CAR SAFETY SYSTEM

Embedded Solution for Voice Recognition Based Vehicle Locking, Tracking and Speed Control

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

In

Electronics & Communication Engineering

By

Vrushabh Sanghavi (09BEC115)
Suvigya Tripathi (09BEC094)

Under the guidance of

Prof. Dhaval Shah

Department of Electronics & Communication Engineering
Institute of Technology
Nirma University
Ahmedabad-382 481
2012-2013

Declaration

This is to certify that

1. The Project work comprises our orginal work towards the degree of Bachelor of Technology in Electronics and Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgement has been made in the text to all other material used.

- Vrushabh Sanghavi
- Suvigya Tripathi

.

Certificate

This is to certify that the Major Project entitled as " **CAR SAFETY SYSTEM: Embedded Solution for Voice Recognition Based Vehicle Locking, Tracking and Speed Control**" submitted by **Vrushabh Sanghavi and Suvigya Tripathi**, towards the partial fulfilment of the requirements for the degree of Bachelor of Technology in Communication Engineering of Nirma University, Ahmedabad is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this minor project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: May 2013 Place: Ahmedabad

Guide: HOD, EE:

(Prof. Dhaval Shah) (Dr. P.N. Tekwani)

Section Head

(Dr. D.K. Kothari)

Acknowledgement

We would like to begin by thanking Professor Dhaval Shah, our project guide has been a wonderful advisor, providing us with support, encouragement, and an endless source of ideas. His breadth of knowledge and his enthusiasm for research amazes and inspires us. We thank him for the countless hours he has spent with us.

We would like to express our gratitude and sincere thanks to Dr. P.N. Tekwani Head of Electrical Engineering Department for allowing us to undertake this project work and for his guidelines during the review process.

We wish to thank Robokits India Ltd. for providing us some unreachable resources and their incredible guidance for working on the project.

Last, but not the least, We would like to thank our batch mates for the help and inspiration they extended.

- Vrushabh Sanghavi (09BEC115)

- Suvigya Tripathi (09BEC094)

Abstract

There are many people who consider fuel consumption, comfort and price to be the important factors to consider when purchasing a vehicle. But what is your main priority when using a vehicle? If the main priority is to arrive at your final destination safely, then considering safety features as a top priority when purchasing a vehicle makes sense. As vehicles represent large investments and are a critical part of operations, businesses need the ability to protect these assets and optimise their usage. The given Car Safety System provide comprehensive tracking and security controls to deliver unparalleled fleet management and extend real-time asset management systems across the transportation fleet.

In times of rapid development of technology, there is increase in day to day theft of cars. Tracking of cars and its remote locking is not used commercially. The project includes safety systems like vehicle tracking, speed and headlight control known as daytime running light. It is based on hardware solution for voice recognition, GSM module and GPS module controlled using Arduino Controller.

GPS Module is a tracking device that provides an easy way to keep track of your car or truck and get information about its location and past history. A basis for remote vehicle monitoring is a small GPS hardware device mounted inside the car, truck or ships. The device is actually an on board Computer with GPS and wireless communication capabilities that transfers all relevant information from vehicle to the fleet management center or to the registered cell phone number. GPS, which stands for Global Positioning System, is a satellite navigation system that can ascertain the latitude and longitude of a GPS receiver device on the vehicle. The GPS consists of more than two dozen global positioning satellites orbiting the earth. Each satellite transmits radio signals, which can help to determine the location, speed and direction of travel of users equipped with GPS receivers. To ensure that the whole world is covered by the constellation of the GPS satellites, they are so arranged that four satellites are positioned in each of six orbital planes. The inbuilt GSM/GPRS Module transfers the data received from satellite to web server and through the unique software application user can track the live or historical data or vehicle. Voice recognition systems already exist in some higher-end vehicles, where you can use them to control the climate, audio, cell-phone, and navigation systems. Early versions were cumbersome to use and had difficulties recognizing voice commands, but the technology has made great strides. Some voice-recognition systems are now used with Bluetooth technology, which pairs up your cellular phone to the cars audio

system. Using voice commands instead of buttons, knobs, and touch screens should reduce driver distraction, which could in turn reduce accidents.

Even when the high beam is warranted by prevailing conditions, drivers generally do not use them. There have long been efforts, particularly in America, to devise an effective automatic beam selection system to relieve the driver of the need to select and activate the correct beam as traffic, weather, and road conditions change. A daytime running light (DRL) is an automotive lighting safety device on the front of a motor vehicle. Installed in pairs, DRLs automatically switch on when a vehicle is moving forward typically emitting white, yellow or amber light to increase the visibility of a vehicle during daylight conditions. The safety or daytime running lights are a low-cost method to reduce daytime crashes. They are especially effective in preventing daytime head-on and front-corner collisions by increasing vehicle visibility and making it easier to detect approaching vehicles from farther away.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Development in the fields of Embedded Systems are making it feasible to deploy a wide range of Real Time Operating Systems. This project on Car Safety System presents a small replica of the car security, locking and tracking system.

The setup including Voice Recognition module, GSM module, GPS module, Proximity Sensor and Arduino Micro-controller (Atmega 128) is installed in a vehicle (Car). Voice call is made remotely to the GSM module installed in the car which is followed by the voice recognition (VR) module. The user speaks a 3 digit PIN and if it matches the predefined PIN, the user is authenticated. The user is entitled to lock the car and seek its position by giving commands. The GPS module continuously tracks the position in terms of latitudes and longitudes. When the user calls for the position, an SMS is sent to the registered cell phone. The message can also be sent to police station if needed.

A proximity sensor is connected to the chasis which senses a metal contact mounted on one of the wheel. This gives the speed of the car. Whenever speed exceeds a threshold value, only then will the high beam be turned on. High beam leads to many accidents, especially in urban areas.Therefore, this system allowes to turn on high beam only at high speed. A similar system is developed for Traffic Control during excessive speeding. If the speed exceeds the limit, a message is sent to RTO. Hence, the project provides a complete security system for an automobile ranging from theft safety, position retreival, speed control and headlight beam control.

