VOICE BASED MOTOR CONTROL

DHRUV GARG	071006
ABHINAV RUSTAGI	071142

Name of supervisor Dr VIVEK SEHGAL

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

May-2011

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

B.Tech

DEPARTMENT OF Electronics and Communication Engineering

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

TABLE OF CONTENTS

Certificate

This is to certify that the project report entitled "**Voice Based Motor Control**", submitted by **Dhruv Garg** and **Abhinav Rustagi** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision. It is certified that this work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	[illegible]
Name of Supervisor	Dr. Vivek Sehgal
Designation	Assistant professor
Date	

ACKNOWLEDGMENT

We would like to express our gratitude to all those who gave us support to complete our project. We express gratitude to our project guide Dr Vivek Sehgal, Department of Electronics and Communication, JUIT, for his help, support, interest, and valuable advice during the project designing and building stage. His enthusiasm and his views for providing only 'high quality work and not less', has made a deep impression on all of us.

Apart from these, countless events, countless people & several incidents have made a contribution to this project that is indescribable.

[illegible]
(Dhruv Garg)

Abhinav
(Abhinav Rustagi)

SUMMARY

The first stage of our project was to have a +12v and +5v dc power supply which was done through Full Wave Rectifier and LM 7805 (voltage regulator) which converts 220 v to +12v and LM7805 further converts it into +5v. After that most important thing was to feed our voice in the computer this was done by using microphone which is then processed by the microprocessor using software visual basic. Then the signal is sent to the operation block via serial communication which converts RS 232 signal to TTL signal and vice versa where microcontroller AT89C2051 is used in order to control the dc motor. If we speak "one", "two" or "three" the motor attached to the microcontroller pin starts rotating at different speeds and if we want to stop the D.C motor we just have to speak "stop" and the motor will stop rotating.

LIST OF FIGURES

Chapter 1

Introduction

The field of computer science that deals with designing computer systems that can recognize spoken words. Note that voice recognition implies only that the computer can take dictation, not that it understands what is being said. Comprehending human languages falls under a different field of computer science called natural language processing.

A number of voice recognition systems are available on the market. The most powerful can recognize thousands of words. However, they generally require an extended training session during which the computer system becomes accustomed to a particular voice and accent. Such systems are said to be speaker dependent.

Many systems also require that the speaker speak slowly and distinctly and separate each word with a short pause. These systems are called discrete speech systems. Recently, great strides have been made in continuous speech systems -- voice recognition systems that allow you to speak naturally. There are now several continuous-speech systems available for personal computers. Instead of typing commands, the user can simply speak into a headset.

In our project we are making speaker independent system which is used to drive a D.C motor, also this system can be used in various application like robotics, industrial appliances, etc.

Chapter 2

Applications

2.1 Health care

In the health care domain, even in the wake of improving speech recognition technologies, medical transcriptionists (MTs) have not yet become obsolete. The services provided may be redistributed rather than replaced.

Voice based system can be implemented in front-end or back-end of the medical documentation process.

Front-End SR is where the provider dictates into a voice based engine, the recognized words are displayed right after they are spoken, and the dictator is responsible for editing and signing off on the document. It never goes through an MT/editor.

Back-End SR or Deferred SR is where the provider dictates into a digital dictation system, and the voice is routed through a voice based machine and the recognized draft document is routed along with the original voice file to the MT/editor, who edits the draft and finalizes the report. Deferred SR is being widely used in the industry currently.

Many Electronic Medical Records (EMR) applications can be more effective and may be performed more easily when deployed in conjunction with a voice based engine. Searches, queries, and form filling may all be faster to perform by voice than by using a keyboard.

2.2Military

High-performance fighter aircraft:-

Substantial efforts have been devoted in the last decade to the test and evaluation of voice based in fighter aircraft. Of particular note is the U.S. program in speech recognition for the Advanced Fighter Technology Integration (AFTI)/F-16 aircraft (F-16 VISTA), and a program in France installing voice based systems on Mirage aircraft, and also programs in the UK dealing with a variety of aircraft platforms. In these programs, voice based system have been operated successfully in fighter aircraft, with applications including: setting radio frequencies, commanding an autopilot system, setting steer-point coordinates and weapons release parameters, and controlling flight displays.

