AUTOMATED CAR PARKING SYSTEM

Project Report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Utkarsh Sharma (121028)

Gaurav Gupta (121029)

Akash Garg (121030)

UNDER THE GUIDANCE OF

Mr. Pardeep Garg

Assistant Professor
(ECE Department)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
SOLAN – 173 234, HIMACHAL PRADESH INDIA
June- 2016

TABLE OF CONTENTS

DECLARATION

We hereby declare that the work reported in the B-Tech thesis entitled **"Automated Car Parking System "** submitted at **Jaypee University of Information Technology, Waknaghat , India,** is an authentic record of our work carried out under the supervision of **Mr. Pardeep Garg**. We have not submitted this work elsewhere for any other degree or diploma.

Utkarsh Sharma Gaurav Gupta Akash Garg

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT, SOLAN, H.P

Date:

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B-Tech. thesis entitled **"Automated Car Parking System"**, submitted by **Utkarsh Sharma, Gaurav Gupta and Akash Garg** at **Jaypee University of Information Technology, Waknaghat, India** is a bonafide record of their original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

Mr. Pardeep Garg

Assistant Professor

Electronics and Communication Engineering

JUIT Waknaghat

Date:

ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who have helped and supported us during this project. Our deepest thanks to **Mr. Pardeep Garg (Assistant Professor)**, our project guide for his exemplary guidance, monitoring and constant encouragement throughout the course of this project work. He has taken great pains to go through the project and make necessary corrections as and whenever needed. We are also grateful to **Mr. Mohan Sharma & Mr. Pramod Kumar (ECE Project lab**) for their practical help and guidance. We would also like to thank all the faculty members of ECE department without whom the progress of this project would have been a distant reality. We also extend our heartfelt thanks to our family and well-wishers.

Name of the students:
Utkarsh Sharma (121028)
Gaurav Gupta (121029)
Akash Garg (121030)

Date:

LIST OF ACRONYMS & ABBREVIATIONS

1. APS – Automated (car) Parking System
2. LCD – Liquid Crystal Display
3. LED – Light Emitting Diode
4. AVR – Advanced Virtual RISC
5. ISIS –Intelligent Schematic Input System
6. IDE – Integrated Development Environment
7. VSM – Virtual System Modelling
8. RISC – Reduced Instruction Set Computer
9. IR – Infrared
10. PCB – Printed Circuit Board
11. GSM – Global System for Mobile Communication
12. OTP – One Time Password

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This report presents the idea and implementation of automated car parking system. Automation or automatic control, is the use of various systems for operating equipments or objects with minimal or reduced human intervention. The biggest benefit of automation is that it saves labour, however, it is also used to save energy and materials and to improve quality, accuracy and precision.

In our project we are basically designing a parking system where one has to just enter the details of his car (like car registration number) at entry gate and then leave his car there and the car will automatically get parked at the available parking slot. If the parking area is full, a warning message of “Parking is full!” will be displayed. In this way that person will be able to save his time and energy.

We have divided our project into four parts i.e. planning part, software implementation, research paper publication and hardware implementation. First of all we planned layout of our project. Then we implemented our project on software. For software implementation we used Atmel AVR Studio (for writing code in embedded C) and ISIS professional proteus (for circuit simulation). After successful implementation of our project on software we wrote a research paper on our idea which is got published in an International Journal (IJIREEICE) March 2016 issue. Finally we have also tested our project on hardware and we got positive results on it.

CHAPTER 1

INTRODUCTION

Nowadays in many multiplex systems there is a severe problem for car parking systems. There are many lanes for car parking, so to park a car one has to look for the all lanes. Conventionally, car parking systems does not have any intelligent monitoring system. Parking lots are monitored by human beings. All vehicles enter into the parking and waste time for searching for parking slot. Sometimes it creates blockage. Condition become worse when there are multiple parking lanes and each lane have multiple parking slots. Moreover there is a lot of men labour involved for this process for which there is lot of investment. So the need is to develop a system where person leaves his car in parking entry and the car should automatically get parked. Use of automated system for car parking monitoring will reduce the human efforts.

The concept for the automated parking system is driven by two factors: a need for parking spaces and a scarcity of available land.

1.1 LITERATURE WORK

Lot of research has been done in the field of automated car parking system to provide best alternatives for easy, safe and quick car parking. One such idea was to read the registration number of the car from car number plate using image processing technique [1] and then using this car registration number finding out car entry time, car exit time and also time period for which car was parked which will help in collecting parking fee.

A automated car parking system was designed using sensors which works in three phases [2]. First phase was scanning phase in which the parking area is examined by the ultrasonic sensors mounted on the robot-car and a path is produced if the space is sufficient. Second phase was positioning phase in which the robot reverses to the edge of the parking space avoiding any collision. Finally in third phase i.e. manoeuvring phase, the robot moves to the parking position in the parking space in a unified pattern.

One similar idea was presented which uses three ultrasonic sensors mounted at the front left corner of the car to obtain the information of surrounding area of parking lot and a compass sensor mounted at the centre of the car to measure the posture of the car [3].These four sensors help in deciding the turning angle of the car.

Another idea proposed was a design of an automated car parking system controlled by an android application [4] that controls the number of cars to be parked/un-parked within a parking area with the help of an Android Application.

Some smart parking systems were also embedded with technologies like GSM modules [5-6] for SMS services to provide more secure and user friendly parking system.

Few other researches were also done using nonholonomic vehicles [7] and soft-computing techniques [8].

Thus, our aim was to propose an automatic parking system in context of parking which are dependent on car drivers which will not only save time, money and energy but also overcome the limitations of such similar pre-existing systems.

1.2 BENEFITS OF AUTOMATED PARKING SYSTEMS (APS):

- Havoc and waiting during parking is reduced.
- Boon for old-age as well as novice drivers who normally experience biggest difficulty while parking vehicles.
- Driving around in search of a parking space is eliminated, thereby reducing engine emissions.
- All APS take advantage of a common concept to decrease the area of parking spaces - removing the driver and passengers from the car before it is parked.
- Ceiling height is minimized since there is no pedestrian traffic (drivers and passengers) in the parking area.
- No walkways, stairways or elevators are needed to accommodate pedestrians in the parking area.
- The parked cars and their contents are more secure since there is no public access to parked cars.
- Only minimal ventilation and lighting systems are needed.
- APS theoretically eliminates the need for parking attendants.

Comparision of manual and automatic parking:

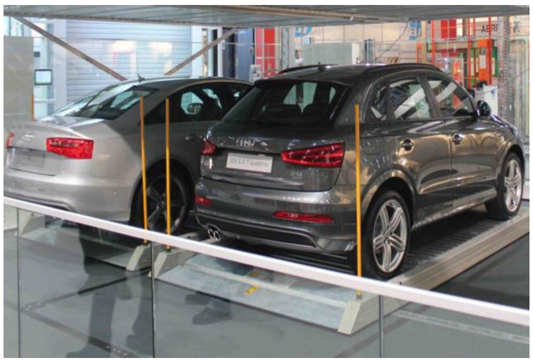


Figure 1.1: Automatic Parking[16]

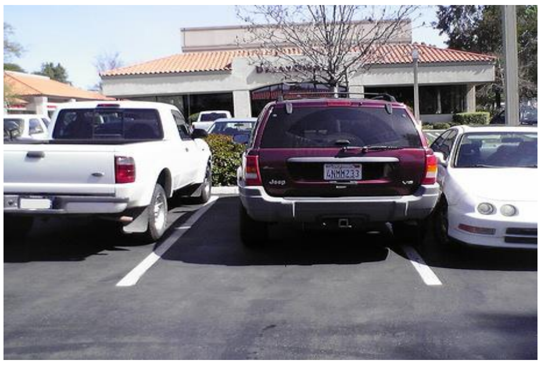


Figure 1.2: Manual Parking[17]

This depicts that automatic parking is compact whereas manual parking is full of glitches many times which consumes lot of space.

1.3 DISADVANTAGES OF AUTOMATED PARKING SYSTEMS (APS):

- It may be costly due to involvement of complex devices for automation.
- If the sensing devices go out-of-order or non-functional then there may be great difficulty in getting cars out of parking, more over it may cause accidents inside parking also.

CHAPTER 2

PROJECT REVIEW AND BACKGROUND MATERIAL

2.1 PROJECT REVIEW:

This project has two parts. First part is to make a smart parking area and second part is to make a smart car. As the car will arrive at the entry gate of the parking area the driver will come out of the car and then he will enter his car’s registration number at that entry gate. As he enters the car registration number a slot number corresponding to that car number will be automatically allocated to him. This allotted slot number will appear on LCD screen of parking area. If parking area is full than a warning message showing “Parking is full!” will be displayed on LCD screen. Now the driver will enter the alloted slot number into his smart car using a keypad embedded on his car and after that the driver may leave for his work. The car will automatically enter the parking area and will get parked to its allocated slot. Also as the car enters the parking area, a timer will automatically start.

2.2 SOFTWARE USED:

2.2.1 AVR STUDIO 4

Atmel AVR Studio is the Integrated Development Environment (IDE) for developing and debugging embedded Atmel AVR applications. The AVR Studio IDE gives you a seamless and easy-to-use environment to write, build, and debug your C/C++ and assembler code. We can burn this code on hardware circuit microcontroller using programmer to make real time applications. We are using AVR Studio 4 for our C codes. It supports more than 300 Atmel AVR and Atmel smart ARM-based devices.It has in-system programming and debugging provides interface to all Atmel in-circuit programmers and debuggers.

2.2.2 CV AVR

Code vision AVR (CV AVR) compiler is a C cross-compiler, Integrated Development Environment (IDE) and Automatic Program Generator designed for the Atmel microcontroller series made AVR. Code vision AVR can be run on Operating System Windows 95, 98, Me, NT4, 2000, XP and 7. C cross-compiler is able to translate almost all orders of ANSI C language, to the extent permitted by the architecture of the AVR, with the addition of some special features to take advantage of the AVR architecture and the needs of the embedded system. It can be used for debugging purposes at the level of C.

2.2.3 ISIS PROFESSIONAL PROTEUS

Proteus 7.0 is a Virtual System Modelling (VSM) that combines circuit simulation, animated components and microprocessor models to co-simulate the complete microcontroller based designs. This is the perfect tool for engineers to test their microcontroller designs before constructing a physical prototype in real time. This program allows users to interact with the design using on-screen indicators and/or LED and LCD displays and, if attached to the PC, switches and buttons. We are using version 7.9 for our simulations.

The ISIS, Intelligent Schematic Input System is the environment for the design and simulation of electronic circuits. The component library includes claims to more than 10,000 circuit components with 6000 Pro spice simulation models. Own components can be created and added to the library.

The VSM, Virtual System Modelling, provides a graphical SPICE circuit simulation and animation directly in the ISIS environment. The SPICE simulator is based on the Berkeley SPICE3F5 model.

It may microprocessor-based systems are simulated. With the VSM-Engine can interact during the simulation directly to the circuit. Changes of buttons, switches or potentiometers are queried in real time as well as LED indicators, LCD display and Hot / Cold -Wires displayed.

The microcontrollers are in the periphery and in the code fully supported (interrupt, ADC, I2C, USB, comparators, etc.). It includes a debugging environment for the program code of the microcontroller. To simulate the .HEX and .COF file of the compiled software are necessary. The clock is simulated in real time.

2.3 HARDWARE COMPONENTS USED:

2.3.1 MICROCONTROLLER (ATMEL AVR, ATMEGA16)

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATmega16 provides the following features: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for boundary scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes.

The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel’s high density non-volatile memory technology. The on-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications. The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

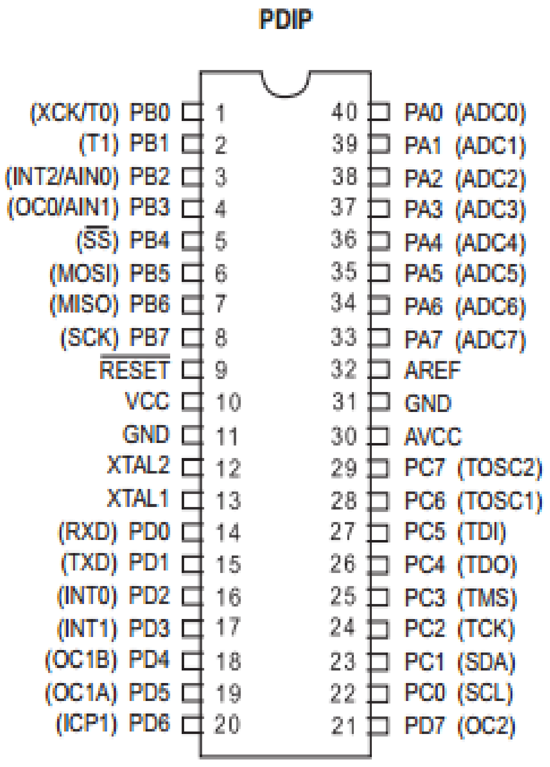


Figure 2.1: Pinout of ATmega16 [18]

PIN DESCRIPTIONS:

VCC -Digital supply voltage.

GND -Ground.

Port A (PA7..PA0) -Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0) -Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running**.**

Port C (PC7..PC0) –Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface.

Port D (PD7..PD0) -Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega16.

RESET -Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

XTAL1 -Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 -Output from the inverting Oscillator amplifier.

AVCC -AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

AREF -AREF is the analog reference pin for the A/D Converter.

FEATURES:

- High-performance, Low-power 8-bit Microcontroller.
- Advanced RISC Architecture - 131 Powerful Instructions, Most Single Most Single-clock Cycle Execution clock Cycle Execution, 32 x 8 General Purpose Working Registers, Fully Static Operation, Up to 16 MIPS Throughput at 16 MHz, On-chip 2-cycle Multiplier.
- High Endurance Non-volatile Memory segments - 16K Bytes of In-System Self-programmable Flash program memory, 512 Bytes EEPROM, 1K Byte Internal SRAM, Write/Erase Cycles: 10,000 Flash/100,000 EEPROM, Data retention: 20 years at 85°C/100 at 25°C(1), Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation, Programming Lock for Software Security.
- JTAG Interface, Boundary-scan Capabilities According to the JTAG Standard, Extensive on-chip debug support, Programming of Flash, EEPROM, fuses, and lock bits through the JTAG Interface.
- Peripheral Features, two 8-bit Timer/Counters with separate prescalers and Compare Modes, one 16-bit timer/counter with separate prescaler, Compare mode, and Capture mode, Real time counter with separate oscillator, four PWM channels, 8-channel, 10-bit ADC 8 single-ended channels, 7 differential channels in TQFP package only 2 differential channels with programmable gain at 1x, 10x, or 200x, Byte-oriented Two-wire serial interface, Programmable serial USART, Master/Slave SPI serial interface, programmable watchdog timer with separate on-chip oscillator, on-chip analog comparator.
- Special Microcontroller Features – power-on reset and programmable brown-out detection, internal calibrated RC oscillator, external and internal interrupt sources, six

sleep modes: idle, ADC noise reduction, power-save, power-down, standby and extended standby.

2.3.2 L293D MOTOR DRIVER IC

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor. In a single l293d chip there two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It’s like a switch.

There are 4 input pins for L293D, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as logic 0 or logic 1.

VCC is the voltage that it needs for its own internal operation 5v; L293D will not use this voltage for driving the motor. For driving the motors it has a separate provision to provide motor supply VSS (V supply). L293d will use this to drive the motor. It means if you want to operate a motor at 9V then we need to provide a Supply of 9V across VSS Motor supply.

The maximum voltage for VSS motor supply is 36V. It can supply a max current of 600mA per channel.Since it can drive motors Up to 36V hence you can drive pretty big motors with this L293D.

VCC pin 16 is the voltage for its own internal Operation. The maximum voltage ranges from 5V and upto 36V.

Motor driver is basically a current amplifier which takes a low-current signal from the microcontroller and gives out a proportionally higher current signal which can control and drive a motor. In most cases, a transistor can act as a switch and perform this task which drives the motor in a single direction.