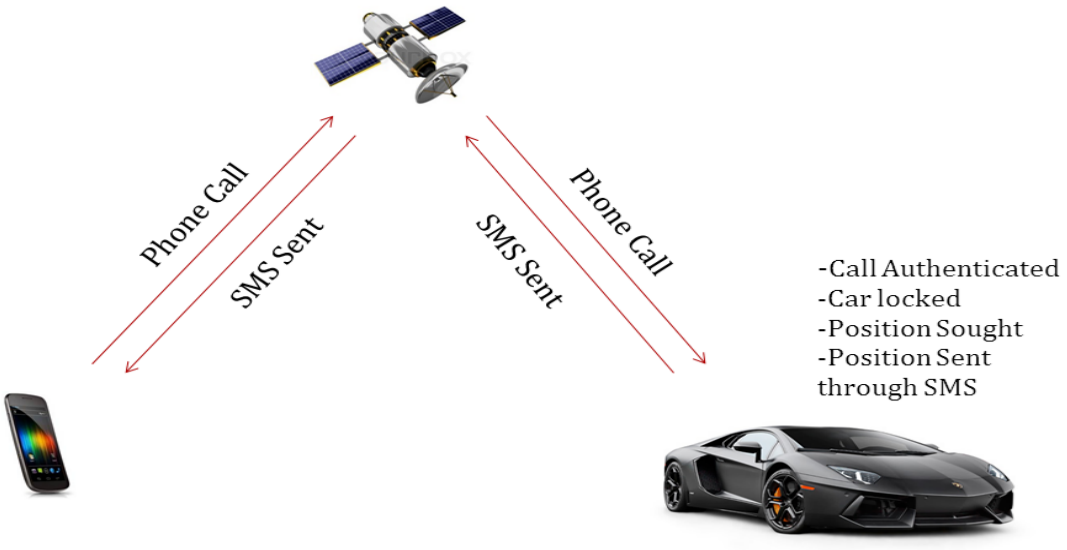

Figure 1.1: Overview of Project

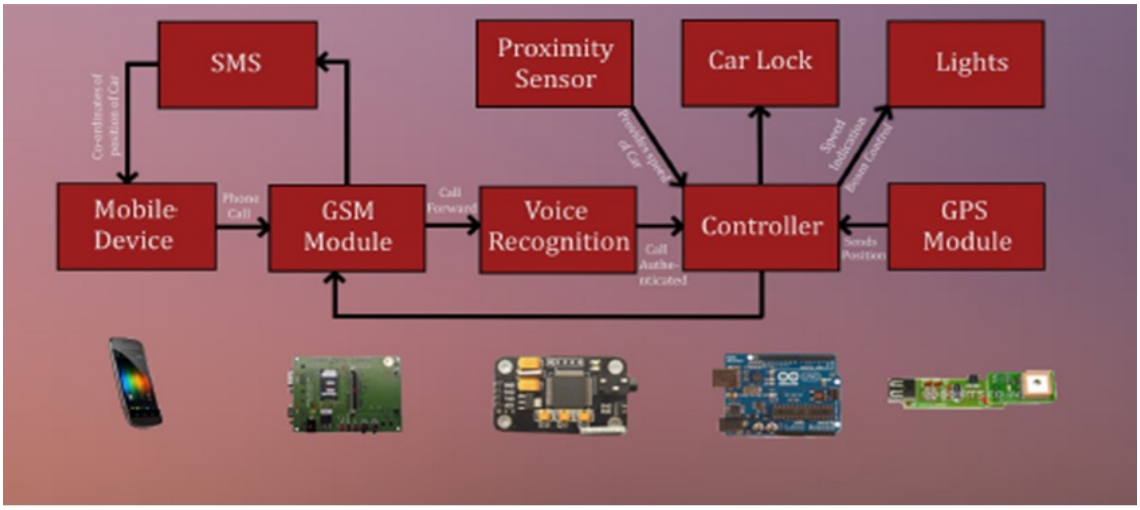

Figure 1.2: Block Diagram of the System

1.2 Why such a system

- Controls device remotely.
- Security- PIN known only to user.
- Ease of access to the places of extreme condition.

1.3 Why Hardware Solution

- Software Solutions are very common.
- High accuray and no software bugs.
- No initial samples , database or code book required.

1.4 Modules Used in the Project

- GSM module
- Voice Recognition Module
- Mirco-controller (Arduino Mega (Atmel 128))
- Global Positioning System
- Proximity Sensor

Chapter 2

GSM Module: Board Specifications

The GSM/GPRS Modem USB uses SIM900 based Quadband 850/900/1800/1900MHz GSM / GPRS modem. It can be connected to USB directly or to a micro-controller UART using TTL levels. It accepts external voltage 5- 2VDC 1A or 4VDC 2A as input. SIM900 features GPRS multi-slot class 10/ class 8 and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

With a tiny configuration of 24mm x 24mm x 3mm, SIM900 can meet almost all the space requirements in the application. The physical interface to the mobile application is a 68-pin SMT pad, which provides all hardware interfaces between the module and controller. One audion channel includes a microphone input and a speaker output.

The SIM900 is integrated with the TCP/IP protocol, extended TCP/IP AT commants are developed to use these protocols easily for data transfer applications. This module is designed with power saving technique so that the current consumption is as low as 1.5mA in sleep mode.

2.1 SIM900 functional diagram

The following figure shows a functional diagram of the SIM900 and illustrates the mainly functional part:

- The GSM baseband engine
- Flash and SRAM

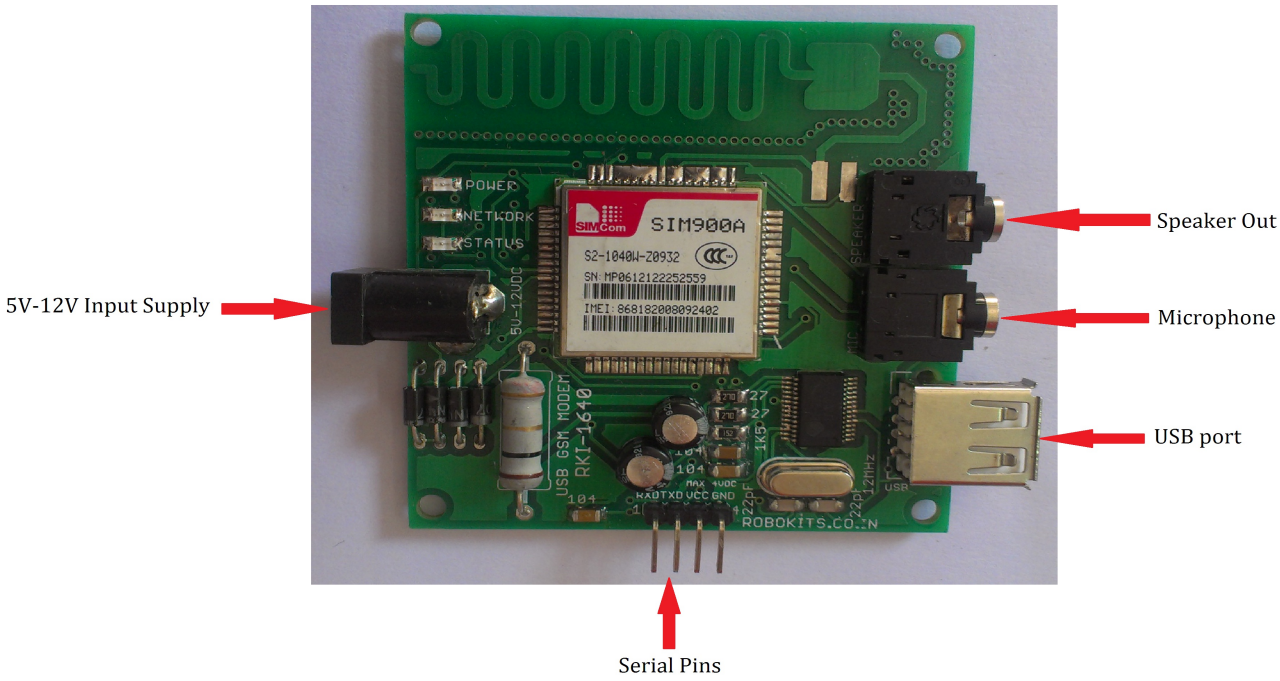

Figure 2.1: Front Side of GSM module

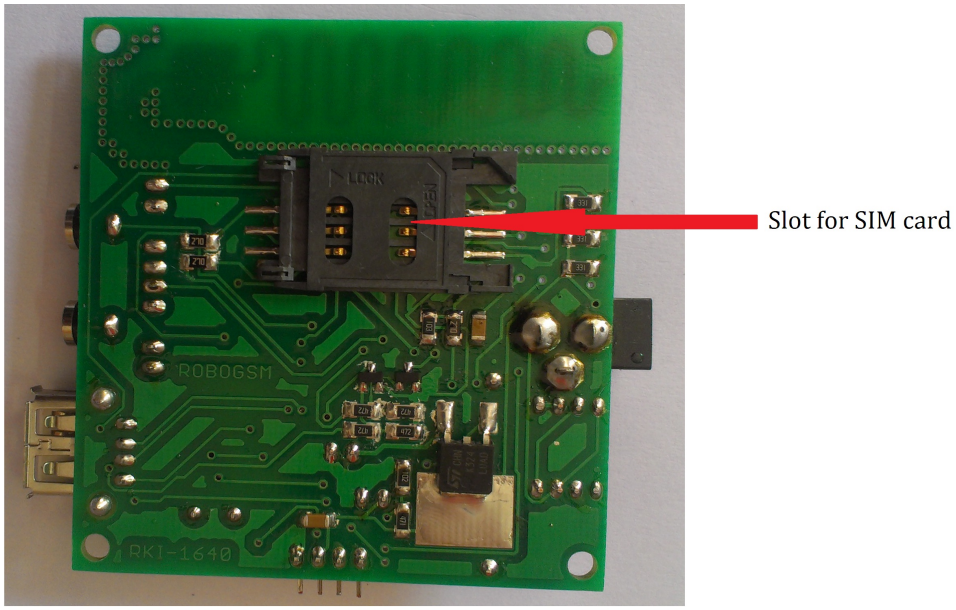

Figure 2.2: Back Side of GSM module

- The GSM radio frequency part
- The antenna intrfaces
- The other interfaces

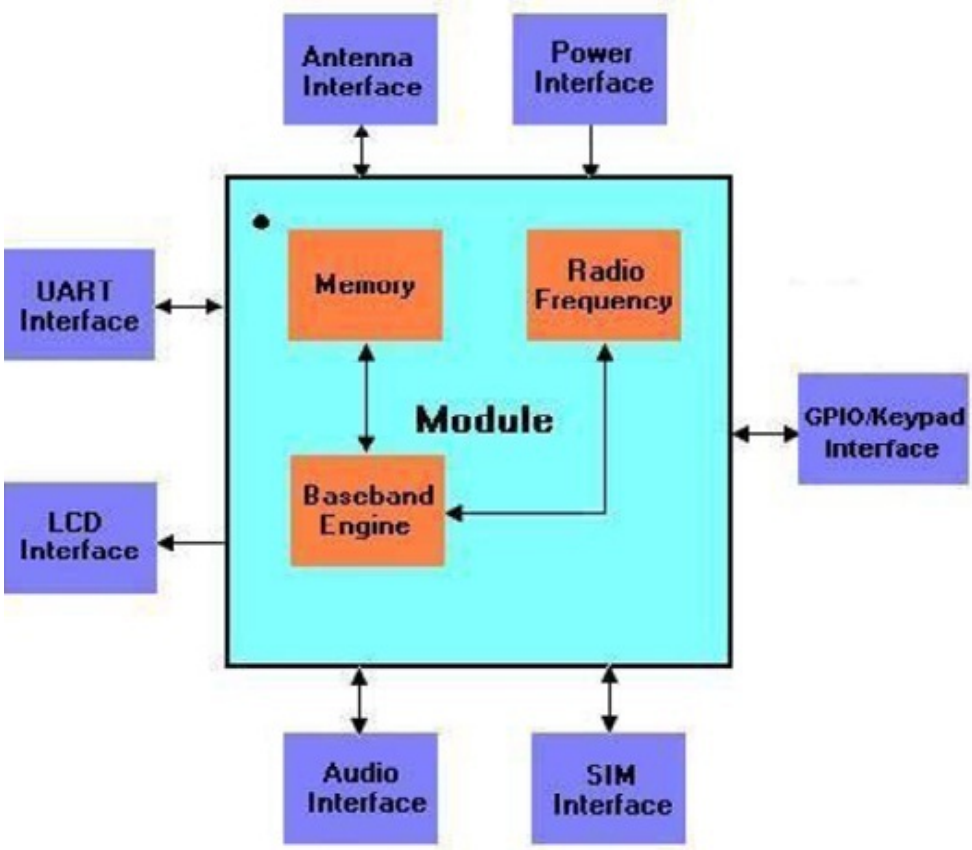

Figure 2.3: SIM900 functional block diagram

2.2 Features

- Accepts commands through Onboard USB serial
- Also can accept commands from microcontroller UART

- Works on 5V-12V 1A SMPS supply
- Speaker and Microphone Jacks onboard for voice
- Onboard PCB antenna which gives it a small size
- Onboard Power, Network and status LEDs for indication
- Can do basic functions which can be done via normal phone like calling, send sms, access data through GPRS etc.
- Can accept commands through Hyperterminal

2.3 AT commands

In any application, controlling device controlls the GSM engine by sending AT commands via its serial interface. The "AT" prefix must be set at the beginnig of each Command line. To terminate a Commad line enter (CR). Commands are usually followed by a response.

The following AT commands were used CAR SAFETY SYSTEM project.

2.3.1 ATA

It is used to answer an incoming call.
Response in case of a voice call if it is successfully connected: **OK**
Response if no connection is established: **NO CARRIER**

2.3.2 ATS0

It is used to set the number of rings before automatically answering the call.
ATS0=n: This parameter setting determines the number of rings before auto-answer. The response is: **OK**

2.3.3 AT+CMGF

It is used to select the message format.
AT+CMGF=1: Sets the module in text mode.

2.3.4 AT+CMGS

It is used to send a message to the registered number.
AT+CMGS="Mobile number"
The mobile number should be written in quotation marks. On sending this command, one field appears where the message can be typed. The message will be sent if it is terminated with **1A 0D (in hexadecimal)**.

Chapter 3

Voice Recognition Module

3.1 Introduction

The module could recognize the voice. It receives configuration commands or responds through serial port interface. With this module, we can control the car or electrical devices by voice.

This module can store 15 pieces of voice instructions. Those 15 pieces are divided into 3 groups, with 5 in one group. First we should record the voice instructions group by group. After that, we should import one group by serial command before it could recognize the 5 voice instructions within that group. If we need to implement instructions in other groups, we should import the group first. This module is speaker independent. If any of the friend speaks the voice instruction instead of us, it may not identify the instruction. Note that speaker dependence requires strictly good MIC.

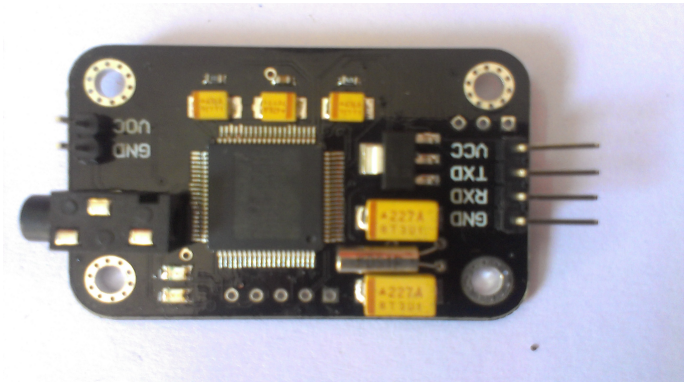

Figure 3.1: Command format

3.2 Technical Parameters

- Voltage: 4.5-5.5V
- Current: <40mA
- Digital Interface: 5V TTL level UART interface
- Analog Interface: 3.5mm mono-channel microphone connector + microphone pin interface
- Recognition accuracy: 99% (under ideal environment)

3.3 Serial Command

This module can be configured by sending commands via serial port. Configuration will not be erased after powered off. Its interface is 5V TTL. The serial data format: 8 data bits, no parity, 1 stop bit. The default baud rate is 9600 and baud rate can be changed.
Command format is "Head+key". "Head" is 0xaa, and "Key" is given in the table.

For the first time use, we need to do some configurations.