Working with Swedish pilots flying in the JAS-39 Gripen cockpit, England (2004) found recognition deteriorated with increasing G-loads. It was also concluded that adaptation greatly improved the results in all cases and introducing models for breathing was shown to improve recognition scores significantly. Contrary to what might be expected, no effects of the broken English of the speakers were found. It was evident that spontaneous voice caused problems for the recognizer, as could be expected. A restricted vocabulary, and above all, a proper syntax, could thus be expected to improve recognition accuracy substantially.

Speaker independent systems are also being developed and are in testing for the F35 Lightning II (JSF) and the Alenia Aermacchi M-346 Master lead-in fighter trainer. These systems have produced word accuracies in excess of 98%.

2.3Helicopters:-

The problems of achieving high recognition accuracy under stress and noise pertain strongly to the helicopter environment as well as to the jet fighter environment. The acoustic noise problem is actually more severe in the helicopter environment, not only because of the high noise levels but also because the helicopter pilot generally does not wear a facemask, which would reduce acoustic noise in the microphone. Substantial test and evaluation programs have been carried out in the past decade in voice based systems applications in helicopters, notably by the U.S. Army Avionics Research and Development Activity (AVRADA) and by the Royal Aerospace Establishment (RAE) in the UK. Work in France has included voice based system in the Puma helicopter. There has also been much useful work in Canada. Results have been encouraging, and voice applications have included: control of communication radios; setting of navigation systems; and control of an automated target handover system.

As in fighter applications, the overriding issue for voice in helicopters is the impact on pilot effectiveness. Encouraging results are reported for the AVRADA tests, although these represent only a feasibility demonstration in a test environment.

