,5**		**5**	**Vdc1**	
Working Temperature	**-20**		**+55**	**°C**	**ETSI 300 220**
Storage Temperature	**-50**		**+150**	**°C**	
Dimensions	**70 X 33 X 8 mm**				
RF Specifications					
RF Sensitivity		**-120**	**-123**	**dBm**	**2.4 kBaud**
Bandwidth		**7.5**		**kHz**	**2.4 kBaud**
Data Rate		**4.8**		**Kbps**	**Manchester**
Output Power		**23**	**27**	**dBm**	**@868MHz @915MHz**

C.6.4. MAC Command Frame

$	C	DATA
24h	43h	Komut



Start of Frame **Data**

Figure C.7. Command Frame Structure
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.5. Start Up

When UFM-A12 is powered,it send ready information “B” to microcontroller

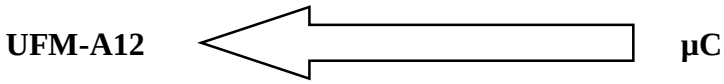
UFM-A12 **µC**

$	C	B
24h	43h	42h

Figure C.8. Frame structure of startup
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.6. RF Channel Selection

The UFM-A12 has 2 different RF channel in the band. User can select the communication channel for any reason. If the user would like to change the RF communication channel, channel selection command should be send a as shown in Figure 4.9. (UFM-A12 WPA Modem Module Operation Guide, 2005)

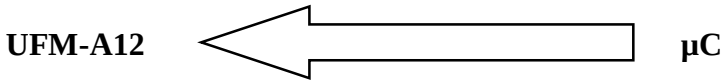


$	C	Kanal No (1- 2)
24h	43h	31h // 32

Figure C.9. RF Channel Selection
(UFM-A12 WPA Modem Module Operation Guide, 2005)

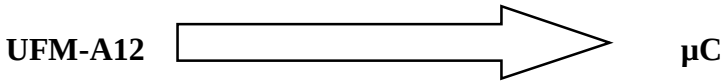
C.6.7. RSSI Level

UFM-A12 has a built-in RSSI (Received Signal Strength Inducator) giving a digital value. The digital RSSI value is ranging from 0 to 67. (UFM-A12 WPA Modem Module Operation Guide, 2005)



$	C	R
24h	43h	52h

Figure C.10. RSSI Request
(UFM-A12 WPA Modem Module Operation Guide, 2005)

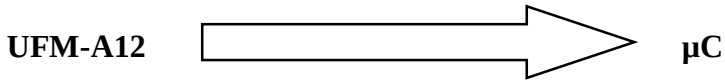


$	C	RSSI Level (10-67)
24h	43h	0Ah // 43h

Figure C.11. RSSI Level Response
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.8. Overflow

If the user give more then 72 byte to the UFM-A12WPA, the module give a overflow error massage to output as shown in Figure 4.12. (UFM-A12 WPA Modem Module Operation Guide, 2005)

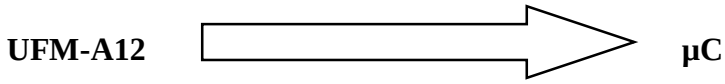


$	C	O
24h	43h	4Fh

Figure C.12. Frame structure of overflow
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.9. TimeOut

If the user give a delay more then 500 msec. between two input data, the module give an timeout error massage to output as shown in Figure 4.13. (UFM-A12 WPA Modem Module Operation Guide, 2005)



$	C	T
24h	43h	54h

Figure C.13. Frame structure of timeout
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.6.10. Error

If the user do not give the data as described in 4.3.4. Data Input to UFM-A12, the module give an error massage to output as shown in Figure 4.14. (UFM-A12 WPA Modem Module Operation Guide, 2005)

UFM-A12 → **µC**

$	C	E
24h	43h	45h

Figure C.14. Frame structure of Error
(UFM-A12 WPA Modem Module Operation Guide, 2005)

C.7. Wireless Video and Audio Unit (JMK - WS-309AS)

Figure C.15. Camera Transmitter and Receiver Unit
(JMK Shenzhen Jiameikang Science 1, 2003)

The technical parameters of Transmitter and Receiver Units are as follows**,**

C.7.1. Technical Parameters of Transmitting Unit:

- Video Camera Parts: 1/3" 1/4" Image Sensors
- System: PAL/CCIR NTSC/EIA
- Effective Pixel: PAL: 628X582 NTSC: 510X492
- Image Area: PAL: 5.78X4.19mm NTSC: 4.69X3.45mm
- Horizontal Definition:380 TV Lines
- Scanning Frequency: PAL/CCIR: 50HZ NTSC/EIA: 60HZ