- Select the serial baud (default 9600)
- Select the communication mode: Common Mode or Compact Mode
- Recording five instructions of the first group (or 2nd or 3rd as required)
- Import the group we need to use (only recognizes 5 instructions within one group at the same time)

After all the setting above, we can speak or send voice instruction to it. If idetified successfully, result will be returned via serial port in the format: **group number + command number**. For example, return **Result: 11** means identified the first command of group 1. If voice instruction is recorded, each time after we power it on, we need to import the group before letting it to identify the voice instructions.

Key (HEX format)	Description	Respond in Common Mode	Respond in Compact Mode
0x00	Enter into "Waiting" state	"Waiting! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x01	Delete the instructions of group 1	"Group1 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x02	Delete the instructions of group 2	"Group2 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x03	Delete the instructions of group 3	"Group3 Deleted ! \ n" : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x04	Delete the instructions of all the 3 groups	" All Groups Deleted !\n " : successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x11 0x12 0x13	Begin to record instructions of group	"ERROR! \ n" : Instruction error "START \ n" : Ready for recording, you can speak now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group1 finished! \ n" : finish recording group 1	0xe0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x46 : finish recording group 1
0x21	Import group 1 and be ready for voice instruction	"Group1 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x22	Import group 2 and be ready for voice instruction	"Group2 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x23	Import group 3 and be ready for voice instruction	"Group3 Imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xcc : Successful 0xe0 : Instruction error 0xe1 : Importing voice group failed
0x24	Query the recorded group	"Used group:0\n" : No group is recorded "Used group:1\n" : Group 1 is recorded "Used group:2\n" : Group 2 is recorded "Used group:3\n" : Group 3 is recorded "Used group:12\n" : Group 1 and Group 2 are recorded "Used group:13\n" : Group 1 and Group 3 are recorded "Used group:23\n" : Group 2 and Group 3 are recorded "Used group:123\n" : All the 3 groups are recorded "ERROR! \ n" : Instruction error	0x00 : No group is recorded 0x01 : Group 1 is recorded 0x02 : Group 2 is recorded 0x04 : Group 3 is recorded 0x03 : Group 1 and Group 2 are recorded 0x05 : Group 1 and Group 3 are recorded 0x06 : Group 2 and Group 3 are recorded 0x07 : All the 3 groups are recorded 0xe0 : Instruction error

Figure 3.2: Command format

3.4 Status LEDs

3.4.1 Recording stage

1. Record indication: D1 (REd) flashes 3 times within 600ms, then off for 400ms, and then flashes quickly for 4 times within 600ms. This show that recording is over.

2. Begin to speak: D1(RED) is off for 400ms, and then is on. Voice during the time while D1(RED) is on will be recorded by this module.

3.4.2 Waiting Mode

In waiting mode, D2 (ORANGE) is off, and D1 (RED) is on for 80ms every other 200ms, fast blinking. Inthis mode, it doesnt recognize voice command, only waiting for serial commands.

3.4.3 Recognition stage

In identification stage, D2 (ORANGE) is off, and D1 (RED) is on for 100ms every other 1500ms, slow flashing. In this stage, the module is processing received voice signal, and if matching, it will send the result immediately via serial port.

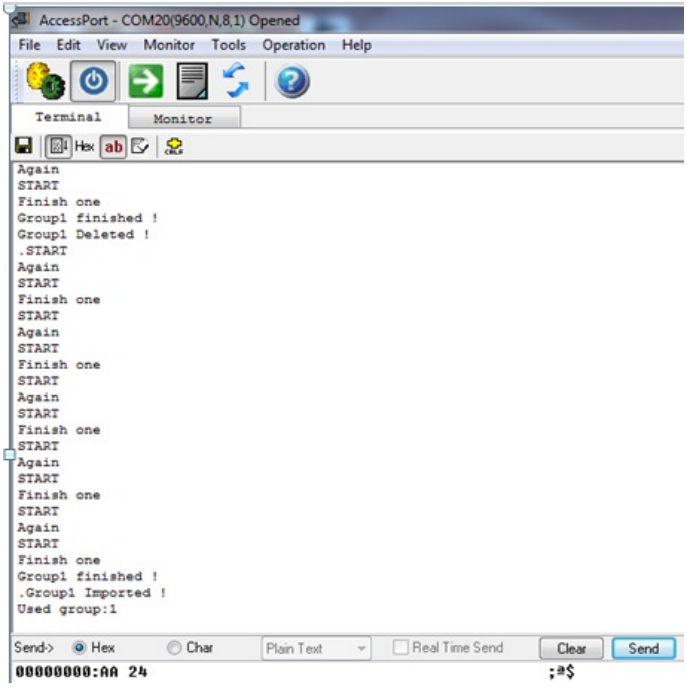

Figure 3.3: Recording Voice

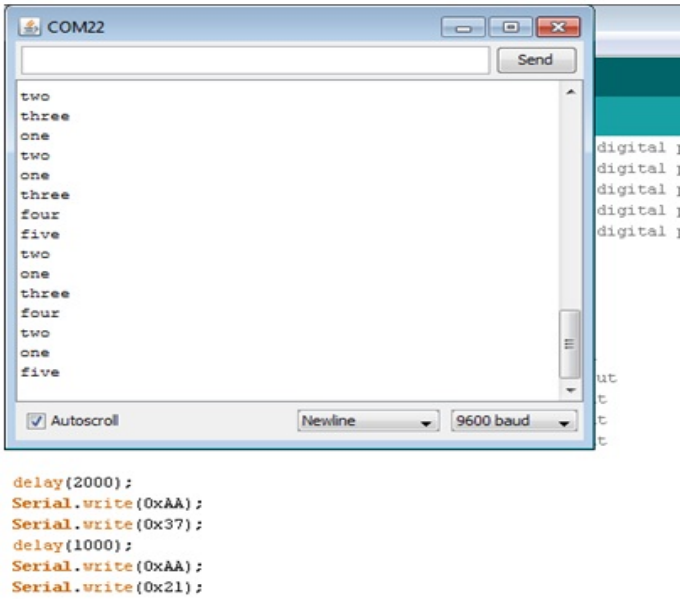

Figure 3.4: Detecting Voice

Chapter 4

Micro-Controller: Arduino Mega (Atmel128)

4.1 Introduction

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles,

where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

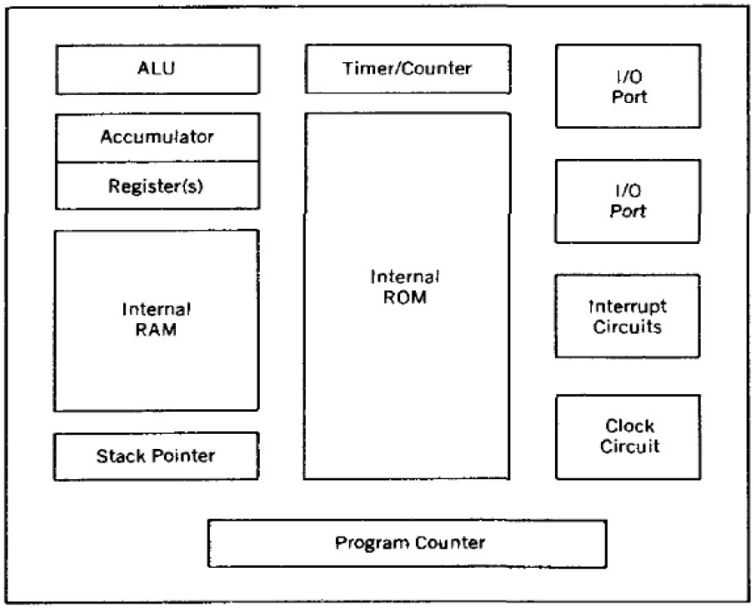

Figure 4.1: Block Diagram of Microcontroller

4.1.1 Arduino Mega

Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an IC serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the megaAVR series

of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits.