L293D IC generally comes as a standard 16-pin DIP (dual-in line package). This motor driver IC can simultaneously control two small motors in either direction; forward and reverse with just 4 microcontroller pins (if you do not use enable pins). Some of the features (and drawbacks) of this IC are:

- Output current capability is limited to 600mA per channel with peak output current limited to 1.2A (non-repetitive). This means you cannot drive bigger motors with this IC. However, most small motors used in hobby robotics should work. If you are unsure whether the IC can handle a particular motor, connect the IC to its circuit and run the motor with your finger on the IC. If it gets really hot, then beware... Also note the words "non-repetitive"; if the current output repeatedly reaches 1.2A, it might destroy the drive transistors.

- Supply voltage can be as large as 36 Volts. This means you do not have to worry much about voltage regulation.

- L293D has an enable facility which helps you enable the IC output pins. If an enable pin is set to logic high, then state of the inputs match the state of the outputs. If you pull this low, then the outputs will be turned off regardless of the input states

- The datasheet also mentions an "over temperature protection" built into the IC. This means an internal sensor senses its internal temperature and stops driving the motors if the temperature crosses a set point
- Another major feature of L293D is its internal clamp diodes. This flyback diode helps protect the driver IC from voltage spikes that occur when the motor coil is turned on and off (mostly when turned off)
- The logical low in the IC is set to 1.5V. This means the pin is set high only if the voltage across the pin crosses 1.5V which makes it suitable for use in high frequency applications like switching applications (upto 5KHz)
- Lastly, this integrated circuit not only drives DC motors, but can also be used to drive relay solenoids, stepper motors etc.

There are 16 pins sticking out of this IC and functionality of each pin before implementing this in a circuit is as follows:

1. Pin 1 and Pin 9 are "Enable" pins. They should be connected to +5V for the drivers to function. If they pulled low (GND), then the outputs will be turned off regardless of the input states, stopping the motors. If you have two spare pins in your microcontroller, connect these pins to the microcontroller, or just connect them to regulated positive 5 Volts.
2. Pin 4, Pin 5, Pin 12 and Pin 13 are ground pins which should ideally be connected to microcontroller's ground.
3. Pin 2, Pin 7, Pin 10 and Pin 15 are logic input pins. These are control pins which should be connected to microcontroller pins. Pin2 and Pin7 control the first motor (left); Pin 10 and Pin 15 control the second motor (right).
4. Pin 3, Pin 6, Pin 11, and Pin 14 are output pins. Tie Pin 3 and Pin 6 to the first motor, Pin 11 and Pin 14 to second motor
5. Pin 16 powers the IC and it should be connected to regulated +5Volts.
6. Pin8 powers the two motors and should be connected to positive lead of a secondary battery. As per the datasheet, supply voltage can be as high as 36 Volts.

A quadruple half H-bridge bidirectional motor driver IC that can drive current of up to 600mA with voltage range of 4.5 to 36 volts. They have separate bridge enable option and are suitable to drive small DC-Geared motors, bipolar stepper motor etc. The specifications are as follows:

- Supply Voltage: 4.5V to 36V
- Output current capability per driver: 600mA
- Pulsed Current: 1.2A Per Driver
- Package: 16-pin PDIP

Table 2.1: L293D Pin Functions [20]

Pin No	Function	Name
1	Enable pin Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage Motors; 9-12V	Vcc $_2$
9	Enable pin Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V	Vcc

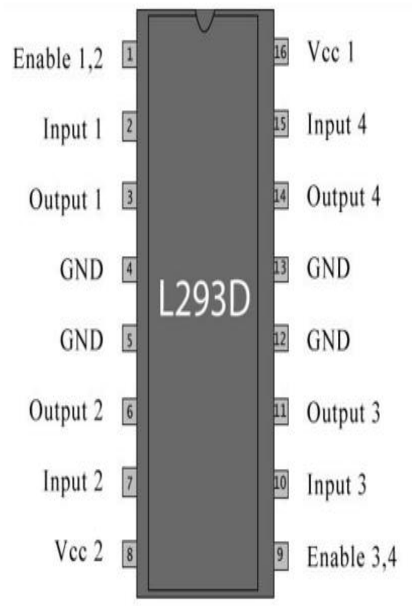


Figure 2.2: Pinout of L293D [20]

2.3.3 INFRARED (IR) SENSOR

IR sensor is a type of active proximity sensors. It emits near infrared energy and measures whether any significant amount of IR light is returned. Infrared radiation is part of the electromagnetic spectrum, which includes radiowaves, microwaves, visible light, and ultraviolet light, as well as gamma rays and X-rays. The IR range falls between the visible portion of the spectrum and the radio waves. IR wavelengths are usually expressed in microns, with IR spectrum extending from 0.7 to 1000 microns. Because every object (except black body) reflects an optimum amount of IR energy at a specific point along IR band, the reflected energy comes from an object and reaches the IR sensor through its optical system, which focuses the energy onto one or more photosensitive detectors. The detector then converts the IR energy into an electrical signal.

Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations. Hence, they are called IR LED's. Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye.

Infrared receivers are also called as infrared sensors as they detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photo diodes as they detect only infrared radiation.

Different types of IR receivers exist based on the wavelength, voltage, package, etc. When used in an infrared transmitter – receiver combination, the wavelength of the receiver should match with that of the transmitter.

Object Detection using IR light:

The basic idea is to send infra red light through IR-LEDs, which is then reflected by any object in front of the sensor.

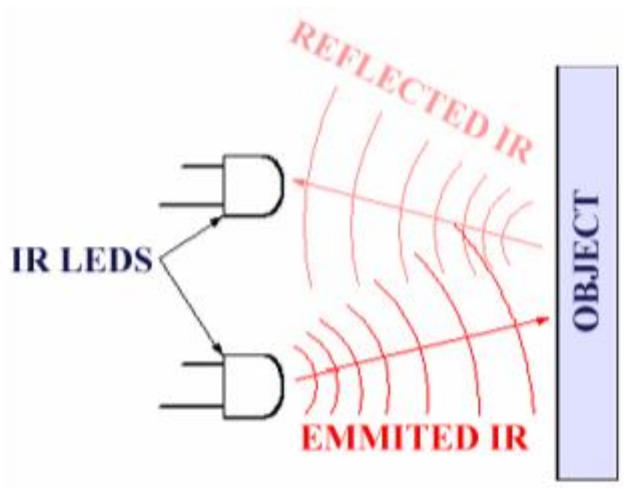


Figure 2.3: IR sensor

We use an IR emitter LED which emits infrared radiations. The radiations are reflected by any object or obstacle in its path. IR has a property that it is reflected by the white line and absorbed by the black surface. Using this principle we construct a black line follower robot. A black line is drawn on a white surface. The emitted IR is thus reflected back when sensor comes over a white surface; however no IR is reflected back in case of black surface. The reflected IR is detected by an IR receiver photodiode. This is an electrical property of receiver photodiode which is the fact that a photodiode produce a voltage difference across its leads when it is subjected to light. When the IR is reflected by white surface the voltage drop across the cathode of the receiver LED decreases.

2.3.4 LCD DISPLAY

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

8-Data pins carries 8-bit data or command from an external unit such as microcontroller. All the LCD's performs the same functions (display characters numbers special characters ASCII characters etc).

All LCDs have

- Eight(8) Data pins
- VCC (Apply 5v here)
- GND (Ground this pin)
- RS (Register select)
- RW (read - write)

- EN (Enable)
- V0 (Set LCD contrast)

V0 (Set LCD contrast)

Set LCD contrast here. Best way is to use variable resistor such as potentiometer. Output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and backward to adjust the LCD contrast.

RS(Register select)

There are two registers in every LCD

Command Register :

When we send commands to LCD these commands go to Command register and are processed there.

When RS=0   Command Register is selected.

Data Register :

When we send Data to LCD it goes to data register and is processed there.

When RS=1   Data Register is selected.

RW(Read - Write)

When RW=1, we want to read data from LCD.

When RW=0, we want to write to LCD.

EN(Enable signal) When you select the register(Command and Data) and set RW(read - write) now its time to execute the instruction. By instruction it means that 8-bit data or 8-bit command present on Data lines of LCD.

This requires an extra voltage push to execute the instruction and EN (enable) signal is used for this purpose. Usually we make it EN=0 and when we want to exectue the instruction we make it high EN=1 for some milli seconds. After this we again make it ground EN=0.

Table 2.2: LCD Functions [19]

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/writ e
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

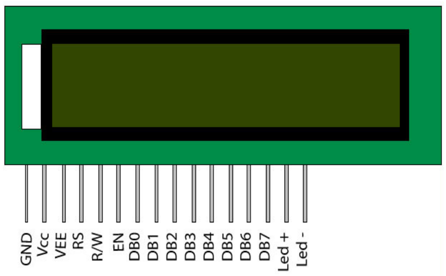


Figure 2.4: LCD Display [19]

we cannot send an integer, float, long, double type data to LCD because LCD is designed to display a character only. The 8 data pins on LCD carries only ASCII 8-bit code of the character to LCD. How ever we can convert our data in character type array and send one by one our data to LCD. Data can be sent using LCD in 8-bit 0r 4-bit mode. If 4-bit mode is used, two nibbles of data (First high four bits and then low four bits) are sent to complete a full eight-bit transfer. 8-bit mode is best used when speed is required in an application and at least ten I/O pins are available. 4-bit mode requires a minimum of seven bits. In 4-bit mode, only the top 4 data pins (4-7) are used.

2.3.5 DC GEAR MOTOR

The DC Gear motor, consisting of a DC electric motor and a gearbox, is at the heart of several electrical and electronic applications. These geared motors have reduction gear trains capable of providing high torque at relatively low shaft speed or revolutions per minute (RPM).

The two types of motors that we are likely to use in robotic adventure are DC motors and RC servo motors. The most common motor for robotics is DC gear motor, which works by gearing down a fast DC motor to make the motor turn at a slower speed and gives the motor a higher torque suitable for robot locomotion.

A dc gear motor is basically a regular dc motor with a special gear box attached to the output shaft. Your robot electrical drive circuitry can control the dc gear motor to rotate the wheels of your robot for locomotion.

You can get a dc motor without a gear head, but generally these are too fast(around 15,000 RPM).For a robot to move at reasonable rate you have to gear down a DC motor to about 30 to 80 RPM. When you gear down a DC motor ,you get a slower speed and plenty of torque.

The outer body of the gear head is made of high density plastic but it is quite easy to open as only screws are used to attach the outer and the inner structure. The major reason behind this could be to lubricate gear head from time to time. The plastic body has a threading through which nut can be easily mounted and vice versa from the gear head. The rear view of the geared motor is similar to the DC motor and it has two wires soldered to it.

These gear assemblies are highly lubricated with grease so as to avoid any sort of wear and tear due to frictional forces. The top part of the gear head is connected to rotating shaft and has one gear that allows the rotation. A strong circular imprint shows the presence of the gear that rotates the gear at the upper portion.

The gears are basically in form of a small sprocket but since they are not connected by a chain, they can be termed as duplex gears in terms of a second cog arrangement coaxially over the base. Among the three gears, two are exactly same while the third one is bigger in terms of the number of teeth at the upper layer of the duplex gear. The third gear is connected to the gear at the upper portion of the gear head.

The DC motor works over a fair range of voltage. The higher the input voltage more is the RPM (rotations per minute) of the motor. For example, if the motor works in the range of 6-12V, it will have the least RPM at 6V and maximum at 12 V.

In terms of voltage, we can put the equation as:

RPM= K1 * V, where,

K1= induced voltage constant

V=voltage applied

Working can be explained by the principle of conservation of angular momentum. The gear having smaller radius will cover more RPM than the one with larger radius. However, the larger gear will give more torque to the smaller gear than vice versa. The comparison of angular velocity between input gear (the one that transfers energy) to output gear gives the gear ratio. When multiple gears are connected together, conservation of energy is also followed. The direction in which the other gear rotates is always the opposite of the gear adjacent to it.

In any DC motor, RPM and torque are inversely proportional. Hence the gear having more torque will provide a lesser RPM and converse. In a geared DC motor, the concept of pulse width modulation is applied.

In a geared DC motor, the gear connecting the motor and the gear head is quite small, hence it transfers more speed to the larger teeth part of the gear head and makes it rotate. The larger part of the gear further turns the smaller duplex part. The small duplex part receives the torque but not the speed from its predecessor which it transfers to larger part of other gear and so on. The third gear's duplex part has more teeth than others and hence it transfers more torque to the gear that is connected to the shaft.

2.3.6 4X3 KEYPAD

A basic 12 button keypad for user input. The buttons are setup in a matrix format. This allows a microcontroller to 'scan' the 7 output pins to see which of the 12 buttons is being pressed.

A Matrix keypad is the most commonly used input device in many of the application areas like digital circuits, telephone communications, calculators, ATMs, and so on. A matrix keypad consists of a set of push button or switches which are arranged in a matrix format of rows and columns. These keypads are available in configurations like 3×4 and 4×4 based on the application it is implemented for.

Matrix keypad can be connected to the microcontroller in numerous ways or techniques, but the fundamental logic is same as making the columns as input and the rows as output. So, in order to detect the key pressed from the keypad, the row lines have to be made low one by one and to read the columns.

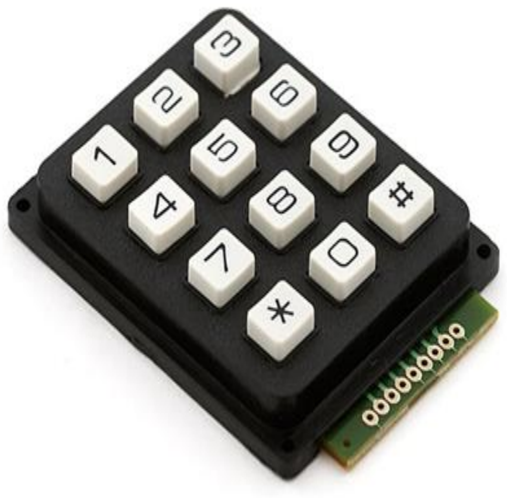


Figure 2.5: 4X3 keypad [21]

Table 2.3: Corresponding row and column for each key [21]

Data Output				Data Input			Key Pressed
R1	**R2**	**R3**	**R4**	**C1**	**C2**	**C3**	
0	1	1	1	0	1	1	1
0	1	1	1	1	0	1	2
0	1	1	1	1	1	0	3
1	0	1	1	0	1	1	4
1	0	1	1	1	0	1	5
1	0	1	1	1	1	0	6
1	1	0	1	0	1	1	7
1	1	0	1	1	0	1	8
1	1	0	1	1	1	0	9
1	1	1	0	0	1	1	*
1	1	1	0	1	0	1	0
1	1	1	0	1	1	0	#

2.3.7 AVR PROGRAMMER

AVR Programmer is used to burn code written on AVR Studio or CV AVR into the microcontroller. When we build our code in AVR Studio or CV AVR, it is converted into a hex file. This hex file is burned from computer to hardware microcontroller using AVR Programmer.

USBasp is a USB in-circuit programmer for Atmel AVR microcontrollers. It simply consists of an ATMega88 or an ATMega8 and a couple of passive components. The programmer uses a firmware-only USB driver, no special USB controller is needed.

Features of USBasp include:

- Works under multiple platforms. Linux, Mac OS X and Windows are tested.
- No special controllers or components are needed.
- Programming speed is up to 5kBytes/sec.
- SCK option to support targets with low clock speed (< 1,5MHz).
- Planned: serial interface to target (e.g. for debugging).

- 10 pin ISP interface.
- Allows you to read or write the microcontroller EEPROM, firmware, fuse bits and lock bits.

These AVR programmers are based on Thomas Fischl's USBasp design and connect to your computer's USB port.

To transfer the Hex file to controller we require a software to access USBASP. At the first time when we connect programmer to the pc our programmer will be detected as usbasp and we have to provide a proper path for drivers to be installed.