- Minimum Illumination: 3LUX
- Sensitivity: +18DB-AGL ON-OFF
- Output Electrical Level: 50mW
- Output Frequency:1.2G/2.4G
- Transmission Signal: Video, Audio
- Linear Transmission Distance:50-100m
- Voltage: DC +9V
- Current: 300mA
- Power Dissipation:≤640mW

  (JMK Shenzhen Jiameikang Science 1, 2003)

C.7.2. Technical Parameters of Receiving Unit:

- Wireless Audio Receiver
- Receiving Method: Electronic Frequency Modulation
- Reception Sensitivity:+18DB
- Receiving Frequency:1.2G/2.4G
- Receiving Signal:Video,Audio
- Voltage:DC 12V
- Current:500mA

  (JMK Shenzhen Jiameikang Science 1, 2003)

C.8. DVR CARD

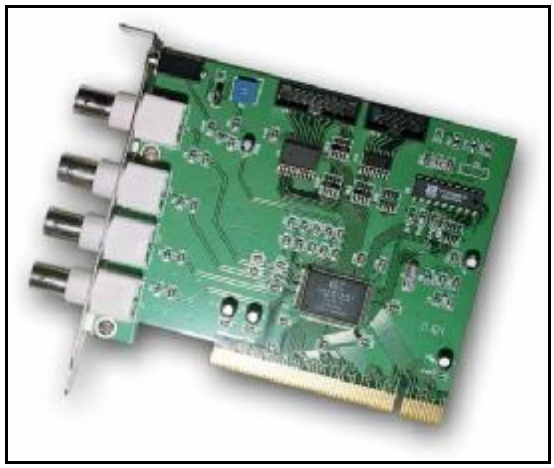


Figure C.16. DVR Card
(JMK Shenzhen Jiameikang Science 2, 2003)

The DVR Card's specifications are as follows;

C.8.1. Basic function:

- 1-4 channels video input
- 1 channel audio input
- 25 frames, mpeg4 software compression
- 150-200M /hour. Use pico software
- Support PTZ control and remote control
- Can build 4ch realtime or 16ch unrealtime system.
  (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.2. Features:

- 4-Channel in 1 card, audio and video real-time synchronous
- Adopt MJPEG-4 compression mode
- Network Function: Support PSTN, ISDN, ADSL, LAN and Internet connections
- 4-channel video per card (inc audio)
- Total Resource at 25F/S
- Provide 16-channel recording, 1 system supports 4 cards

- Resolution: 768*576 (PAL), 640*480 (NTSC)
- Support multi-level P/T/Z control through LAN/WAN
- Fully compatible with P/T/Z control protocols of industry mainstream KALATEL, PELCO, PHILIPS, UNIVISION, ULTRAK, VICON encoders (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.3. Technical Specification:

- Compression mode: MPEG-4 compression mode
- Resolution: 760*576 (PAL)
- 1-Channel in 1 card or 4-Channel in 1 card
- Resource occupancy at 120-150M/H per channel (inc audio)
- Alarm Control: Able to connect 16-channel normal alarm annunciator (infrared probe, vibration sensor, etc)
- Auto-linkage alarm recording once alarm sensor triggers
- Alarm recording time adjustable
- Provide 3-channel alarm output, able to linkage control lamplight, alarm bell and other peripherals
- Each channel able to set video motion detection alarm function and also able to set motion detection area at discretion
- Auto-linkage alarm video recording once motion detection sensor triggers
- Provide video signal lost and image dodge alarm
  (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.4. Remote Transmission & Control:

- Perform remote image transmission through LAN, normal telephone line, ISDN and other communication modes
- Support remote video recording data playback, P/T/Z control and client alarm functions
  (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.5. Audio/Video Recording & Playback:

- MPEG4 compression format, able to implement 1-channel~4-channel real-time recording (25F/S per channel, total resource at 25-100 frames)
- Image resolution at 400-line (much higher than 240-line of normal video recorder)
- Multi recording mode time schedules recording, real-time recording and alarm recording
- Recording frame rate per channel adjustable (1/60F/S-25F/S)
- Alarm recording: Auto-turn into real-time recording mode once alarm recording triggers
- Voice recording with audio and video real-time synchronous, and support scene listen-in
- Able to save recording data into local harddisk, or use U-disk/CD-recorder for long time storage
- Able to perform selective playback modes according to recording time, such as different speed playback, fast forward / fast reverse
- Able to capture single frame image for printing or electronic zoom in
- Support recording and playback real-time synchronous

  (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.6. System Require:

- Display Card: Geforce 4 or ATI R70000 64M or more
- Memory:256M or more
- CPU: PIII 800MHz or more
- Operate System: Windows
- Display model:1024*768 32bit
- Install Directx 8.0 or more advanced

  (JMK Shenzhen Jiameikang Science 2, 2003)

C.8.7. Main connection:

- Computer→Video Capture Card→BNC→Video Data Cable→Receiver