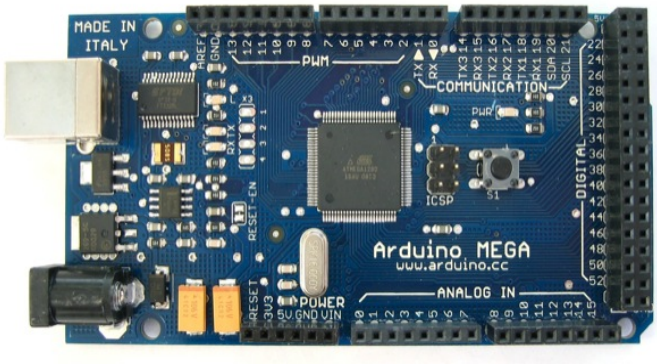

Figure 4.2: Arduino Mega 1280

4.2 Software

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface.[citation needed] A program or code written for Arduino is called a sketch.

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier.

Users only need define two functions to make a runnable cyclic executive program:

- **setup()**: a function run once at the start of a program that can initialize settings
- **loop()**: a function called repeatedly until the board powers off

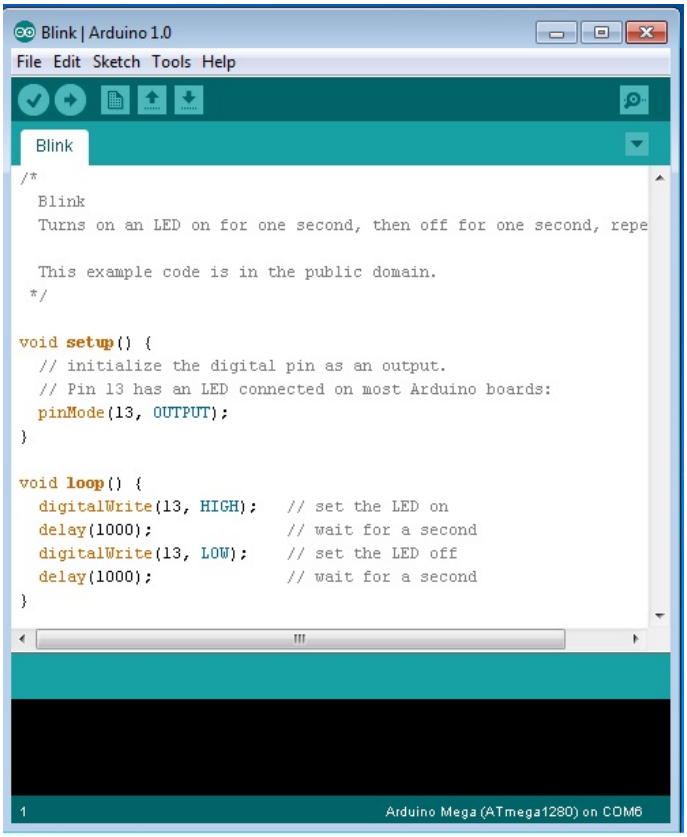

Figure 4.3: Arduino IDE

Chapter 5

Global Positioning System

5.1 Introduction

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver.

Figure 5.1: GPS Module

5.2 Basic Concepts of GPS

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include the time the message was transmitted satellite position at time of message transmission.

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light. Each of these distances and satellites' locations define a sphere. The receiver is on the surface of each of these spheres when the distances and the satellites' locations are correct. These distances and satellites' locations are used to compute the location of the receiver using the navigation equations. This location is then displayed, perhaps with a moving map display or latitude and longitude; elevation information may be included. Many GPS units show derived information such as direction and speed, calculated from position changes.

The Global Positioning System (GPS) is actually a constellation of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails). The U.S. military developed and implemented this satellite network as a military navigation system, but soon opened it up to everybody else. Each of these 3,000- to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making two complete rotations every day. The orbits are arranged so that at any time, anywhere on Earth, there are at least four satellites "visible" in the sky.

A GPS receiver's job is to locate four or more of these satellites, figure out the distance to each, and use this information to deduce its own location. This operation is based on a simple mathematical principle called trilateration. Although four satellites are required for normal operation, fewer apply in special cases. If one variable is already known, a receiver can determine its position using only three satellites. For example, a ship or aircraft may have known elevation. Some GPS receivers may use additional clues or assumptions such as reusing the last known altitude, dead reckoning, inertial navigation, or including information from the vehicle computer, to give a (possibly degraded) position when fewer than four satellites are visible.

5.3 NMEA Protocol

NMEA 0813 Data Protocol is defined by the National Marine Electronics Association. It uses a simple ASCII, serial communication protocol that defines how data is transmitted in a sentence from one sender (satellite) to multiple receivers at a time. Typical Baud rate is 9600 with no handshake taking place.

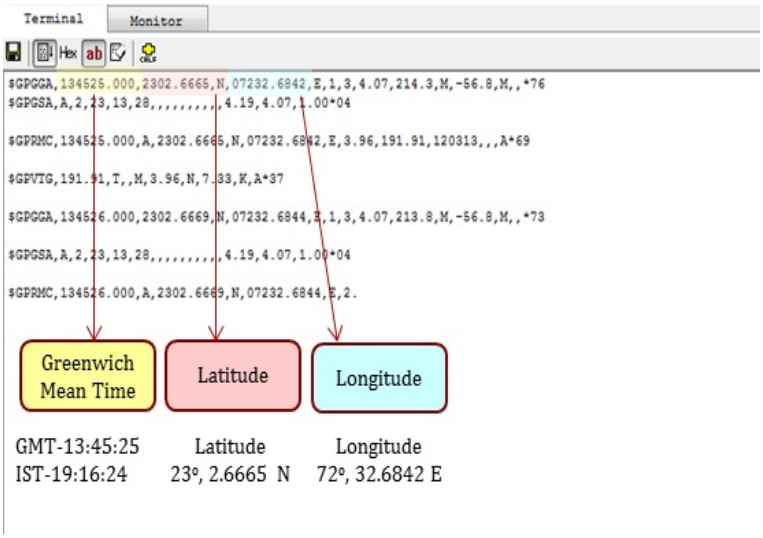

Figure 5.2: NMEA Protocol

5.4 Application

- **Astronomy:** Both positional and clock synchronization data is used in Astrometry and Celestial mechanics calculations.
- **Cellular telephony:** Clock synchronization enables time transfer, which is critical for synchronizing its spreading codes with other base stations to facilitate inter-cell handoff and support hybrid GPS/cellular position detection for mobile emergency calls and other applications.
- **Geofencing:** Vehicle tracking systems and pet tracking systems use GPS to locate a vehicle, person, or pet.
- **Geotagging:** Applying location coordinates to digital objects such as photographs and other documents for purposes such as creating map overlays.
- GPS Aircraft Tracking
- **Navigation:** Navigators value digitally precise velocity and orientation measurements.
- **Robotics:** Self-navigating, autonomous robots using a GPS sensors, which calculate latitude, longitude, time, speed, and heading.
- **Tectonics:** GPS enables direct fault motion measurement in earthquakes.

Chapter 6

Proximity Switch: Speed Calculation

6.1 Introduction

Proximity Switch is a sensor able to detect the presence of nearby objects without any physical contact.A proximity switch often emits an electromagnetic field or a beam of electromagnetic radiation , and looks for changes in the field or return signal. The object being sensed is often referred to as the proximity switch's target. Different proximity switch targets demand different switchs The maximum distance that this switch can detect is defined nominal range. Proximity switches open or close an electrical circuit when they make contact with or come within a certain distance of an object. Proximity switches are most commonly used in manufacturing equipment, robotics, and security systems.