2.4Training air traffic controllers

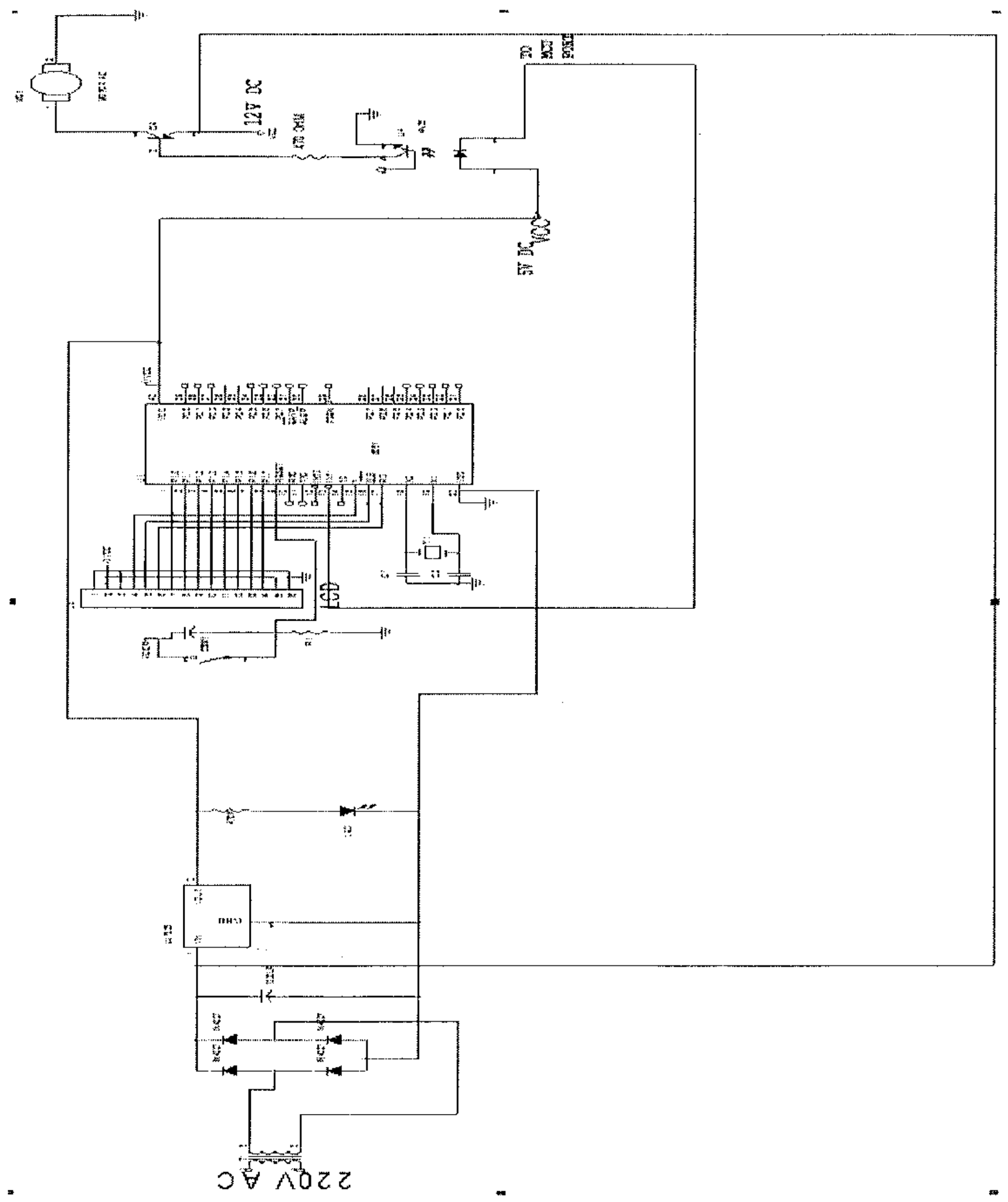
Training for air traffic controllers (ATC) represents an excellent application for voice based systems. Many ATC training systems currently require a person to act as a "pseudo-pilot", engaging in a voice dialog with the trainee controller, which simulates the dialog which the controller would have to conduct with pilots in a real ATC situation. Voice based and synthesis techniques offer the potential to eliminate the need for a person to act as pseudo-pilot, thus reducing training and support personnel. In theory, Air controller tasks are also characterized by highly structured speech as the primary output of the controller.

The USAF, USMC, US Army, US Navy and FAA as well as a number of international ATC training organizations such as the Royal Australian Air Force and Civil Aviation Authorities in Italy, Brazil, and Canada are currently using ATC simulators with voice based system from a number of different vendors.