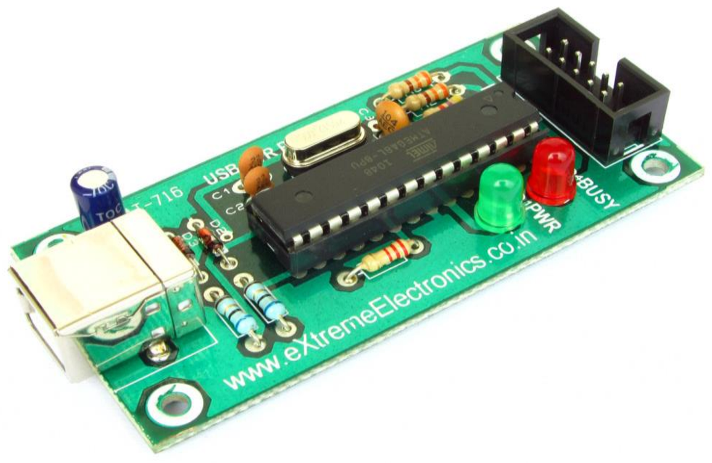


Figure 2.6: AVR Programmer

DESCRIPTION OF WORK DONE

We have divided our project into four phases i.e. conceptualization and planning phase, software implementation phase, research paper publication and hardware implementation phase. In first phase we have planned the layout of our car automated car parking area. In second phase we have implemented our whole project in software. For software implementation we are using Atmel AVR Studio (for writing code in embedded C) and ISIS professional proteus (for circuit simulation). After successful implementation of our project on software we published a research paper on our idea in international journal (IJIREEICE). And finally in fourth phase we have successful implemented our project work on hardware.

Timeline of project:

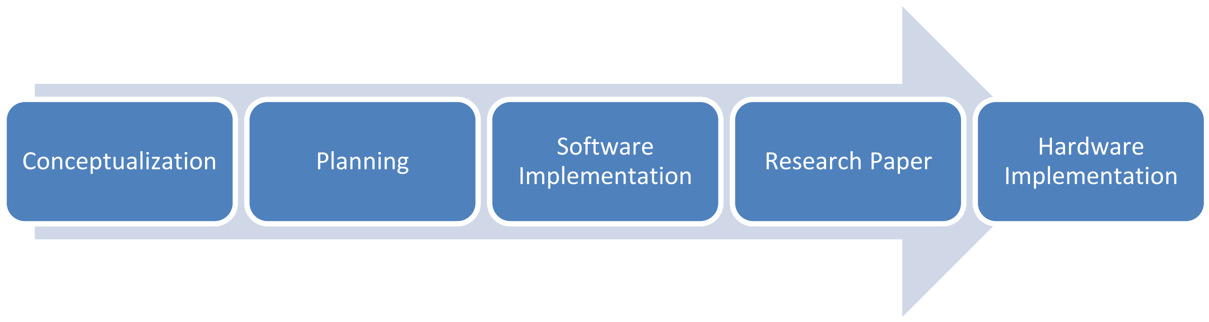


Figure 3.1: Project Timeline

Our project is divided into two major parts. First part is to design a smart parking area and second part is to design smart car. The first step towards our work was to design a parking area keeping in mind the proper movements of car like turn-around gaps and reversing space etc. Keeping all these things in mind the parking area was aesthetically designed where there is sufficient space for parking, turning, reversing and the parking entry point. Minimization of space has been done in such a way that there should be no glitches in functioning and safety of parked cars.

Next step was to select the primary key of car which we found that it had to be the car registration number through which the car will be registered in the parking. We also had to make the registration convenient so we used the LCD Display for the output and a simple 4X3 Keypad for Input. Both these components united with the microcontroller together made the Entry Gate area most simplified and convenient for this area. Management of slot numbers allocated is also done in this segment by code.

Finally carving a way out for the car to be parked at specified slot number was done. The principle used was to follow a black trajectory and indication marks which were sensed by the IR sensors for the making a way for parking of car. The marks signify indications to skip, turn and stop which is mutually coordinated with the slot number allotted for car parking. In this way we automated the car as well as parking system.

3.1 SOFTWARE IMPLEMENTATION

To implement our project on software grounds we used AVR Studio 4 and Proteus 7.8. Here we simulated our automated car parking model which gave us the desired results.

3.1.1 SMART PARKING AREA

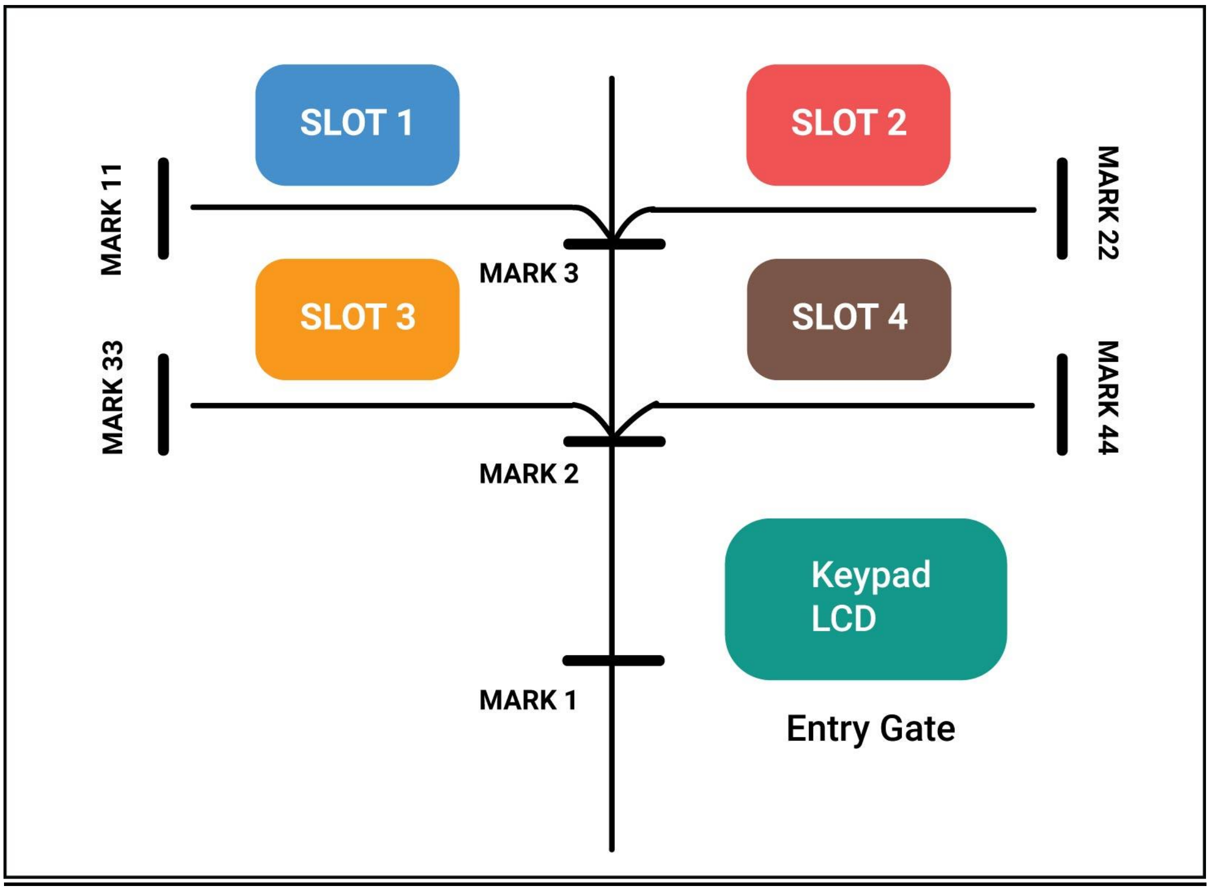


Figure 3.2: Layout of ‘smart parking area’

Description of layout of “Smart Parking Area”:

Figure 3.2 illustrates layout of our smart parking area. We have designed our parking area for four cars but we can extend our layout further for as many number of cars as required keeping the basic concept same. When the car comes to the parking area, it will stop at Entry Gate which is represented by Mark 1. Here the driver of the car will come out of the car and will enter his car registration number as the unique id of his car. As he enters his car number, an available slot will be allocated to his car. Now the driver has to just enter the allocated slot number into his car and then he can leave from the parking area. With the help of set of four sensors embedded on the car, the car will move further over the black line and will be parked onto the allocated slot. For example if the car has to be parked at slot number 1, then the car will move from Mark 1 to Mark 3, then it will rotate by 90 degree towards left and finally it will move further till Mark 11. As it reaches Mark 11, the controller will come to know that the car is successfully parked at slot 1.

ALGORITHM:

START

Step1: Car comes to Smart parking area.

Step 2: Car stops at entry gate (i.e. Mark 1).

Step 3: Car driver enters details like his phone number and car registration number on keypad at entry gate.

Step 4: Controller of parking area saves details and allocates an empty slot to the car which is displayed on LCD screen (if no slot available “parking area full!” is displayed).

Step 5: Driver enters the allocated slot on keypad embedded on smart car.

Step 6: Smart car moves inside parking area and a clock timer will start for that car.

Step 7: If allocated slot is 3, car moves forward till ‘Mark 2’, after which it will take a left turn and again move forward till ‘Mark 33’ is encountered.

Step 8: If allocated slot is 4, car moves forward till ‘Mark 2’, after which it will take a right turn and again move forward till ‘Mark 44’ is encountered.

Step 9: If allocated slot is 1, car moves forward till ‘Mark 3’, after which it will take a left turn and again move forward till ‘Mark 11’ is encountered.

Step 10: If allocated slot is 2, car moves forward till ‘Mark 3’, after which it will take a right turn and again move forward till ‘Mark 22’ is encountered.

Step 11: Once car is parked, a success message is send to the car owner/driver with an OTP.

Step 12: When car owner/driver comes back, he has to entry car registration number and valid OTP at exit gate, and his car will be automatically be de-parked from parking area to the entry gate.

Step 13: At this moment clock timer will stop and the time period will be recorded. Using this time interval, parking fee will be calculated and the car owner can take his car back after paying this fee.

STOP

FLOW CONTROL:

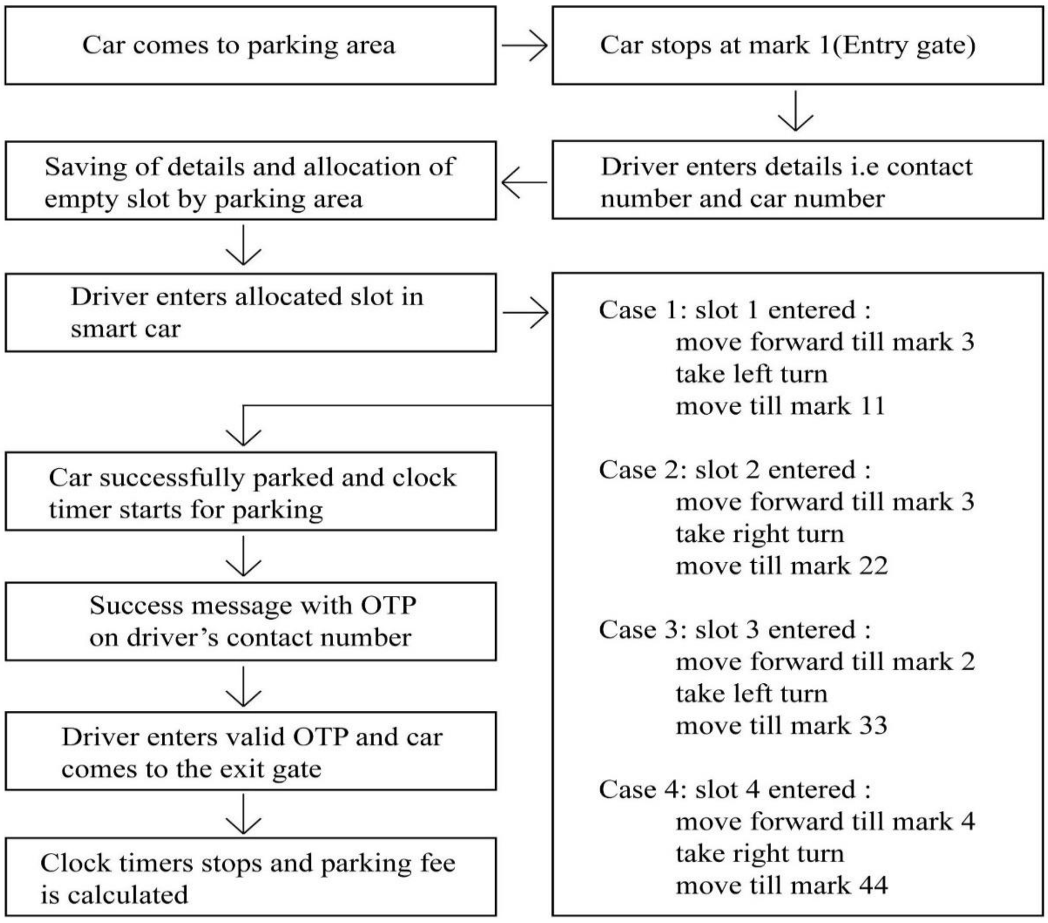

Car comes to parking area
Car stops at mark 1(Entry gate)
Driver enters details i.e contact number and car number
Saving of details and allocation of empty slot by parking area
Driver enters allocated slot in smart car
Case 1: slot 1 entered : move forward till mark 3 take left turn move till mark 11
Case 2: slot 2 entered : move forward till mark 3 take right turn move till mark 22
Case 3: slot 3 entered : move forward till mark 2 take left turn move till mark 33
Case 4: slot 4 entered : move forward till mark 4 take right turn move till mark 44
Car successfully parked and clock timer starts for parking
Success message with OTP on driver's contact number
Driver enters valid OTP and car comes to the exit gate
Clock timers stops and parking fee is calculated

Figure 3.3: Flow Control of 'smart parking area'

PROGRAM:

```
#include<avr/io.h>
#include<util/delay.h>
```

```
voidlcdcmd(unsigned char a)
{         PORTD=a;PORTC=0b00000100;
          _delay_ms(100);
          PORTC=0b00000000;
          _delay_ms(100);
}
voidlcddata(unsigned char a)
{         PORTD=a; PORTC=0b00000101;
          _delay_ms(100);
          PORTC=0b00000001;
          _delay_ms(100);
}
void refresh()
{                   lcdcmd(0x01);lcdcmd(0x80);
                    lcddata('E');lcddata('n');lcddata('t');lcddata('e');lcddata('r');lcddata(' ');
                    lcddata('c');lcddata('a');lcddata('r');lcddata(' ');
                    lcddata('n');lcddata('o');lcddata('.');lcddata(':');lcddata(' ');
}
void warning()
{                   lcdcmd(0x01);lcdcmd(0x80);
                    lcddata('I');lcddata('n');lcddata('v');lcddata('a');lcddata('l');lcddata('i');
                    lcddata('d');lcddata(' ');
                    lcddata('c');lcddata('a');lcddata('r');lcddata(' ');
                    lcddata('n');lcddata('o');lcddata('.');lcddata('.');lcddata('.');
                    _delay_ms(500);
                    refresh();
}
voidparkingfull()
{                   lcdcmd(0x01);lcdcmd(0x80);
                    lcddata('P');lcddata('a');lcddata('r');lcddata('k');lcddata('i');lcddata('n');
                    lcddata('g');lcddata(' ');
                    lcddata('i');lcddata('s');lcddata(' ');
                    lcddata('f');lcddata('u');lcddata('l');lcddata('l');lcddata('!');

}
void success(unsigned char a)
{                   lcdcmd(0x01);lcdcmd(0x80);
                    lcddata('S');lcddata('l');lcddata('o');lcddata('t');//lcddata(':');
                    lcddata(a);
                    lcddata(' ');
                    lcddata('a');lcddata('l');lcddata('l');lcddata('o');lcddata('t');lcddata('e');
                    lcddata('d');
                    _delay_ms(5000);
}
void main()
{         intcarcount=0;int a[4][4];int count=0,i=0,j=0;
DDRB = 0x00; PORTB = 0x0F; DDRA=0x0F; PORTA=0X00; DDRC=0xFF; DDRD=0xFF;
lcdcmd(0x38);             // Initialize LCD
lcdcmd(0x0E); lcdcmd(0x01);lcdcmd(0x80);
          refresh();
```

```
            for(i=0;i<4;i++){
                        for(j=0;j<4;j++){
                        a[i][j]=0;
                        }}
while (1) //loop key check forever
   {        if(carcount<4){
//first column
       PORTA = 0b00001110;
      _delay_ms(50);
       if (!(PINB & 0x01)){ lcddata('1');a[carcount][count]=1;count++;}
       if (!(PINB & 0x02)){lcddata('4');a[carcount][count]=4;count++;}
       if (!(PINB & 0x04)){lcddata('7');a[carcount][count]=7;count++;}
if (!(PINB & 0x08)) {refresh();
                        for(j=0;j<4;j++){
                        a[carcount][j]=0;}
                        count=0;}
      //second column
       PORTA =0b00001101 ;
     _delay_ms(50);
       if (!(PINB & 0x01)){lcddata('2');a[carcount][count]=2;count++;}
       if (!(PINB & 0x02)){lcddata('5');a[carcount][count]=5;count++;}
       if (!(PINB & 0x04)){lcddata('8');a[carcount][count]=8;count++;}
       if (!(PINB & 0x08)){lcddata('0');a[carcount][count]=0;count++;}
       //third column
       PORTA = 0b00001011;
       _delay_ms(50);
       if (!(PINB & 0x01)){lcddata('3');a[carcount][count]=3;count++;}
       if (!(PINB & 0x02)){lcddata('6');a[carcount][count]=6;count++;}
       if (!(PINB & 0x04)){lcddata('9');a[carcount][count]=9;count++;}
if (!(PINB & 0x08)){//lcddata('#');
                        if(count>4||count<3){
                                    warning();
                                    count=0;
                                    }
                                    else{
                                    carcount++;
                                    success(carcount+48);
                                    refresh();
                                    }
            }
}
            else{
                        parkingfull();
                        }
            }
}
```