6.2 Types of Proximity

There are four basic types of proximity switches: infrared, acoustic, capacitive, and inductive.

6.2.1 Infrared proximity switches

It works by sending out beams of invisible infrared light. A photodetector on the proximity switch detects any reflections of this light. These reflections allow infrared proximity switches to determine whether there is an object nearby. As proximity switches with just a light source and photodiode are susceptible to false readings due to background light, more complex switches

modulate the transmitted light at a specific frequency and have receivers which only respond to that frequency. Even more complex proximity sensors are able to use the light reflected from an object to compute its distance from the sensor.

6.2.2 Acoustic proximity sensors

It is in principle to infrared models, but use sound instead of light. They use a transducer to transmit inaudible sound waves at various frequencies in a preset sequence, then measure the length of time the sound takes to hit a nearby object and return to a second transducer on the switch. Essentially, acoustic proximity sensors measure the time it takes for sound pulses to "echo" and use this measurement to calculate distance, just like sonar.

6.2.3 Capacitive proximity switches

It senses distance to objects by detecting changes in capacitance around it. A radio-frequency oscillator is connected to a metal plate. When the plate nears an object, the radio frequency changes, and the frequency detector sends a signal telling the switch to open or close. These proximity switches have the disadvantage of being more sensitive to objects that conduct electricity than to objects that do not.

6.2.4 Inductive proximity switches

It senses distance to objects by generating magnetic fields. They are similar in principle to metal detectors. A coil of wire is charged with electrical current, and an electronic circuit measures this current. If a metallic part gets close enough to the coil, the current will increase and the proximity switch will open or close accordingly. The chief disadvantage of inductive proximity switches is that they can only detect metallic objects.

6.3 Speed Calculation

The inductive proximity switch and a piece of metal is attached to one of the wheels of the robot. Speed is calculated by the following formula.

$$Speed = \frac{Distance}{Time} \tag{6.1}$$

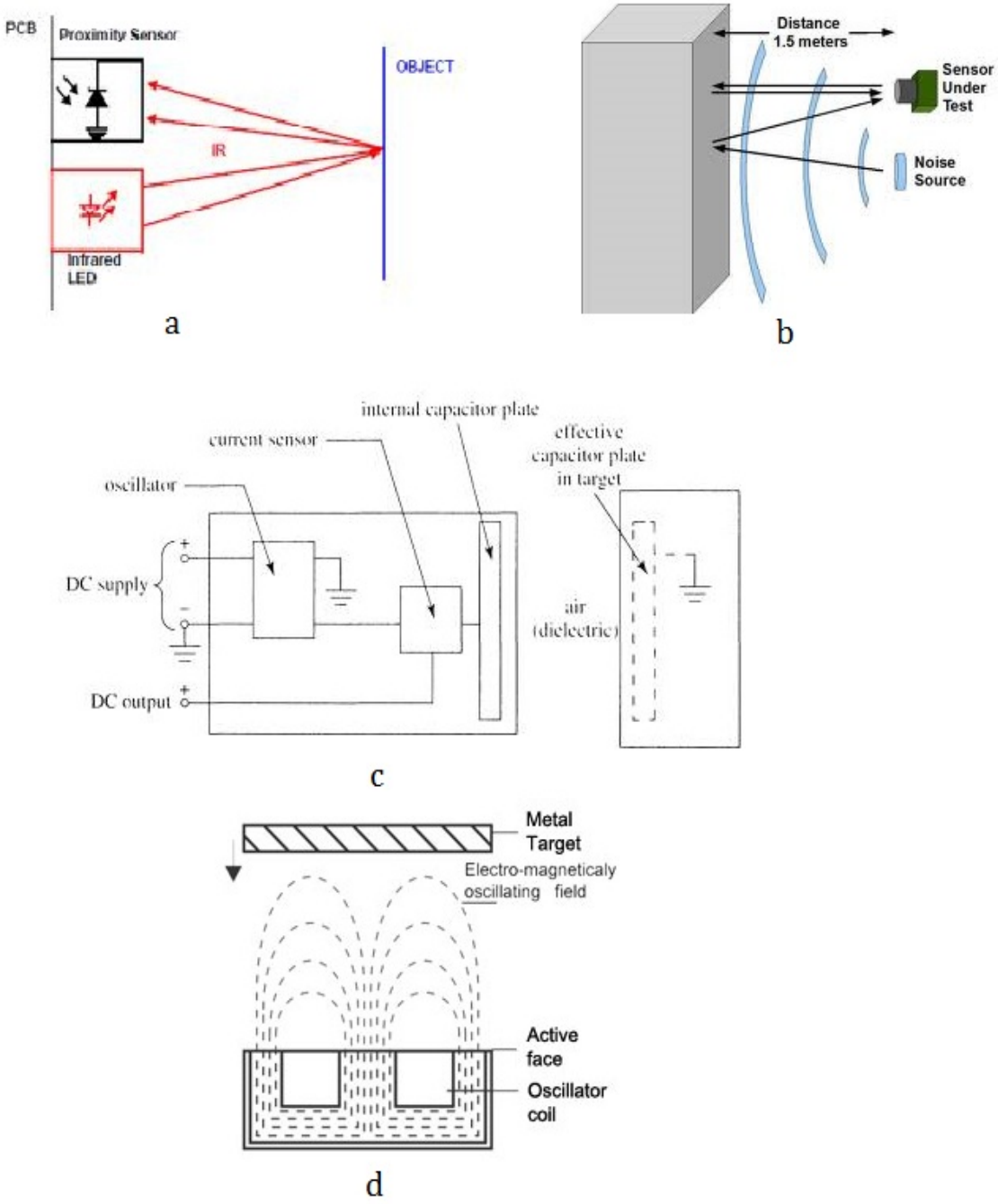

Figure 6.1: Types of proximity switches. a. Infrared b. Acoustic c. Capacitive d. Inductive

In our case, time is kept constant as 1 second. The constant time is generated using Timer Interrupt of 1 second. Distance is calculated as the distance moved in that one second which means number of time the metal is detected by the proximity in one second.

$$Distance = Number.of.times.metal.detected \times [Circumferance.of.the.wheel] \tag{6.2}$$

$$Speed = \frac{Number.of.time.metal.detected \times [Circumferance.of.the.wheel] \times [0.036km]}{1hr} \tag{6.3}$$

Chapter 7

Application of the Project

7.1 Vehicle Anti-theft Safety

This system can be used to track the vehicle in case of vehicle theft. This system provides user a voice control over his vehicle. The vehicle can be locked and traced from a distant place with the help of GSM technology. Exact location of the vehicle can be obtained in a message on the registered number. This provides complete safety of the vehicle which includes locking and tracking.

7.2 Overspeed Limiting

This system also provides a control on over speeding which may lead to severe accidents. In case, the vehicle over exceeds the predefined limit of any area in the city or breaks any rules as defined by Road Traffic Police, a SMS can be sent directly to RTO indicating the car number, driver information, speed, location etc. RTO can then trace the owner and fine him. This system helps in ensuring the traffic rules and road safety.

7.3 Vehicle Headlight Beam Contol

In major cities, there is always a problem of High headlight beam creating difficulty in driving due to glaring problem at night. This system uses speed as a parameter to control the focusing of the headlights. High beam Headlights are mostly required on highways and not in the cities. In this system, light beam will be turned high only when the vehicle crosses the city speed limit. This will prevent the problem of glaring and ensure the safety at night.

7.4 Emergency in case of accident

In case of a major accident of the vehicle, this system can be integrated with the air bag mechanism which will call or send a message to the nearest hospital with the accident location. This will help in preventing casuality and ensuring the first aid treatment as soon as possible.