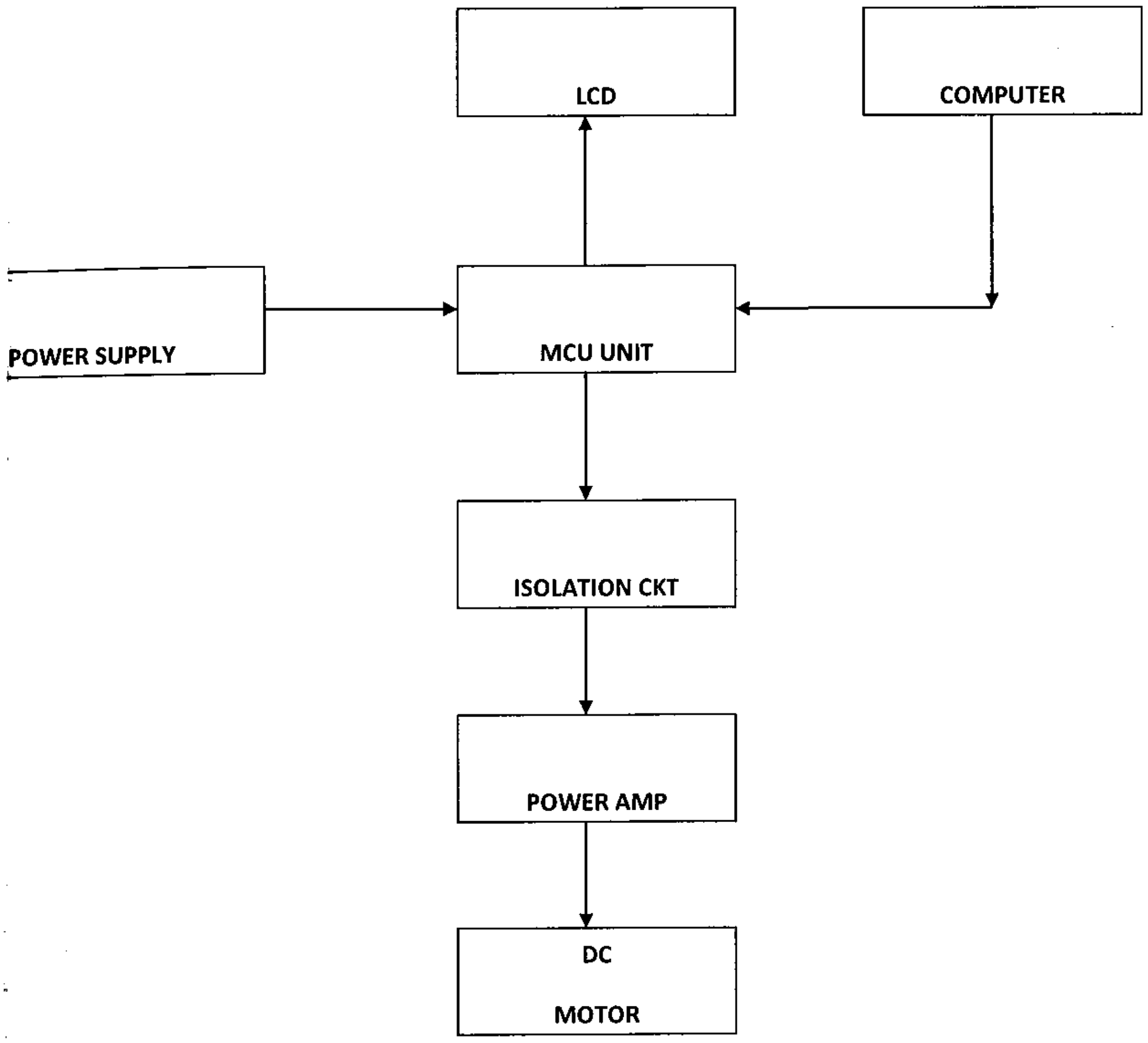
Chapter 3

Circuit Design

3.1 Circuit Design



3.2 Block Diagram



Chapter 4

Hardware Approach

4.1 Components

4.1.1 Transformer

A transformer is a static device that transfers electrical energy from one circuit to another through inductively coupled conductors—the transformer's coils. A varying current in the first or primary winding creates a varying magnetic flux in the transformer's core and thus a varying magnetic field through the secondary winding. This varying magnetic field induces a varying electromotive force (EMF) or "voltage" in the secondary winding. This effect is called mutual induction.

If a load is connected to the secondary, an electric current will flow in the secondary winding and electrical energy will be transferred from the primary circuit through the transformer to the load. In an ideal transformer, the induced voltage in the secondary winding (V_s) is in proportion to the primary voltage (V_p), and is given by the ratio of the number of turns in the secondary (N_s) to the number of turns in the primary (N_p) as follows:

$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

By appropriate selection of the ratio of turns, a transformer thus allows an alternating current (AC) voltage to be "stepped up" by making N_s greater than N_p, or "stepped down" by making N_s less than N_p.

Induction law

The voltage induced across the secondary coil may be calculated from Faraday's law of induction, which states that:

$$V_s = N_s \frac{d\Phi}{dt},$$

where V_s is the instantaneous voltage, N_s is the number of turns in the secondary coil and Φ is the magnetic flux through one turn of the coil.