PROTEUS SIMULATION:

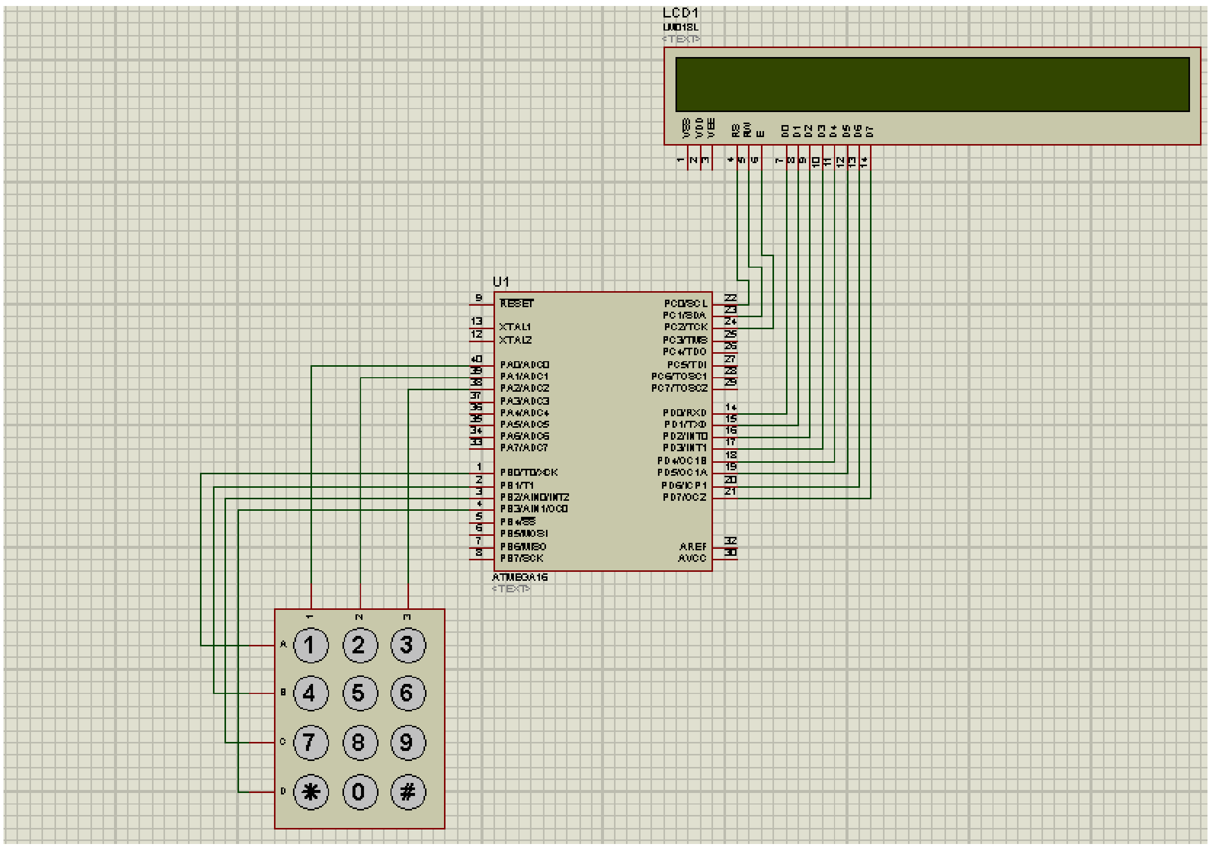


Figure 3.4: Proteus simulation of 'smart parking area'

Working of Smart Parking Area:

The figure 3.4 shows proteus simulation of smart parking area. It consists of three components i.e. AVR Microcontroller, 4X3 Keypad and LCD. When the car arrives at entry gate, the LCD initially displays "Enter Car Number". Using keypad the car owner will enter his car's registration number and an empty slot will be allocated to him which will be displayed on the LCD. If parking area is full, then a warning message will be displayed on LCD "Parking is full!" While entering car registration number if user makes any mistake then he can rest the LCD and re-enter his car registration number by pressing "*". Also once the user has entered his car number, he has to press "#" to get empty slot from the controller.

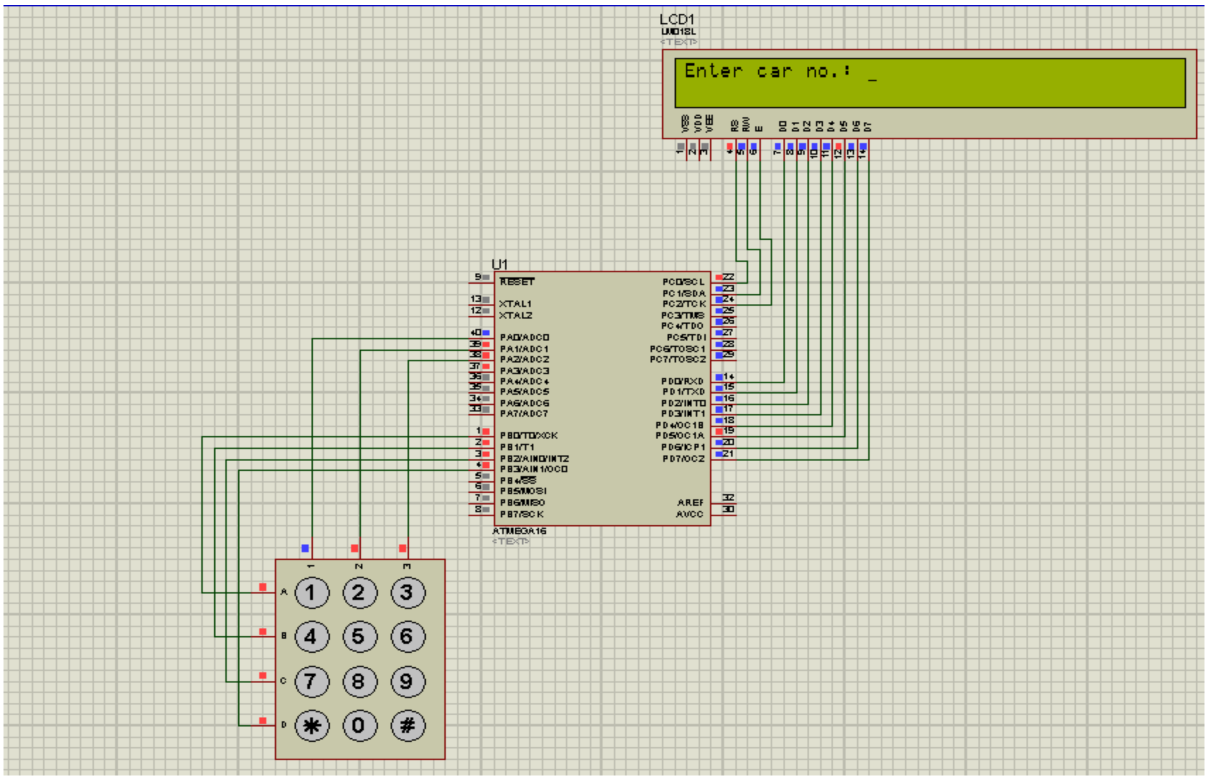


Figure 3.5: Proteus simulation of smart parking area-asking to enter car no.

Figure 3.5 shows the LCD displaying “Enter car no.:” at entry gate of parking area.

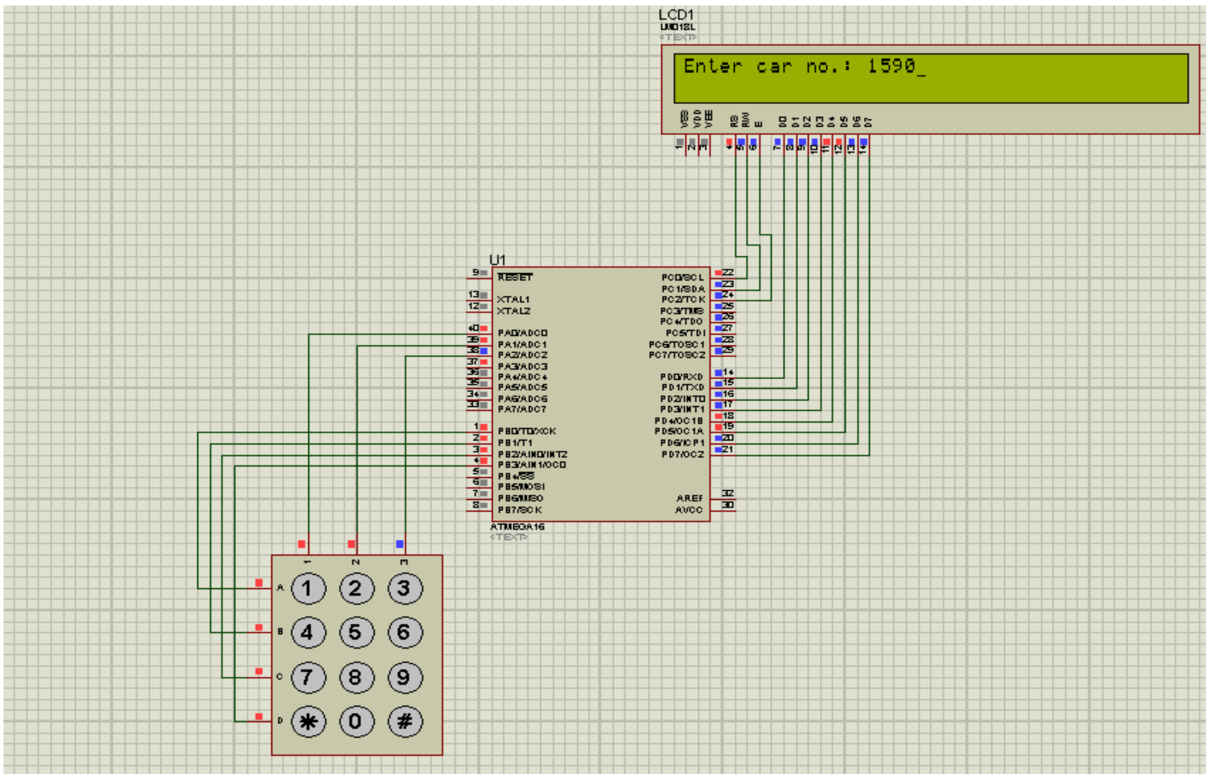


Figure 3.6: Proteus simulation of smart parking area-user entering his car no.

Figure 3.6 shows the user entering his car registration number at entry gate using keypad. In this case user has entered 1590 as his car registration number.

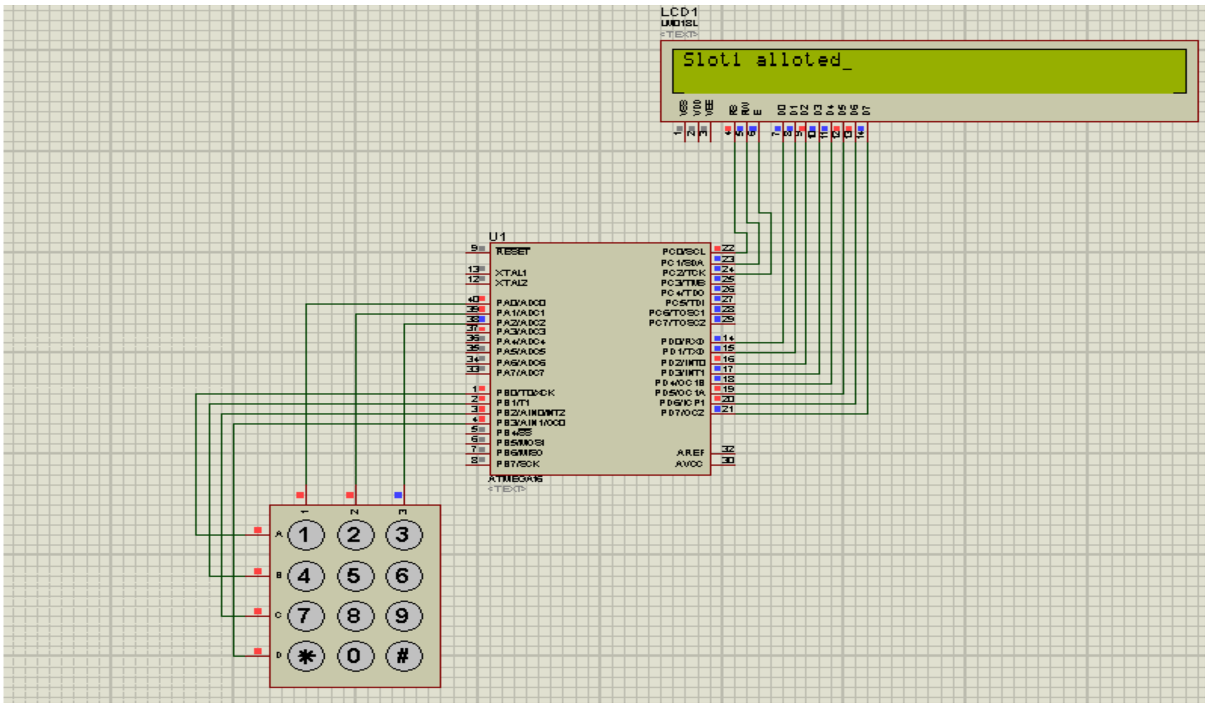


Figure 3.7: Proteus simulation of smart parking area-slot alloted

Figure 3.7 shows the allotment of an empty slot once the user press “#” from keypad. In this case slot 1 has been allocated to the user for parking of his car.