7.5 Smart Homes

The voice recognition module with GSM and controller can be used to control various appliances at home when the user is at a distant place. User can moniter his house and ensure its safety while he is at a distant place. The user can control lights, fans, heaters, air conditioners etc. He can even use Voice control system for controlling television, switching of lights etc.

7.6 Robotic applications

Human access to some of the extreme condition is not possible. Hence, such places requires autonomous robots. These robots can be controlled by the human operator at a distant place by using this complete setup. The operator can manipulate the controlling of robot by sending commands through this system. This can be used in pick and place robots.

7.7 Application for physically challenged

Physically challenged find it difficult to communicate or travel by theirselves. For example, handicapped people always reqiure some to moniter them and take them to different places. This voice recognition system helps them to the go to the places without depending on others.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This report includes overview of The Car Safey System. This system includes various hardwares including GSM, GPS, Voice Recognition Module, Proximity Sensor and Micro-controller. This report provides full description of all the modules used in this project. The description includes basic idea of the modules and basic programming concepts. This project is a multipurpose system for ensuring the security of vehicle, emergency at the time of accidents, overspeed control and vehicle headlight focus control. This system includes voice recognition module providing voice security and the location will be sent only to the registered phone number. This helps in effective tracking of the vehicle.

One of the major problems faced while doing the project was sending a message to a cell phone using a GSM module and microcontroller. GSM module worked very well when accessed through hyperterminal, but caused errors when accesased through a controller. Continuous trial, research work and learning more about microcontroller helped us to solve the errors. The other problem which was encountered was during the integration of GSM module and GPS module. Sometime GPS reads and provides false readings which are to be send as a message by GSM. This problem was also solved by optimizing the microcontroller code. The major problem was faced while working with proximity sensor. This sensor was used to calculate the speed of the vehicle. For this, we required a constant time which was generated by Timer interrupt called every 1 second. This interrupt caused interruption with GSM function. GSM function has a large delay, more then 1 second, and hence gets interrupted evrey second. We came across a solution of turning

off the timer during a function call. This solution worked well with both, timer and GSM function.

Encountering various problems both in hardware and software, helped us to learn more about the various modules we used. This project also enhanced our coding skills and helped us in developing more scientific and technical approach for handling errors and finding solutions.

8.2 Future Work

Uptil now we have developed a prototype and tested for limitied resources and limited conditions. In future, work can be done to improve voice recognition and making it strictly user dependent. The voice database of few more person related to the user can be stored which will be helpful in authentication purpose by other members too. The prototype includes various diffecent hardware modules, but in future, all the hardwares can be embedded on a single board. This will also reduce the cost of product. The application can also be extended to home applications, industrial applications, robotics etc.

Chapter 9

Refrence

1. AT commands reference guide-Multitech systems
2. www.robokits.co.in
3. SIM900 ATC Command Manual
4. SIM900 ATC Command Set
5. www.elechouse.com
6. HM2007L Datasheet
7. http://arduino.cc/en
8. www.prolifictech.com
9 Datasheet Arduino
10. www.simplelabs.co.in
11. Sohilp.blogspot.in
12. www.datasheet4u.com