Ideal power equation

If the secondary coil is attached to a load that allows current to flow, electrical power is transmitted from the primary circuit to the secondary circuit. Ideally, the transformer is perfectly efficient; all the incoming energy is transformed from the primary circuit to the magnetic field and into the secondary circuit. If this condition is met, the incoming electric power must equal the outgoing power:

$$P_{\text{incoming}} = I_p V_p = P_{\text{outgoing}} = I_s V_s,$$

given the ideal transformer equation

$$\frac{V_s}{V_p} = \frac{N_s}{N_p} = \frac{I_p}{I_s}.$$

Detailed Operation

When a voltage is applied to the primary winding, a small current flows, driving flux around the magnetic circuit of the core. The current required to create the flux is termed the magnetizing current; since the ideal core has been assumed to have near-zero reluctance, the magnetizing current is negligible, although still required to create the magnetic field.

The changing magnetic field induces an electromotive force (EMF) across each winding.[32] Since the ideal windings have no impedance, they have no associated voltage drop, and so the voltages V_P and V_S measured at the terminals of the transformer, are equal to the corresponding EMFs. The primary EMF, acting as it does in opposition to the primary voltage, is sometimes termed the "back EMF". This is due to Lenz's law which states that the induction of EMF would always be such that it will oppose development of any such change in magnetic field.

Leakage Flux

The ideal transformer model assumes that all flux generated by the primary winding links all the turns of every winding, including itself. In practice, some flux traverses paths that take it outside the windings. Such flux is termed leakage flux, and results in leakage inductance in series with the mutually coupled transformer windings. Leakage results in energy being alternately stored in and discharged from the magnetic fields with each cycle of the power supply. It is not directly a power loss (see "Stray losses" below), but results in inferior voltage regulation, causing the secondary voltage to fail to be directly proportional to the primary, particularly under heavy load. Transformers are therefore normally designed to have very low leakage inductance.

Cores

Laminated steel cores

Transformers for use at power or audio frequencies typically have cores made of high permeability silicon steel. The steel has a permeability many times that of free space, and the core thus serves to greatly reduce the magnetizing current, and confine the flux to a path which closely couples the windings. Later designs constructed the core by stacking layers of thin steel laminations, a principle that has remained in use. Each lamination is insulated from its neighbors by a thin non-conducting layer of insulation. The effect of laminations is to confine eddy currents to highly elliptical paths that enclose little flux, and so reduce their magnitude. Thinner laminations reduce losses,[54] but are more laborious and expensive to construct.

Thin laminations are generally used on high frequency transformers, with some types of very thin steel laminations able to operate up to 10 kHz.

Solid cores

Powdered iron cores are used in circuits (such as switch-mode power supplies) that operate above main frequencies and up to a few tens of kilohertz. These materials combine high magnetic permeability with high bulk electrical resistivity. For frequencies extending beyond the VHF band, cores made from non-conductive magnetic ceramic materials called ferrites are common. Some radio-frequency transformers also have movable cores (sometimes called 'slugs') which allow adjustment of the coupling coefficient (and bandwidth) of tuned radio-frequency circuits.

Windings

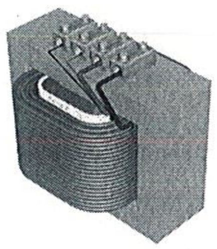


Figure 1: Windings in a transformer

The conducting material used for the windings depends upon the application, but in all cases the individual turns must be electrically insulated from each other to ensure that the current travels throughout every turn. For small power and signal transformers, in which currents are low and the potential difference between adjacent turns is small, the coils are often wound from enamelled magnet wire. Larger power transformers operating at high voltages may be wound with copper rectangular strip conductors insulated by oil-impregnated paper and blocks of pressboard.

For signal transformers, the windings may be arranged in a way to minimize leakage inductance and stray capacitance to improve high-frequency response. This can be done by splitting up each coil into sections, and those sections placed in layers between the sections of the other winding. This is known as a stacked type or interleaved winding. Both the primary and secondary windings on power transformers may have external connections, called taps, to intermediate points on the winding to allow selection of the voltage ratio.

4.1.2 RECTIFIER

Rectifier

A rectifier is an electrical device that converts alternating current (AC), which periodically reverses direction, to direct current (DC), which is in only one direction, a process known as rectification.