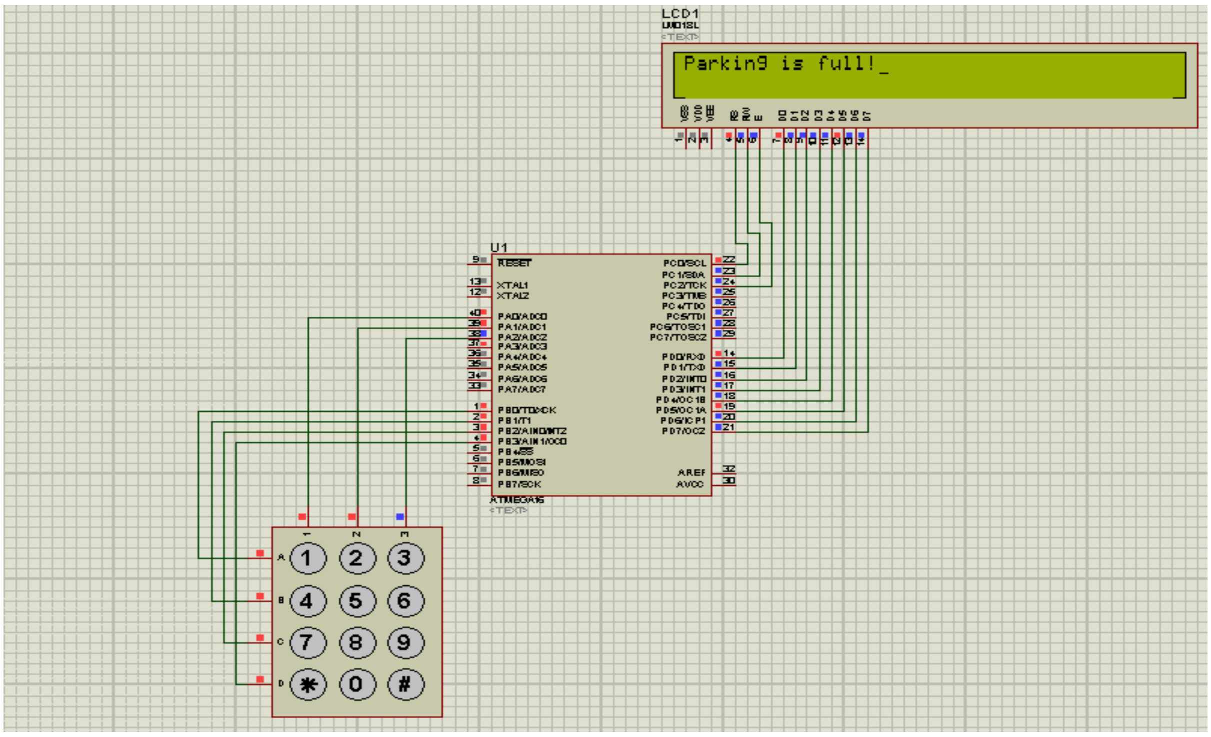


Figure 3.8: Proteus simulation of smart parking area-parking full message displayed

Figure 3.8 shows the “Parking is full!” message being displayed on LCD when all four slots have been occupied by four cars.

3.1.2 SMART CAR

Description of “Smart Car”:

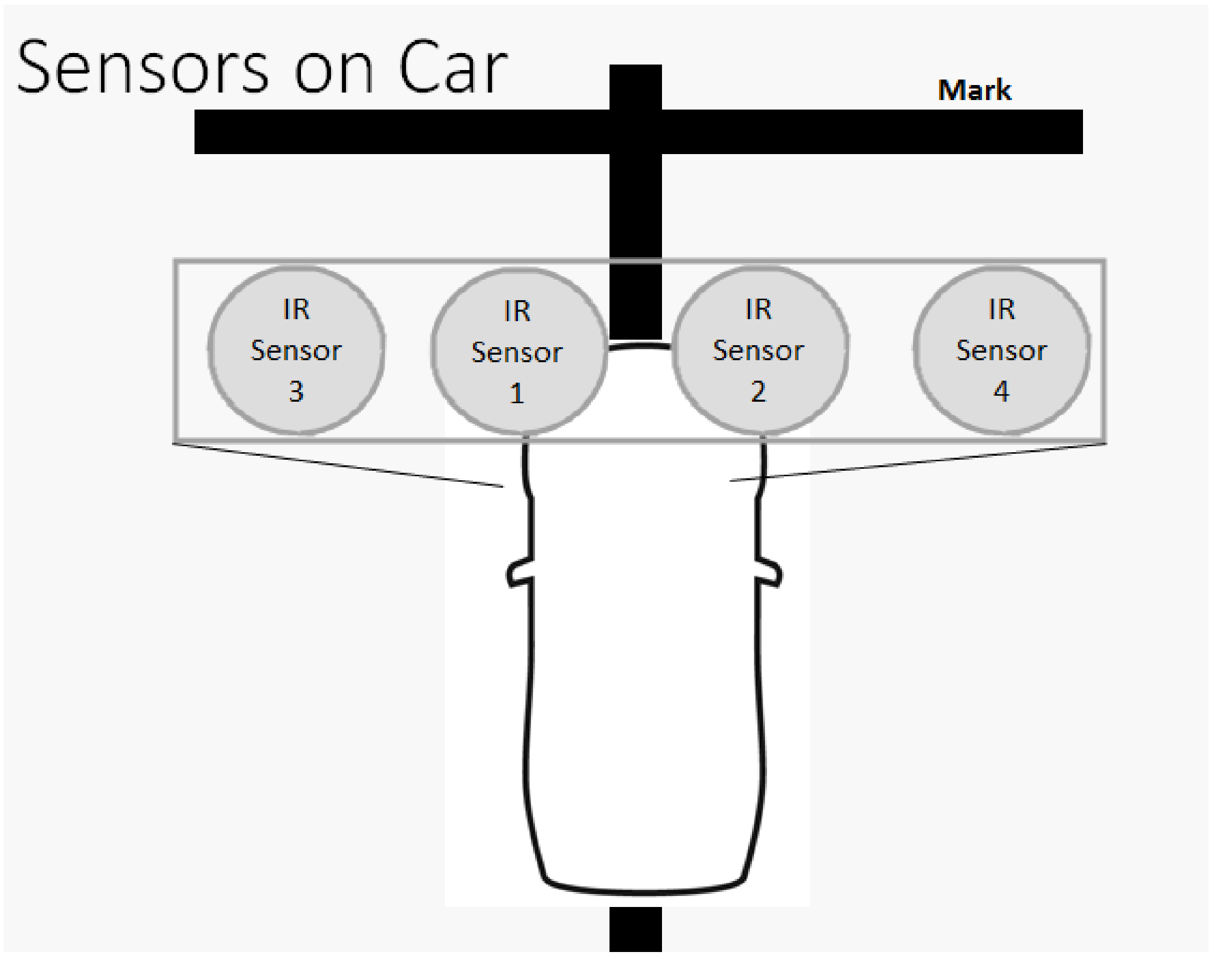


Figure 3.9: Design of “Smart Car”

Figure 3.9 illustrates the design of Smart Car. Our smart car is similar to any other car with the only difference that it has a set of four IR Sensors enbedded at its bumper. The inner two sensors are used to control the car so that the car always moves over the black path while the outer two sensors are used to detect any “Mark” being present at the parking area. Using these four sensors the car can be easily parked at its allocated slot number without the help of the driver.

ALGORITHM:

START

Step1: Car moving over black track, all four sensors are ON.

Step 2: If Sensor 1 turns OFF, that means car has lost its track and has moved towards right, so to bring it back on track switch OFF right motor till Sensor 1 turns ON.

Step 3: If Sensor 2 turns OFF, that means car has lost its track and has moved towards left, so to bring it back on track switch OFF left motor till Sensor 2 turns ON.

Step 4: If Sensor 3 and Sensor 4 both turns OFF, that means a 'Mark' is encountered. Depending upon the Mark number and the allocated slot number process the movement of car and park it accordingly.

Step 5: Once 'Mark 11/22/33/44' is encountered, it means that car is successfully parked so send a success message to car owner/driver.

STOP

FLOW CONTROL:

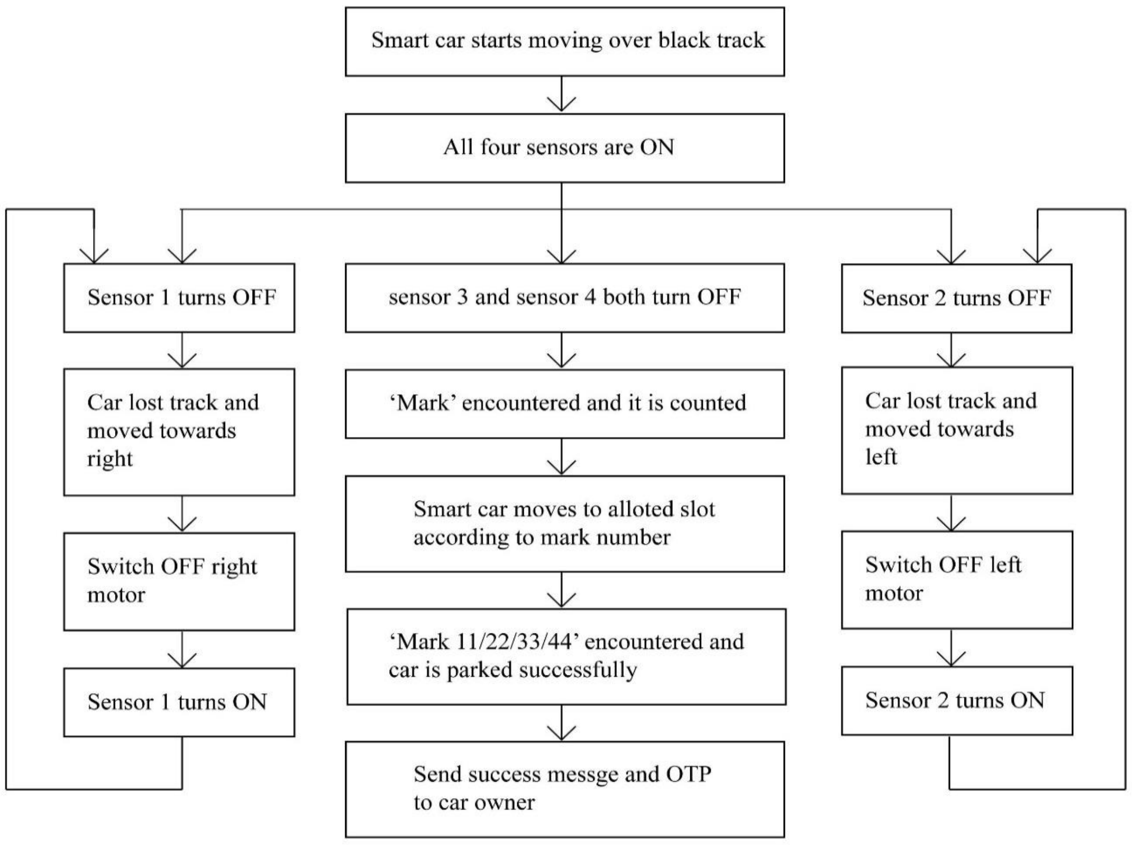

Smart car starts moving over black track
All four sensors are ON
Sensor 1 turns OFF
sensor 3 and sensor 4 both turn OFF
Sensor 2 turns OFF
Car lost track and moved towards right
'Mark' encountered and it is counted
Car lost track and moved towards left
Smart car moves to alloted slot according to mark number
Switch OFF right motor
Switch OFF left motor
'Mark 11/22/33/44' encountered and car is parked successfully
Sensor 1 turns ON
Sensor 2 turns ON
Send success messge and OTP to car owner

Figure 3.10: Flow Control of 'smart car'