Arduino Code

```
   #include <string.h>
#include <ctype.h>
#include <avr/io.h¿
#include <avr/interrupt.h>


   /*gps init*/
int gsml1=23;
int gsml2=25;
int gsml3=27;
int gsml4=29;
int gpsl1=31;
int vrl1=33;
int vrl2=35;
int vrl3=37;
int el=39;
int flag1=0;
int flag2=0;
// int flag3=0;
int swi=45;
int d8=47;
int d9=49;
int d10=51;
int d11=53;
int time=0;
int delay1=0;
int l1= 4;
int l2=5;
int r1=6;
int r2=7;
byte r=0;
byte u=0;
int o=0;
byte w[]=0,0,0,0;
byte f[]=0,0,0,0;
int i=0;
int sensor=9;
int val=0;
int beam1a=43;
int beam1b=41;
int beam2a=39;
int beam2b=37;
int sw=0;
int seconds;
int cntr;
float spd;
int spd2;
float diff;
int speeed;
int ledPin = 13; // LED test pin
int byteGPS=-1;
char linea[300] = "";
char comandoGPR[7] = "$GPRMC";
int cont=0;
int bien=0;
int conta=0;
int indices[13];
int x=0;
int y=0;
byte com=0;
byte rec=0;
char lat[10]=0,0,0,0,0,0,0,0,0,0;
char lat1[10]=0,0,0,0,0,0,0,0,0,0;
char lon[10]=0,0,0,0,0,0,0,0,0,0;
char lon1[10]=0,0,0,0,0,0,0,0,0,0;
char ab=0;
char bc=0;
int k=0;
int z=0;
int g=0;


   /*gsm initi*/
char a[10];
char b=0;
char c=0;
char d=0;
char e=0;
int timesToSend = 1; // Numbers of SMS to send
int count = 0;


   void setup()
{
Serial1.begin(9600);
Serial2.begin(9600);
Serial3.begin(9600);
Serial.begin(9600);
pinMode(ledPin, OUTPUT); // Initialize LED pin
pinMode(gsml1, OUTPUT);
pinMode(gsml2, OUTPUT);
```

```
pinMode(gsml3, OUTPUT);
pinMode(gsml4, OUTPUT);
pinMode(gpsl1, OUTPUT);
pinMode(vrl1, OUTPUT);
pinMode(vrl2, OUTPUT);
pinMode(vrl3, OUTPUT);
pinMode(el, OUTPUT);
pinMode(d8, INPUT);
pinMode(d9, INPUT);
pinMode(d10,INPUT);
pinMode(d11,INPUT);
pinMode(beam1a,INPUT);
pinMode(beam1b,INPUT);
pinMode(beam2a,INPUT);
pinMode(beam2b,INPUT);
pinMode(swi,INPUT);
pinMode(l1, OUTPUT);
pinMode(l2, OUTPUT);
pinMode(r1, OUTPUT);
pinMode(r2, OUTPUT);
pinMode(13, OUTPUT);
pinMode(sensor, INPUT);
timer_init();
for (int i=0;i < 300;i++) // Initialize a buffer for received data
{
linea[i]=' ';
}
Serial3.write(0xAA);
Serial3.write(0x37); //changes to compact mode
Serial3.write(0xAA);
Serial3.write(0x21);
}
void timer_init(void)
{
// initialize Timer1
cli(); // disable global interrupts
TCCR1A = 0; // set entire TCCR1A register to 0
TCCR1B = 0; // same for TCCR1B
// set compare match register to desired timer count:
OCR1A = 15624;
// turn on CTC mode:
TCCR1B —= (1 << WGM12);
// Set CS10 and CS12 bits for 1024 prescaler:
TCCR1B —= (1 <<CS10);
TCCR1B —= (1 << CS12);
// enable timer compare interrupt:
TIMSK1 —= (1 <<OCIE1A);
sei(); // enable global interrupts
}
ISR(TIMER1_COMPA_vect)
{
seconds++;
//if(seconds==1)
display_speed();
cntr=0;
spd=0;
spd2=0;
speeed=0;
} void display_speed(void)
{
spd=(cntr*33)*0.036*10;
Serial.println("speed : ");
Serial.println(spd);
Serial.print(" Km/h ");
if(spd > 35)
{
digitalWrite(beam1b, LOW);
digitalWrite(beam2b, LOW);
digitalWrite(beam1a, HIGH);
digitalWrite(beam2a, HIGH);
}
else
{
digitalWrite(beam1b, HIGH);
digitalWrite(beam2b, HIGH);
digitalWrite(beam1a, LOW);
digitalWrite(beam2a, LOW);
}
}
void sense(void)
{
val=digitalRead(sensor);
if(val==LOW)

while(val==LOW)
{
val=digitalRead(sensor);
}
cntr+=1;
}
}
void gsm1()
{
Serial1.print("AT"); // sets the SMS mode to text
Serial1.print(0x0D);
Serial1.print(0x0A);
delay(500);
a[0]=Serial1.read();
a[1]=Serial1.read();
a[2]=Serial1.read();
if(Serial1.available() > 0)
{
Serial.println(a);
```

```
Serial.println(a[1]);
Serial.println(a[2]);
digitalWrite(gsml1, HIGH);
delay(2000);
digitalWrite(gsml1, LOW);
Serial1.flush();
}
Serial1.println("AT+CMGF=1"); // sets the SMS mode to text
Serial1.print(0x0D);
Serial1.print(0x0A);
delay(100);
delay(1500);
Serial1.println("AT+CMGS=¨+919426347767¨"); // send the SMS number
delay(2000);
Serial1.print("CAR LOCKED"); // the SMS body
Serial.println("CAR LOCKED"); // the SMS body
Serial1.print(lat); // the SMS body
Serial.println(lat); // the SMS body
Serial1.print(lat1); // the SMS body
Serial.println(lat1); // the SMS body
Serial1.print(lon); // the SMS body
Serial.println(lon); // the SMS body
Serial1.print(lon1); // the SMS body
Serial.println(lon1); // the SMS body
Serial1.print(char(26));
Serial1.print(char(13));
digitalWrite(gsml4,HIGH);
delay(15000);
}
count++;
}


   void gps()
{
digitalWrite(gpsl1, HIGH);
byteGPS=Serial2.read(); // Read a byte of the serial port
if (byteGPS == -1)
{ // See if the port is empty yet
delay(100);
}
else
{
linea[conta]=byteGPS; // If there is serial port data, it is put in the buffer
conta++;
Serial.print((byte)byteGPS);
if (byteGPS==13) // If the received byte is = to 13, end of transmission
{
digitalWrite(gpsl1, LOW);
cont=0;
bien=0;
for (int i=1;i < 7;i++) // Verifies if the received command starts with $GPR
{
if (linea[i]==comandoGPR[i-1])
{
bien++;
}
}
if(bien==6) // If yes, continue and process the data
{
for (int i=0;i < 300;i++)
{
if (linea[i]==',') // check for the position of the "," separator
indices[cont]=i;
cont++;

if (linea[i]=='*') // ... and the "*"
indices[12]=i;
cont++;
}
}
Serial.println(""); // ... and write to the serial port
Serial.println("");
Serial.println("————————");
for (int i=0;i < 12;i++)
{
switch(i)
{
case 0 :Serial.print("Time in UTC (HhMmSs): ");break;
case 1 :Serial.print("Status (A=OK,V=KO): ");break;
case 2 :Serial.print("Latitude: ");
for(int g=indices[2],k=0;g < (indices[3]-1);g++,k++)
{
lat[k]=linea[g+1];
}
break;
case 3 :Serial.print("Direction (N/S): ");
for(int g=indices[3],k=0;g < (indices[4]-1);g++,k++)
{
lat1[k]=linea[g+1];
}
break;
case 4 :Serial.print("Longitude: ");
for(int g=indices[4],k=0;g < (indices[5]-1);g++,k++)
{
lon[k]=linea[g+1];
}
```

```
break;
case 5 :Serial.print("Direction (E/W): ");
for(int g=indices[5],k=0;g < (indices[6]-1);g++,k++)
{
lon1[k]=linea[g+1];
}
break;
case 6 :Serial.print("Velocity in knots: ");break;
case 7 :Serial.print("Heading in degrees: ");break;
case 8 :Serial.print("Date UTC (DdMmAa): ");break;
case 9 :Serial.print("Magnetic degrees: ");break;
case 10 :Serial.print("(E/W): ");break;
case 11 :Serial.print("Mode: ");break;
case 12 :Serial.print("Checksum: ");break;
}
for (int j=indices[i];j < (indices[i+1]-1);j++)
{
Serial.print(linea[j+1]);
}
Serial.println("");
}
Serial.println("————————");
}
conta=0; // Reset the buffer
for (int i=0;i < 300;i++){ //
linea[i]=' ';
}
}
}
}


   void loop() // run over and over agai
n {
sw=digitalRead(swi);


   if(sw==LOW)
{
TIMSK1=0x00;
sw=digitalRead(swi);
gps();
while(Serial3.available())
{
com = Serial3.read(); //read data on serial
switch(com)
{
case 0x14:
digitalWrite(vrl1, HIGH); // correct pin
delay(2000);
digitalWrite(vrl1,LOW);
flag1++;
break;
case 0x12:
if(flag1==1 || flag1==2)
{
digitalWrite(gsml2, HIGH); // turn on led2
delay(1000);
//gsm2();
digitalWrite(gsml2,LOW);
flag2++;
}
if(flag1 > 2)
{
digitalWrite(gsml1, HIGH);
delay(2000);
digitalWrite(gsml1,LOW);
}
break;
case 0x13:
digitalWrite(gsml3, HIGH); // turn on led3
delay(1000);
Serial.println("Enter into gsm") ;
gsm1();
flag1=0;
flag2=0;
digitalWrite(gsml3,LOW);
break;
case 0x11:
digitalWrite(gsml4, HIGH); // turn on led4
delay(1000);
digitalWrite(gsml4,LOW);
break;
case 0x15:
digitalWrite(gpsl1, HIGH); // turn son led5
delay(1000);
digitalWrite(gpsl1,LOW);
break;
default:
digitalWrite(gsml1, HIGH); // turn son led5
delay(1000);
digitalWrite(gsml1,LOW);
flag2++;
break;
}
}
}
if(sw==HIGH)
{
sw= digitalRead(swi);
sense();
w[0]=digitalRead(d8);
w[1]=digitalRead(d9);
w[2]=digitalRead(d10);
w[3]=digitalRead(d11);
r= w[0]+w[1]*2+w[2]*4+w[3]*8;
```

```
switch(r)
{
case B00001111:
analogWrite(l1,0);
analogWrite(l2,0);
analogWrite(r1,0);
analogWrite(r2,0);
break;
case B00001101:
analogWrite(l1,0);
analogWrite(l2,200);
analogWrite(r1,200);
analogWrite(r2,0);
timer_init();
break;
case B00000111:
analogWrite(l1,0);
analogWrite(l2,200);
analogWrite(r1,200);
analogWrite(r2,0);
break;
case B00000101:
for(int l=50;l < 256;l++)
{
for(int m=0;m¡1000;m++)
{
sense();
analogWrite(l1, l);
analogWrite(l2, 0);
analogWrite(r1, l);
analogWrite(r2, 0);
f[0]=digitalRead(d8);
f[1]=digitalRead(d9);
f[2]=digitalRead(d10);
f[3]=digitalRead(d11);
u= f[0]+f[1]*2+f[2]*4+f[3]*8;
if(r!=u)
{
break;
}
}
}
break;
case B00001010:
//Serial.println(”reverse”);
for(int l=50;l¡256;l++)
{
for(int m=0;m¡1000;m++)
{
analogWrite(l1, 0);
analogWrite(l2, l);
analogWrite(r1, 0);
analogWrite(r2, l);
sense();
f[0]=digitalRead(d8);
f[1]=digitalRead(d9);
f[2]=digitalRead(d10);
f[3]=digitalRead(d11);
u= f[0]+f[1]*2+f[2]*4+f[3]*8;
if(r!=u)
{
break;
}
}
}
break;
}
}
}
```