A device which performs the opposite function (converting DC to AC) is known as an inverter. There are two types of rectifiers-:

1) Half wave rectifier

2) Full wave rectifier

Half-wave rectification

In half wave rectification, either the positive or negative half of the AC wave is passed, while the other half is blocked. Because only one half of the input waveform reaches the output, it is very inefficient if used for power transfer. Half-wave rectification can be achieved with a single diode in a one-phase supply, or with three diodes in a three-phase supply.

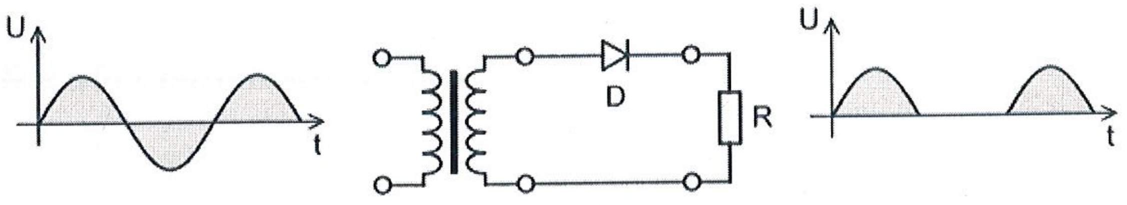


Figure 2: Half wave rectifier

Disadvantage of Half-Wave-Rectifier:

1. Half wave rectification involves a lot of wastage of energy and hence it is not preferred.
2. A small current flows during reverse bias due to minority charge carriers. As the output across (RL) is negligible.

3. The resulting D.C. voltage is not steady enough for some purpose. The following device is used when a very steady D.C. voltage is required.

Full-wave rectification

A full-wave rectifier converts the whole of the input waveform to one of constant polarity (positive or negative) at its output. Full-wave rectification converts both polarities of the input waveform to DC (direct current), and is more efficient. However, in a circuit with a non-center tapped transformer, four diodes are required instead of the one needed for half-wave rectification. Four diodes arranged this way are called a diode bridge or bridge rectifier.

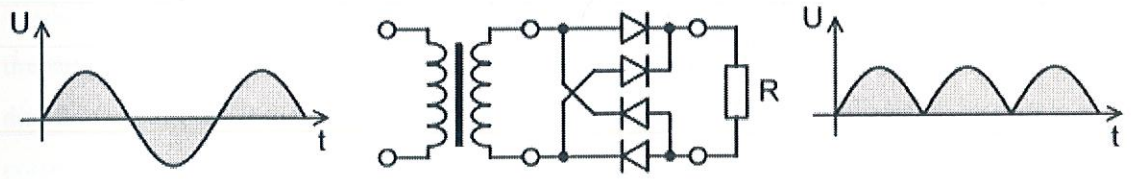


Figure 3: Full wave rectifier

Graetz bridge rectifier: a full-wave rectifier using 4 diodes.

Rectifier output smoothing

While half-wave and full-wave rectification suffice to deliver a form of DC output, neither produces constant-voltage DC. In order to produce steady DC from a rectified AC supply, a smoothing circuit or filter is required. In its simplest form this can be just a reservoir capacitor or smoothing capacitor, placed at the DC output of the rectifier. There will still remain an amount of AC ripple voltage where the voltage is not completely smoothed.

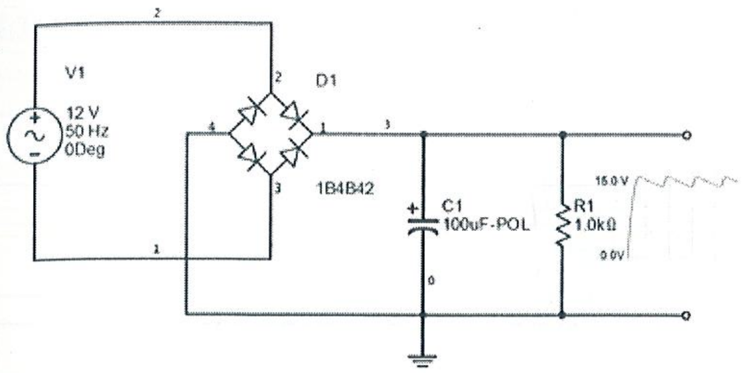


Figure 4: Full wave rectifier output

Sizing of the capacitor represents a tradeoff. For a given load, a larger capacitor will reduce ripple but will cost more and will create higher peak currents in the transformer secondary and in the supply feeding it. In extreme cases where many rectifiers are loaded onto a power distribution circuit, it may prove difficult for the power distribution authority to maintain a correctly shaped sinusoidal voltage curve.