PROGRAM:

```
#include<avr/io.h>               //Header file
#include<util/delay.h>            //Header file
#define sensor1 PA0               //Left Sensor
#define sensor2 PA1               //Right sensor
#define sensor3 PA2
#define sensor4 PA3
voidsw();
void park();
voidparkToSlot();
int mark=0, slotNumber=0, temp_counter=0;

voidlcd_cmd(unsigned char a)
{
        PORTD=0b11110000 & a;
        PORTD|= 0<<0 | 0<<1 | 1<<2;
        _delay_ms(50);
        PORTD&=0b11110000;
        a=a<<4;
        PORTD= 0b11110000 & a;
        PORTD|= 0<<0 | 0<<1 | 1<<2;
        _delay_ms(50);
        PORTD&=0b11110000;
}

voidlcd_data(unsigned char b)
{
        PORTD =0b11110000 & b;
        PORTD|= 1<<0 | 0<<1 | 1<<2;
        _delay_ms(50);
        PORTD&=0b11110001
        b=b<<4;
        PORTD= 0b11110000 & b;
        PORTD|= 1<<0 | 0<<1 | 1<<2;
        _delay_ms(50);
        PORTD&=0b11110001;
}

void refresh()
{
        lcd_cmd(0x01);lcd_cmd(0x80);//lcd_cmd(0x01);lcd_cmd(0x80);
        lcd_data('E');lcd_data('n');lcd_data('t');lcd_data('e');lcd_data('r');lcd_data(' ');
        lcd_data('s');lcd_data('l');lcd_data('o');lcd_data('t');lcd_data(' ');
        lcd_data('n');lcd_data('o');lcd_data('.');lcd_data(':');lcd_data(' ');
}

voiderrorMessage()
{
        lcd_cmd(0x01);lcd_cmd(0x80);
```

```
            lcd_data('I');lcd_data('n');lcd_data('v');lcd_data('a');lcd_data('l');lcd_data('i');
.           lcd_data('d');lcd_data(' ');
            lcd_data('s');lcd_data('l');lcd_data('o');lcd_data('t');lcd_data(' ');
            lcd_data('n');lcd_data('o');lcd_data('.');
            _delay_ms(2000);
}

void display()
{
lcd_cmd(0x01);lcd_cmd(0x80);
            lcd_data('E');lcd_data('n');lcd_data('t');lcd_data('e');lcd_data('r');lcd_data('e');
.           lcd_data('d');lcd_data(' ');
            lcd_data('s');lcd_data('l');lcd_data('o');lcd_data('t');lcd_data(':');lcd_data(' ');
lcd_data(slotNumber+48);
            _delay_ms(1000);
            lcd_cmd(0x01);lcd_cmd(0x80);
            lcd_data('P');lcd_data('r');lcd_data('e');lcd_data('s');lcd_data('s');lcd_data(' ');
            lcd_data('#');lcd_data(' ');
            lcd_data('t');lcd_data('o');lcd_data(' ');
            lcd_data('p');lcd_data('a');lcd_data('r');lcd_data('k');
            _delay_ms(1000);
            park();
}

voiddisplay_car_parked()
{
            lcd_cmd(0x01);lcd_cmd(0x80);
            lcd_data('C');lcd_data('a');lcd_data('r');lcd_data(' ');
            lcd_data('p');lcd_data('a');lcd_data('r');lcd_data('k');lcd_data('e');lcd_data('d');
            lcd_data('!');
            _delay_ms(1000);
}

int main(void)          //main function
{
DDRB=0xFF;              //output pin high
DDRA=0xF0;              //input pin low
DDRC=0b00000111;
PORTC=0b11110000;
DDRD =0b11111111;
lcd_cmd(0x02);  //IMPORTANT
lcd_cmd(0x28); lcd_cmd(0x0E); lcd_cmd(0x01); lcd_cmd(0x80);
refresh();

while(1)
{           PINA=0x0F;
}
}
void sw1()
{
        if(bit_is_clear(PINA,sensor2))       //checks whether right sensor is low
        PORTB=0b00000000;              //both left and right side motor low if true
        else
        PORTB=0b00000100;             //right side motor turns on and left motor off if false
}
void sw2()
{
        if(bit_is_clear(PINA,sensor2))       //checks whether right sensor is low
        PORTB=0b00000001;                 //right side motor turns off and left motor on if true
        else
        PORTB=0b00000101;                 //both left and right side motor high if false
}
voidsw()
{
        if(bit_is_clear(PINA,sensor1))    //Checks whether left sensor is low
        sw1();          //terminate to this function if true
        else
        sw2();          //terminate to this function if false
        if(bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4))
        {       if(slotNumber==0)
                {       park();
                }
                if(slotNumber!=0)
                {       parkToSlot();
                }
        } // if(bit_is_clear(PINA,sensor3)&&bit_is_clear(PINA,sensor4))
}

void park() // to stop the car, take slot no. & enter it into car keypad
{
        if(mark==0 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4)
        &&slotNumber==0) //Then stop the car for allocation of slot no. &entrying slot no. in
        car
        {       PORTB=0b00000000; // stop the car
        }
        PORTC=0b00000110;
        _delay_ms(50);
        if(!(PINC & 0x10)){lcd_data('1');slotNumber=1;}
        if(!(PINC & 0x20)){lcd_data('4');slotNumber=4;}
        if(!(PINC & 0x40)){lcd_data('7');slotNumber=-1;} // We are making project for 4
        slots so if slot no. entered is greater than 4, make it -1 & display error afterwards.
        if(!(PINC & 0x80)){refresh();slotNumber=0;} // If user enter *, then clear or refresh
        the screen.

        PORTC=0b00000101;
        _delay_ms(50);
        if(!(PINC & 0x10)){lcd_data('2');slotNumber=2;}
        if(!(PINC & 0x20)){lcd_data('5');slotNumber=-1;}
        if(!(PINC & 0x40)){lcd_data('8');slotNumber=-1;}
        if(!(PINC & 0x80)){lcd_data('0');slotNumber=-1;}
        PORTC=0b00000011;
        _delay_ms(50);
        if(!(PINC & 0x10)){lcd_data('3');slotNumber=3;}
        if(!(PINC & 0x20)){lcd_data('6');slotNumber=-1;}
        if(!(PINC & 0x40)){lcd_data('9');slotNumber==-1;}
        if(!(PINC & 0x80)){parkToSlot();} // If user enter #, then call the function to move
        the CAR to the slot

        if(slotNumber==-1)
        {
        slotNumber=0;
        errorMessage();
        refresh();
        }
        if(slotNumber>=1 &&slotNumber<=4)
        display();
}// park()
voidparkToSlot() // to park car to the allocated slot
{
        if(mark==0 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4)
        &&slotNumber!=0) // i.e. driver has entered the slot
        {       mark=1;
                PINA=0x0F;
                sw();//move car forward
        }
        if(mark!=0 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4)
        &&slotNumber!=0) // i.e. car has reached second mark
        {
                if(slotNumber==3) // move left i.e. to slot 3
                {       mark=2;
                        // stop the car & rotate the car by 90 degree anti-clockwise(i.e. towards
                        left) and then move forward
                        PORTB=0b00000000;
                        for(int i=1;i<=100000;i++) { // for left side rotatin stop left motor &
                        start right motor
                        PORTB=0b00000101;
                        }
                        PORTB=0b00000000;
                        PINA=0x0F;
                        mark=33;
                        sw();

                }//if(slotNumber==3)

                if(mark==33 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4))
                {       display_car_parked();
                }

                if(slotNumber==4) // move right i.e. to slot 4
```

```
{         mark=2;
          // stop the car & rotate the car by 90 degree clockwise(i.e. towards
          right) and then move forward
          PORTB=0b00000000;
          for(int i=1;i<=100000;i++){// for right side rotatin stop right motor &
          start left motor
          PORTB=0b00000001;
          }
          PORTB=0b00000000;
          PINA=0x0F;
          mark=44;
          sw();

}//if(slotNumber==4)

if(mark==44 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4))
{         display_car_parked();
}

if(slotNumber==1) // move forward to mark 3 and then to left i.e. to slot 1
{         mark=2;
          if(temp_counter==0) // then move forward as it is mark 2
          {         temp_counter=1;
                    PINA=0x0F;
                    sw();
          } //if(temp_counter==0)
          if(temp_counter==1) // then stop as it is mark 3
          {         mark=3; // stop the car & rotate the car by 90 degree anti-
                    //clockwise(i.e. towards left) and then move forward
                    PORTB=0b00000000;
                    for(int i=1;i<=100000;i++){// for left side rotatin stop left
                    motor & start right motor
                    PORTB=0b00000101;
                    }
                    PORTB=0b00000000;
                    PINA=0x0F;
                    mark=11;
                    sw();
          }//if(temp_counter==1)
}//if(slotNumber==1)

if(mark==11 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4))
{         display_car_parked();
}

if(slotNumber==2) // move forward to mark 3 and then to right i.e. to slot 2
{
          mark=2;
          if(temp_counter==0) // then move forward as it is mark 2
          {
```

```
                    temp_counter=1;
                    PINA=0x0F;
                    sw();//move car forward (do not stop, rather move forward & then stop
                    on mark 3)
                    } //if(temp_counter==0)

                    if(temp_counter==1) // then stop as it is mark 3
                    {
                    mark=3;
                    // stop the car & rotate the car by 90 degree clockwise(i.e. towards
                    right) and then move forward
                    PORTB=0b00000000;
                    for(int i=1;i<=100000;i++){// for right side rotatin stop right motor &
                    start left motor
                    PORTB=0b00000101;
                    }
                    PORTB=0b00000000;
                    PINA=0x0F;
                    mark=22;
                    sw();
                    }//if(temp_counter==1)
            }//if(slotNumber==2)

            if(mark==22 &&bit_is_clear(PINA,sensor3) &&bit_is_clear(PINA,sensor4))
            {       display_car_parked();
            }
    }
}// parkToSlot()
```

PROTEUS SIMULATION:

Figure 3.11 shows the proteus simulation of smart car. It consists of AVR Microcontroller, Keypad, LCD display, L293D Motor Driver, 2 Motors and four switches. These four switches represent four sensors which are embedded on front bumper of car. All these components combines together to form a nonholonomic car which corresponds to an original car.

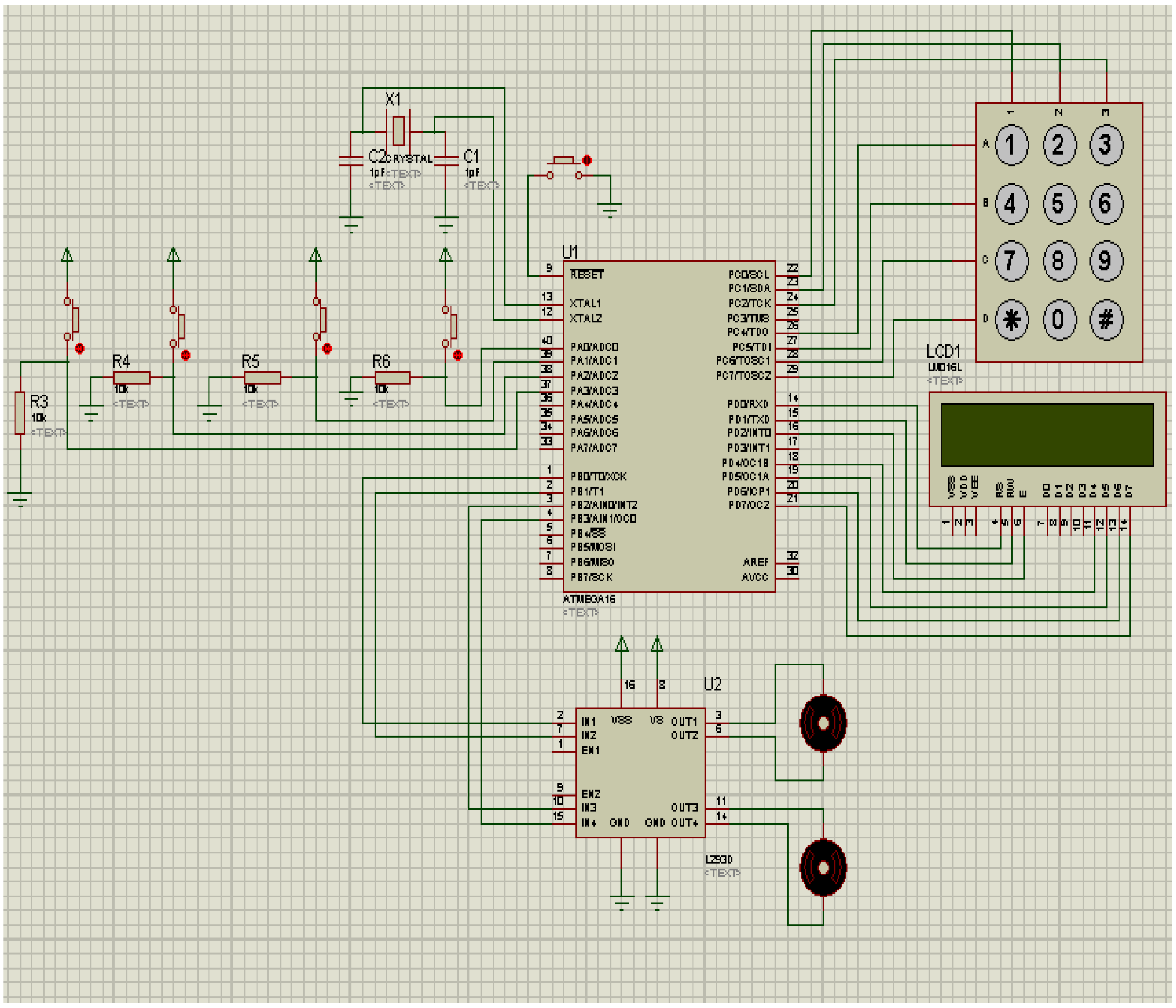


Figure 3.11: Proteus simulation of 'smart car'

Working of "Smart Car":

Since the car will be moving over a Black Path, and the two sensors (sensor 1 and 2) will be projected just outside the path where the surface of parking area is white (light color), so when the car will be moving at that time all four sensors will be ON. If by mistake car moves towards left, then sensor 2 will automatically come over black path, and as black color will absord all infrared radiations of nothing will be sensed by receiver and sensor 2 will turn OFF. As sensor 2 gets turned OFF, code burned in microcontroller will guide the car to come back to its path i.e. to move slightly towards right and for this controller will switch off right wheel for some time and only left wheel will rotate, which will move the car towards right. When the car will come back over the path, sensor 2 will again turn ON and both wheels will start rotating again. Exactly opposite will happen if the car by mistake moves towards right.

Table 3.1: Working of Sensor 1 and Sensor 2 of car

Sensor 1	**Sensor 2**	**Movement of car**
OFF	OFF	Not Possible
OFF	ON	Car has lost its track and has moved towards right, to bring it back on track switch off left motor & switch on right motor
ON	OFF	Car has lost its track and has moved towards left, to bring it back on track switch off right motor & switch on left motor
ON	ON	Car is moving over black track

Sensor 3 and 4 are are to detect Marks present on parking area. When car will be moving both these sensors will be ON because they will be over white surface. When ever any Mark is encounterd, since marks are of black color, they will turn off both sensor 3 and 4. So whenever both these sensors get OFF we can notify that a Mark has been encountered.

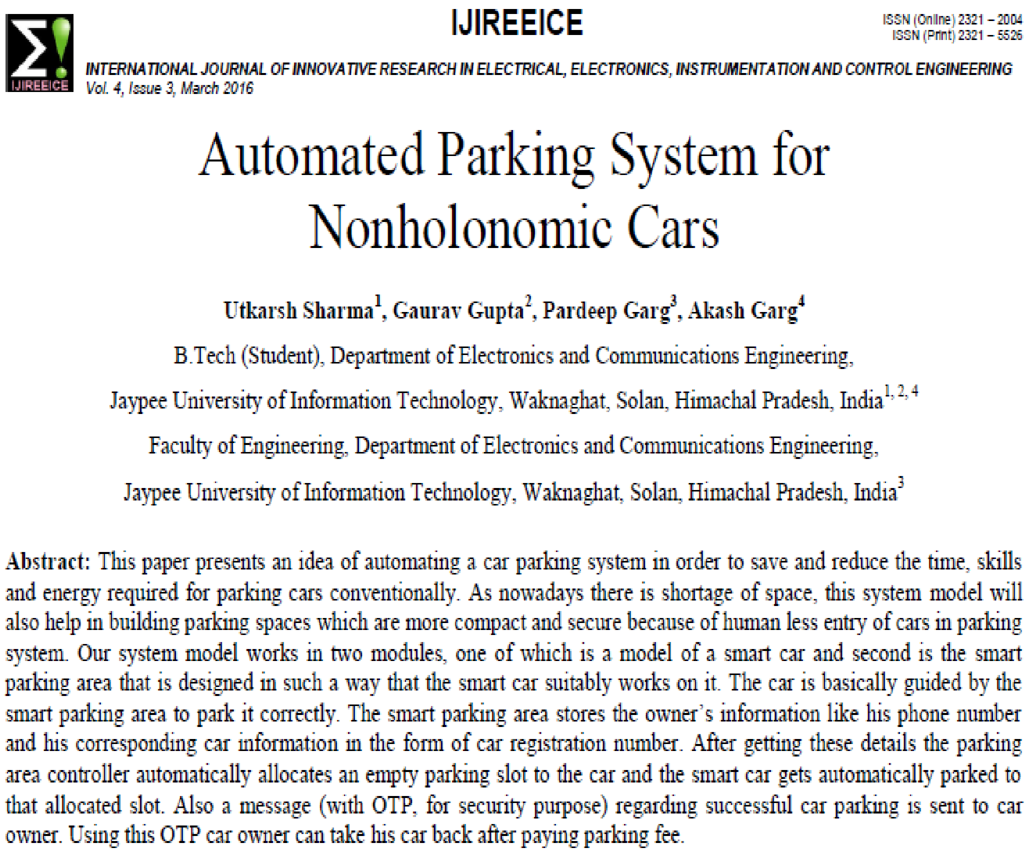
Table 3.2: Working of Sensor 3 and Sensor 4 of car

Sensor 3	**Sensor 4**	**Movement of car**
OFF	OFF	Car has reached to a “Horizontal Mark” (stop at the mark till the next command to move is given)
OFF	ON	Not Possible
ON	OFF	Not Possible
ON	ON	Car is moving over black track

3.2 PUBLICATION

Utkarsh Sharma, Gaurav Gupta, Pardeep Garg, Akash Garg, “Automated Parking System for Nonholonomic Cars”, International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, Vol. 4, Issue 3, March 2016, pp. 196-200 (DOI: 10.17148/IJIREEICE.2016.4352) [Google Citation]

http://www.ijireeice.com/upload/2016/march-16/IJIREEICE%2052.pdf


IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 3, March 2016

Automated Parking System for Nonholonomic Cars

Utkarsh Sharma[1], Gaurav Gupta[2], Pardeep Garg[3], Akash Garg[4]

B.Tech (Student), Department of Electronics and Communications Engineering,

Jaypee University of Information Technology, Waknaghat, Solan, Himachal Pradesh, India[1, 2, 4]

Faculty of Engineering, Department of Electronics and Communications Engineering,

Jaypee University of Information Technology, Waknaghat, Solan, Himachal Pradesh, India[3]

Abstract: This paper presents an idea of automating a car parking system in order to save and reduce the time, skills and energy required for parking cars conventionally. As nowadays there is shortage of space, this system model will also help in building parking spaces which are more compact and secure because of human less entry of cars in parking system. Our system model works in two modules, one of which is a model of a smart car and second is the smart parking area that is designed in such a way that the smart car suitably works on it. The car is basically guided by the smart parking area to park it correctly. The smart parking area stores the owner’s information like his phone number and his corresponding car information in the form of car registration number. After getting these details the parking area controller automatically allocates an empty parking slot to the car and the smart car gets automatically parked to that allocated slot. Also a message (with OTP, for security purpose) regarding successful car parking is sent to car owner. Using this OTP car owner can take his car back after paying parking fee.

Figure 3.12: Snapshot of first page of our research paper

3.3 HARDWARE IMPLEMENATATION

Our next and final step was to implement our idea on hardware so that this automated parking system can be physically realised.