For a given tolerable ripple the required capacitor size is proportional to the load current and inversely proportional to the supply frequency and the number of output peaks of the rectifier per input cycle. The load current and the supply frequency are generally outside the control of the designer.

A half-wave rectifier will only give one peak per cycle and for this and other reasons is only used in very small power supplies. A full wave rectifier achieves two peaks per cycle and this is the

Best that can be done with single-phase input. For three-phase inputs a three-phase bridge will give six peaks per cycle.

To further reduce this ripple, a capacitor-input filter can be used. This complements the reservoir capacitor with a choke (inductor) and a second filter capacitor, so that a steadier DC output can be obtained across the terminals of the filter capacitor. The choke presents high impedance to the ripple current.

4.1.3 LM 7805(VOLTAGE REGULATOR)

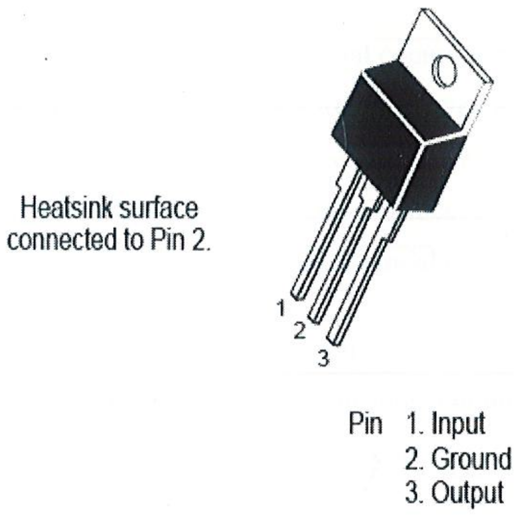


Figure 5: LM 7805

These voltage regulators are monolithic integrated circuits designed as fixed–voltage regulators for a wide variety of applications including local, on–card regulation. These regulators employ internal current limiting, thermal shutdown, and safe–area compensation. With adequate heat sinking they can deliver output currents in excess of 1.0 A. Although designed primarily as a fixed voltage regulator, these devices can be used with external components to obtain adjustable voltages and currents.

The 78xx (sometimes LM78xx) is a family of self-contained fixed linear voltage regulator integrated circuits. The 78xx family is commonly used in electronic circuits requiring a regulated power supply due to their ease-of-use and low cost. For ICs within the family, the xx is replaced with two digits, indicating the output voltage (for example, the 7805 has a 5 volt output, while the 7812 produces 12 volts). These devices support an input voltage anywhere from a couple of volts over the intended output voltage, up to a maximum of 35 or 40 volts, and typically provide 1 or 1.5 amps of current.

Advantages

- 78xx series ICs do not require additional components to provide a constant, regulated source of power, making them easy to use, as well as economical and efficient uses of space. Other voltage regulators may require additional components to set the output voltage level, or to assist in the regulation process. Some other designs (such as a switching power supply) may need substantial engineering expertise to implement.
- 78xx series ICs have built-in protection against a circuit drawing too much power. They have protection against overheating and short-circuits, making them quite robust in most applications. In some cases, the current-limiting features of the 78xx devices can provide protection not only for the 78xx itself, but also for other parts of the circuit.