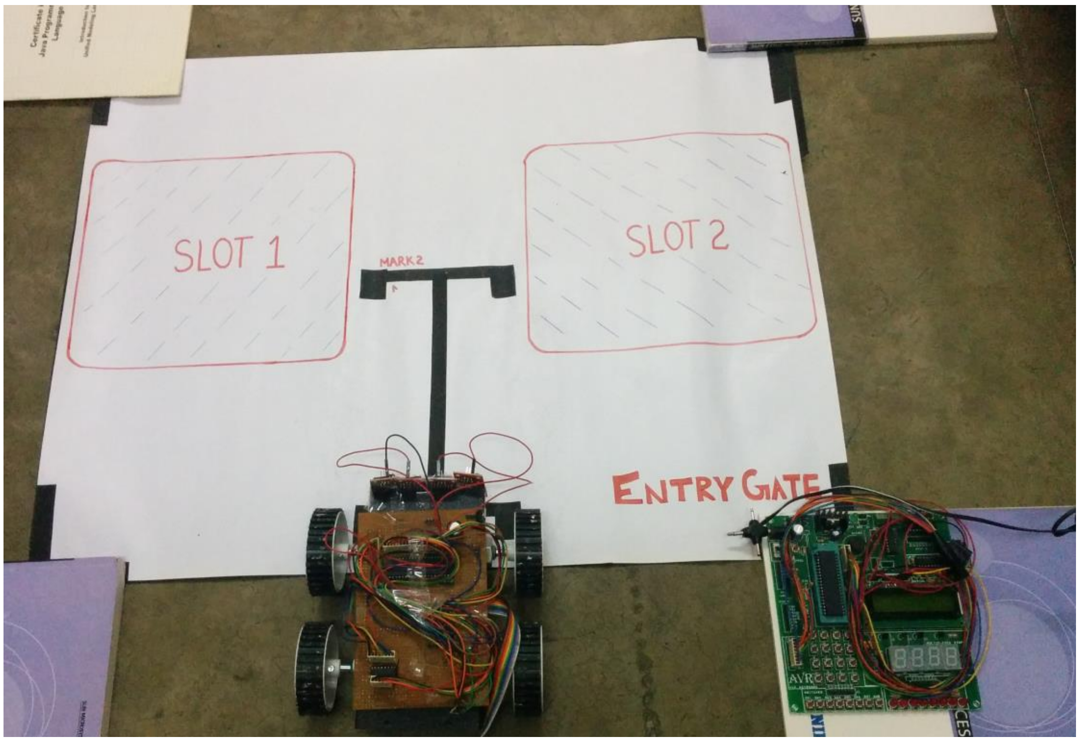


Figure 3.13: Photo of smart parking area with smart car

Figure 3.13 represents layout of our whole project i.e. smart parking area with smart car. For keeping our project simple we have used only two slots in our hardware implementation instead of using four slots as we did for software implementation.

3.3.1 SMART PARKING AREA

The main part of Smart Parking Area is its control board which is present at entry gate of parking system. This control board consists of LCD display, AVR Microcontroller and Keypad interface as shown in figure 3.14.

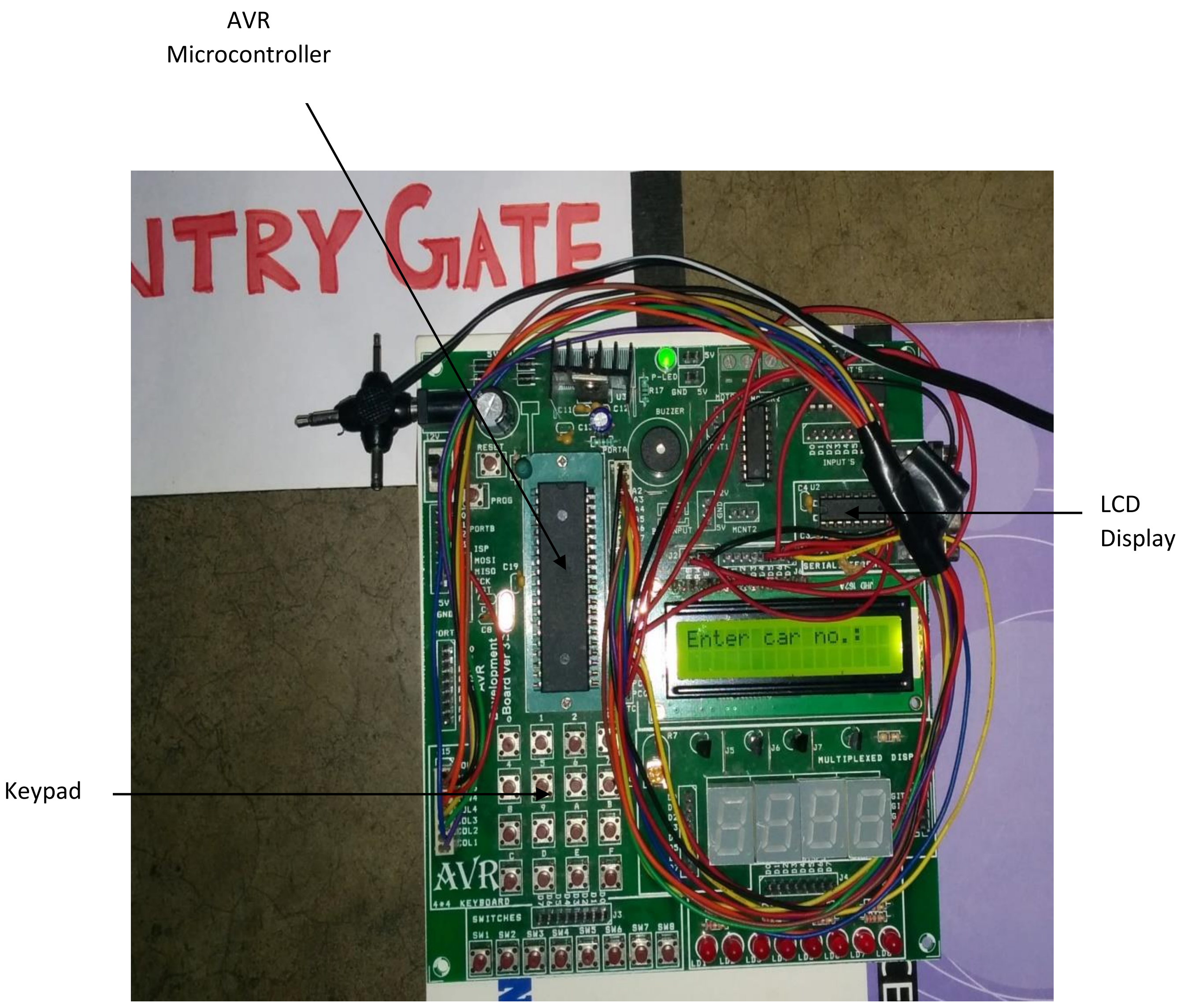


Figure 3.14: Photo of smart parking area control board (at Entry Gate)

3.3.2 SMART CAR

Figure 3.15 shows the Smart Car which consists of car chassis, IR sensors, car wheels and Printed Circuit Board(PCB). PCB consists of AVR Microcontroller, L293D motor driver and L7805 voltage regulator.

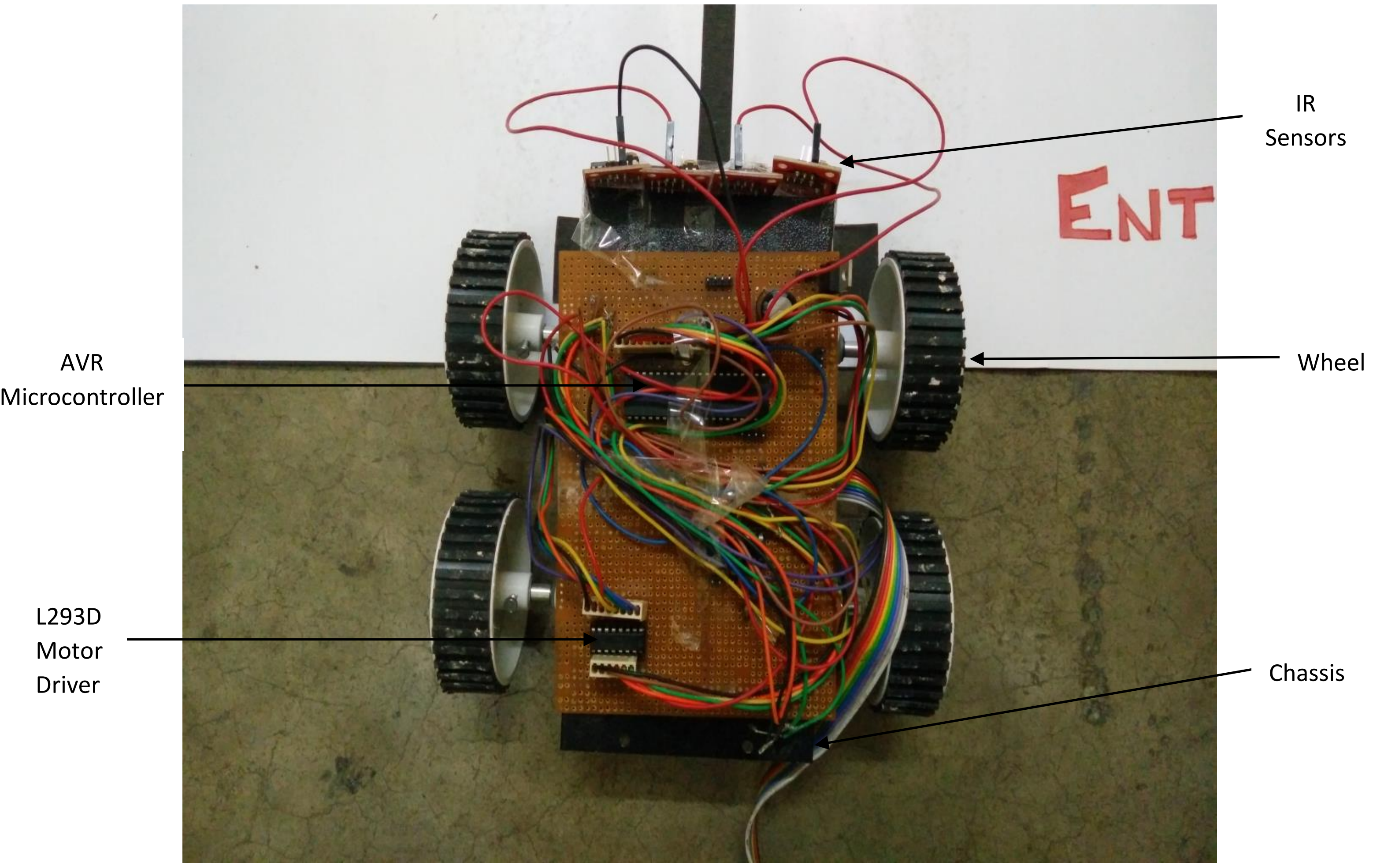


Figure 3.15: Photo of smart car (nonholonomic car)

3.3.3 WORKING OF AUTOMATED CAR PARKING SYSTEM

Initially as we can see in figure 3.14, the LCD at control board present at entry gate of smart parking area displays "Enter Car Number". Now the user will enter his car registration number on the control board using the keypad as shown in figure 3.16. Once the user has entered his car registration number, an empty slot will be allocated to his car. In figure 3.17 slot 1 has been allocated to him.

In case no slot is empty, an warning message "Parking is full!" will be displayed on the LCD screen of control board. This case is showwn in figure 3.18.

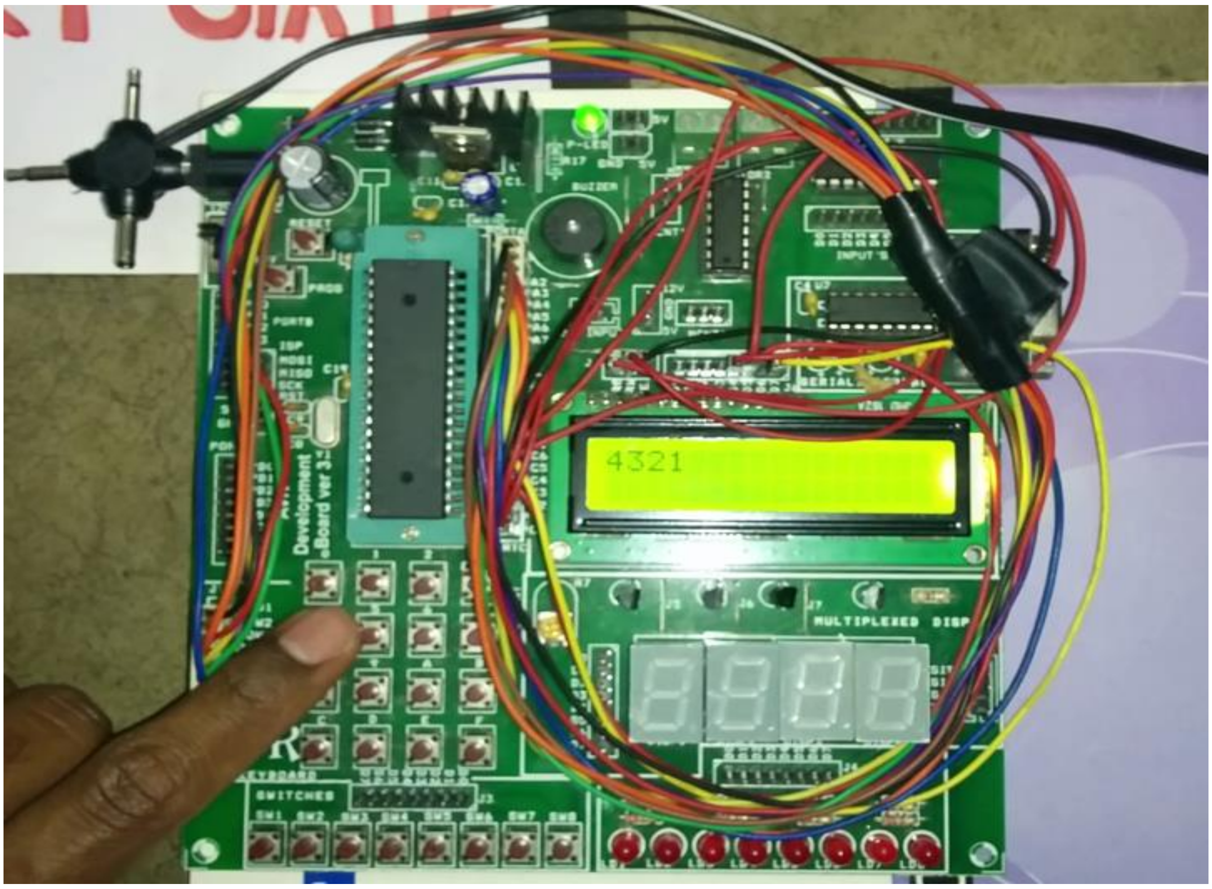


Figure 3.16: User entering his car registration number

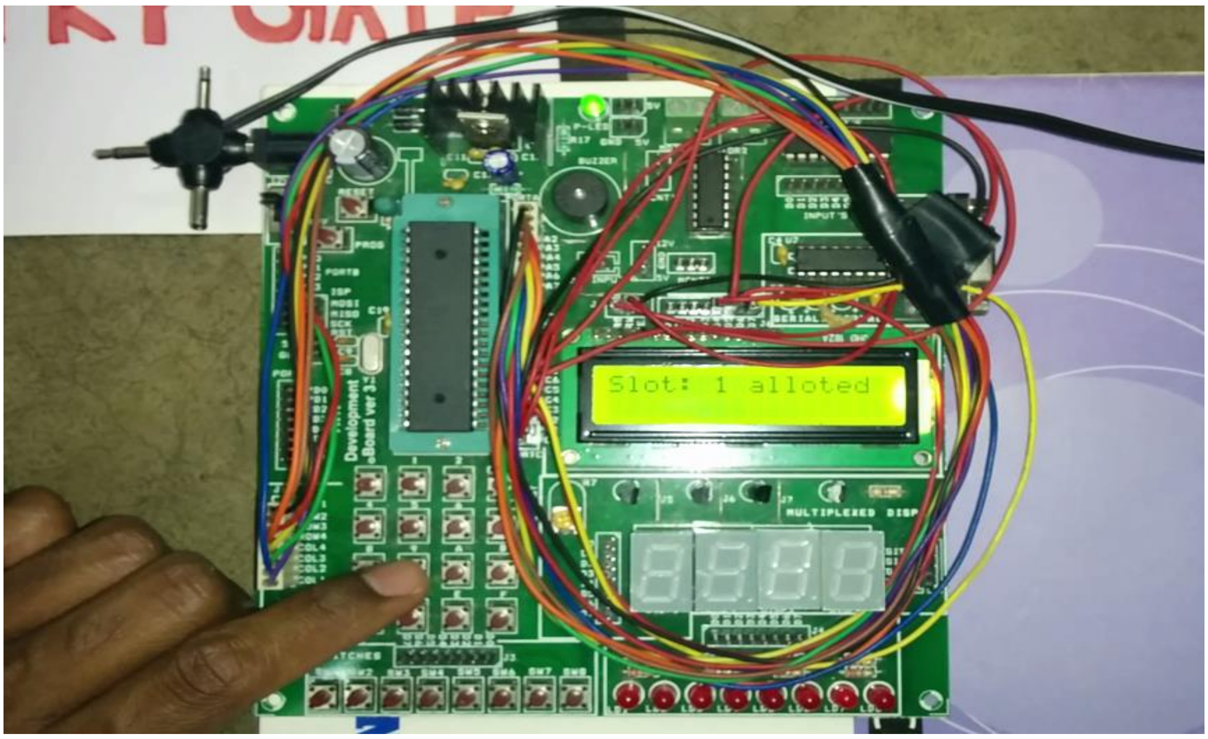


Figure 3.17: An empty slot is allocated to the user (here slot 1 is allocated)

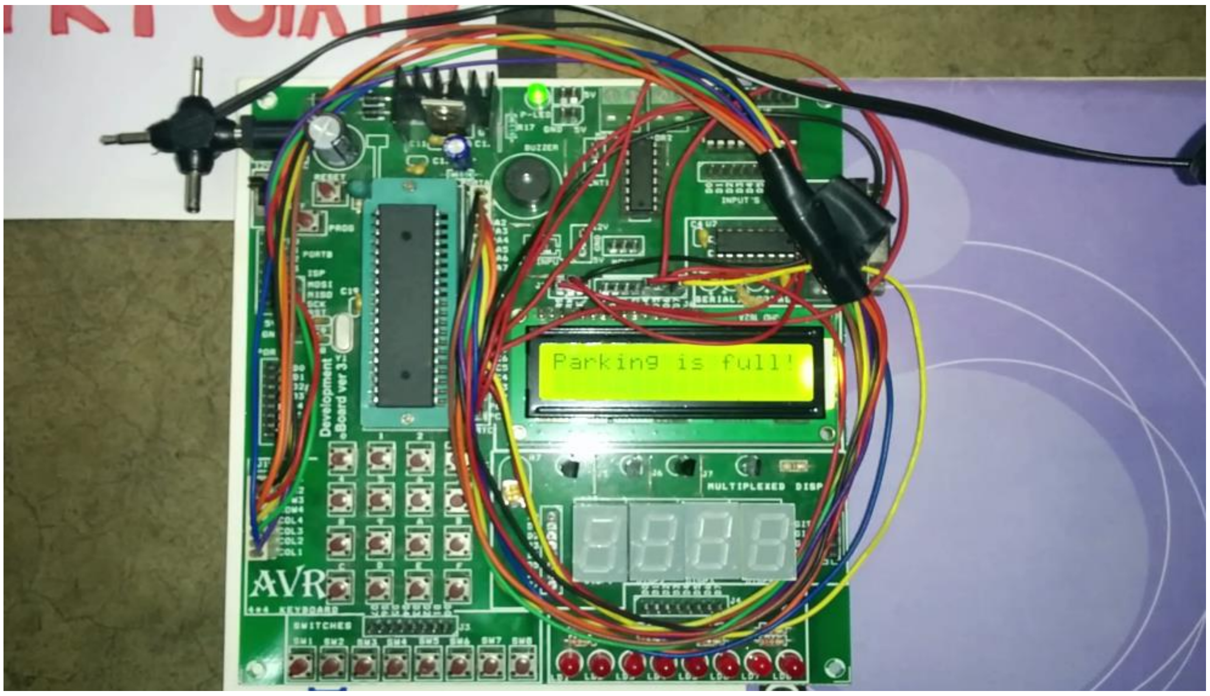


Figure 3.18: When parking area is full, “Parking is full!” is displayed

Once allotment of slot is done, the car will automatically move to the allocated slot with the help of set of four sensors embedded at front of the chassis. Figure 3.19 illustrates the movement of the robot car over the black trajectory while figure 3.20 shows the car being parked at the alloted slot i.e. slot 1.

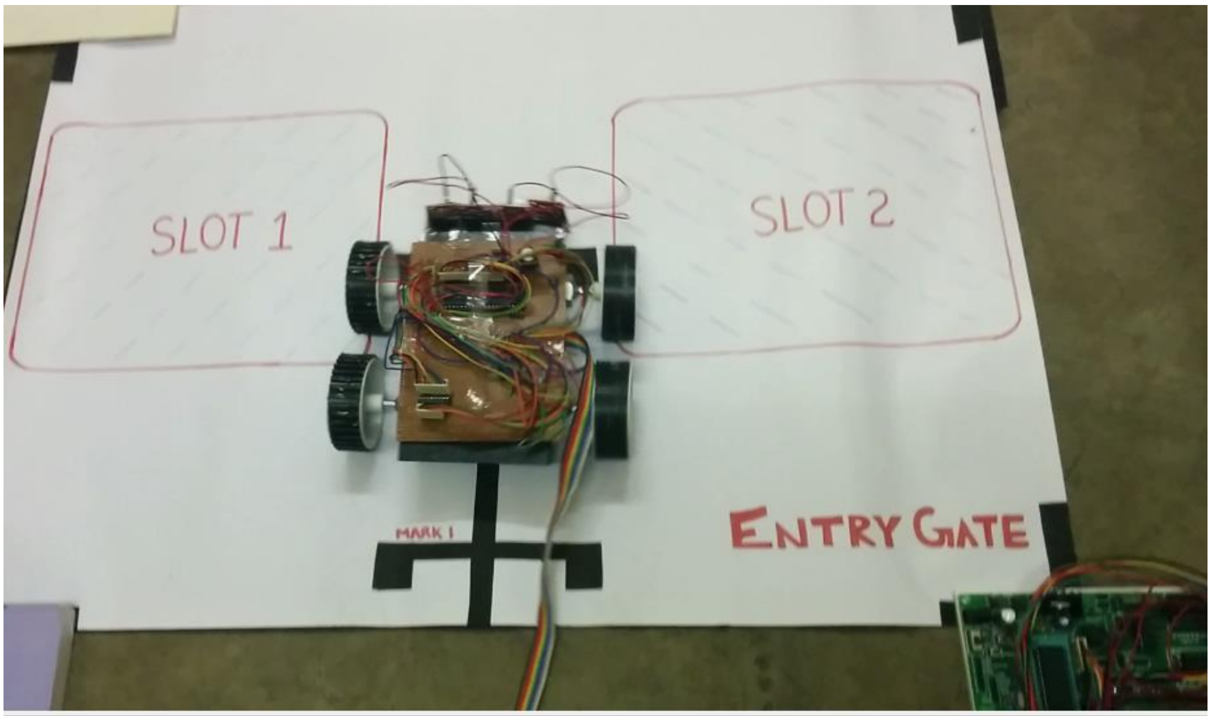


Figure 3.19: Car getting parked at parking area

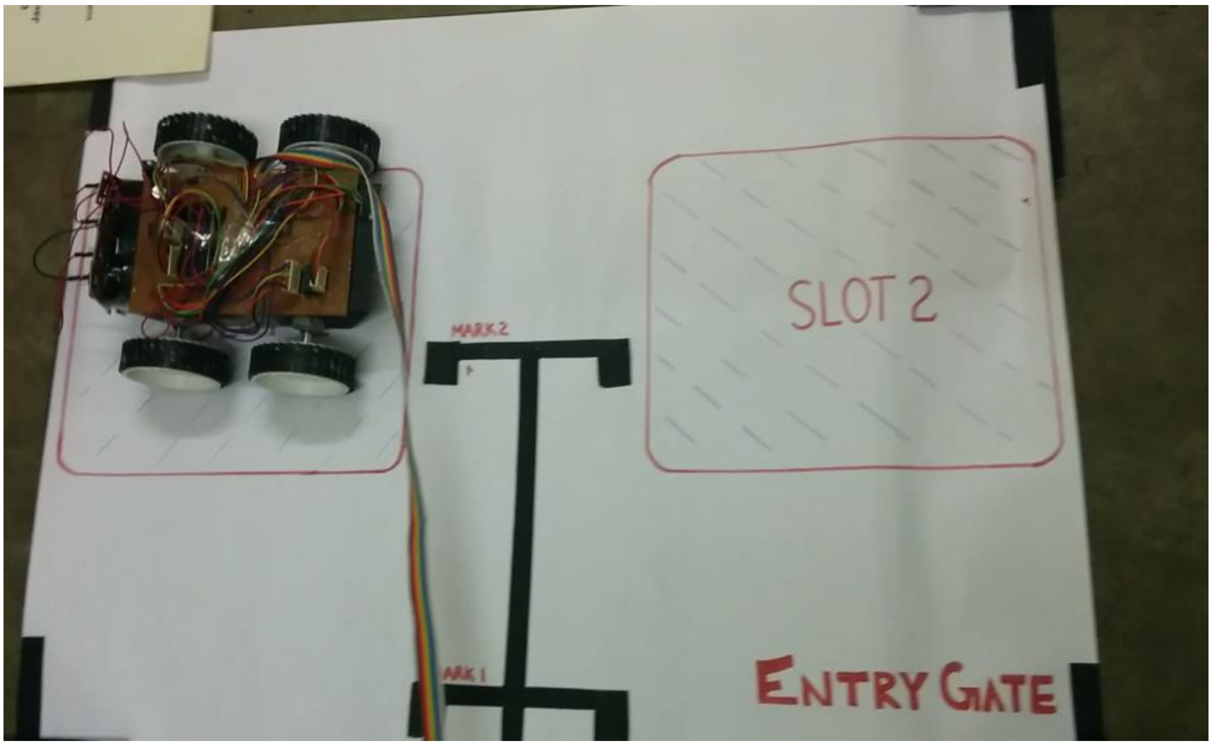


Figure 3.20: Car being parked at slot 1

CONCLUSION

This report proposes an idea of automated car parking system in contrast to manual parking which was dependent on car driver. This idea is implemented using sensors and microcontrollers. This type of automated parking model not only saves time, space and energy but also provides an intelligent way of parking which reduces problems like car collisions and accidents which may arise due to human error and negligence. This idea is divided into two modules, first module is designing of smart parking area and second module is designing of smart car which can move within that smart parking area.

Further the design and convention used by us can be made "Standard" so that each and every parking area and smart car using same standards. Almost all big companies like Ford, Hyundai, Volvo, BMW and Google are working on making driverless car. In future cars will be moving on road without any driver driving it. So our idea of providing an automated car parking area for these driverless cars will be an add-on to these new upcoming technologies because it will provide better parking facility to those driverless cars.

This report is divided into three chapters. First chapter gives introduction of our project with its benefits and disadvantages. Second chapter gives review of our project with brief introduction of the software and the hardware components being used in our project. Third chapter explains in details the work done by us i.e. the software implementation, research paper publication and the hardware implementation of our project.

"Our Software and hardware implementation are both successful and through this we can say that this system is purely a realisable concept and the future will be surely headed towards it."

FUTURE WORK

We can add more features to make our project more user friendly and secure. First of all we will can add a GSM module to our smart parking area. When the car owner reaches the entry gate, along with car registration number he has to also enter his mobile phone number. When the car gets parked to the allotted slot, a SMS will be send to the car owner in which along with allotted slot number, entry time and a random number will also be send. This random number will act like OTP (one time password) which the car owner has to enter when he comes back to take back his car at exit gate. The car will be given to the person only if he enters correct OTP corresponding to his car registration number. This will increase security to our system. Also a timer will start at the time when the car will enter the parking area and that timer will stop when the car will come out of the parking area. The time between the entry and exit of the car can be calculated, which will help in further calculation of parking fee.

REFERENCES

[1] M. M. Rashid, A. Musa, M. Ataur Rahman, and N. Farahana, A. Farhana, “Automatic Parking Management System and Parking Fee Collection Based on Number Plate Recognition,” *International Journal of Machine Learning and Computing, Vol. 2, No. 2, April 2012.*

[2] K. Jiang School of Manufacturing & Mechanical Engineering, The University of Birmingham B 15 2TT UK
and L. D. Seneviratne Department of Mechanical Engineering, King’s College London WC2R 2LS UK, “A sensor guided autonomous parking system for nonholonomic mobile robots,” *Proceedings of the 1999 IEEE*
International Conference on Robotics & Automatlon Detroit, Michigan May 199.

[3] Zhi-Long Wang, Chih-Hsiung Yang, Tong-Yi Guo, “The Design of An Autonomous Parallel Parking Neuro-Fuzzy Controller for A Car-like Mobile Robot,” *SICE Annual Conference 2010 August 18-21, 2010, The Grand Hotel, Taipei, Taiwan.*

[4] Prof. D. J. Bonde , Rohit S. Shende, Ketan S. Gaikwad, Akshay S. Kedari,Amol U. Bhokre, “Automated Car Parking System Commanded by Android Application,” *D. J. Bonde et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3001-3004*

[5] T. Bhanusri1 K.Prabhakara Rao2, “Advanced Car Parking System with GSM Supported Slot Messenger,” *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 10, Issue 1, Ver. II (Jan - Feb. 2015), PP 14-18*

[6] Shihong Qin, Xiangling Yao, “An intelligent parking system based on GSM module”, *Appl. Math. Inf. Sci. 7, No. 1L, 55-59 (2013)*

[7] I.E. Paromtchik and C. Laugier, “Autonomous Parallel Parking of a Nonholonomic Vehicle”, *IEEE Intelligent Vehicles Symposium, NJ*

[8] F. Gbmez-Bravo, F. Cuesta, A. Ollera, “Autonomous parking and navigation by using soft-computing techniques”

[9] A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators by G.W. Lucas

[10] Daniel Howley, Technology Reporter (February 13, 2016), “Self-Driving Cars Are Coming Soon to a Highway Near You,” Yahoo News. Retrieved from “https://in.news.yahoo.com/self-driving-cars-are-coming-soon-to-a-highway-221223190.html”

[11] A. Albagul, K. Alsharef, M. Saad, Y. Abujeela, “Design and Fabrication of an Automated Multi-level Car Parking System,” Manufacturing Engineering, Automatic Control and Robotics.

[12] Automated car parking system, https://www.youtube.com/watch?v=QZLfv8u2pyU

[13]Smart Car Parking Lot Management System,

https://www.youtube.com/watch?v=56AiTPYecTM

[14] Automatic Car Parking Indicator System,

http://www.projectsof8051.com/automatic-car-parking-indicator-system/

[15]Automated parking system, https://en.wikipedia.org/wiki/Automated_parking_system

[16]Figure of automatic parking of car, http://krishnaparkinfracon.com/wp-content/uploads/2012/05/multilevel-car-parking-systems.jpg

[17]Figure of manual parking of car, http://www.myparkingsign.com/blog/wp-content/uploads/Bad-parking-2.jpg

[18] Pin out of Atmega 16, http://logic-bratsk.ru/radio/micro/atmel_pic/at_dip40.htm

[19] LCD datasheet,

http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet

[20] L293D datasheet,

http://www.engineersgarage.com/electronic-components/l293d-motor-driver-ic

[21] 4X3 keypad datasheet,

http://microforbetterlive.blogspot.in/2009/05/access-4x3-keypad.html