Some Features

□ □ Output Current in Excess of 1.0 A
□ □ No External Components Required
□ □ Internal Thermal Overload Protection
□ □ Internal Short Circuit Current Limiting
□ □ Output Transistor SafeArea Compensation
□□ Output Voltageoffered in 2% and 4% Tolerance

4.1.4 LCD (LIQUID CRUSTAL DISPLAY)

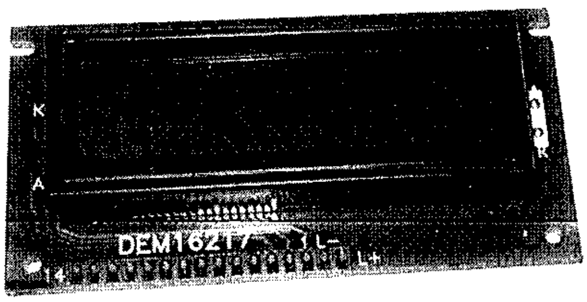


Figure 6: Liquid Crystal Display

A liquid crustal cell consist of a thin layer (about 10 micro meter) of a liquid crystal sandwiched between two glass sheets with transparent electrodes on their inside faces. With both glass sheets transparent, the cell is known as transmittive type cell. When one glass is transparent and other has a reflective coating, the cell is called reflective type. The LCD does not produce any illumination of its own. It, in fact depends entirely on illumination falling on it from external source for its visual effects.

LCD Display

Liquid crystal displays (LCD) are widely used in recent years as compares to LEDs. This is due to the declining prices of LCD, the ability to display numbers, characters and graphics, incorporation of a refreshing controller into the LCD, their by relieving the CPU of the task of refreshing the LCD and also the ease of programming for characters and graphics.

LCD pin description

The LCD discuss in this section has the most common connector used for the Hitachi 44780 based LCD is 14 pins in a row and modes of operation.

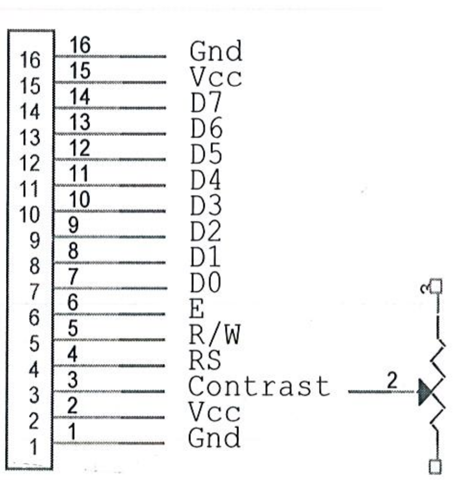


Figure7: LCD Pin Description Diagram

V_{CC}, V_{SS}, V_{EE}

The voltage V_{CC} and V_{SS} provided by +5V and ground respectively while V_{EE} is used for controlling LCD contrast. Variable voltage between Ground and Vcc is used to specify the contrast (or "darkness") of the characters on the LCD screen.

RS (register select)

There are two important registers inside the LCD. The RS pin is used for their selection as follows. If RS=0, the instruction command code register is selected, then allowing to user to send a command such as clear display, cursor at home etc.. If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W (read/write)

The R/W (read/write) input allowing the user to write information from it. R/W=1, when it read and R/W=0, when it writing.

EN (enable)

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high power, a high-to-low pulse must be applied to this pin in order to for the LCD to latch in the data presented at the data pins.

D0-D7 (data lines)

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD's internal registers. To displays the letters and numbers, we send ASCII codes for the letters A-Z, a-z, and numbers 0-9 to these pins while making RS =1. There are also command codes that can be sent to clear the display or force the cursor to the home position or blink the cursor.

We also use RS =0 to check the busy flag bit to see if the LCD is ready to receive the information. The busy flag is D7 and can be read when R/W =1 and RS =0, as follows: if R/W =1 and RS =0, when D7 =1(busy flag =1), the LCD is busy taking care of internal operations and will not accept any information. When D7 =0, the LCD is ready to receive new information.

Pin	Symbol	I/O	Description
1	VSS	-	Ground
2	VCC	-	+5V power supply
3	VEE	-	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register.
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	PB0	I/O	The 8 bit data bus
8	PB1	I/O	The 8 bit data bus
9	DB2	I/O	The 8 bit data bus
10	DB3	I/O	The 8 bit data bus
11	DB4	I/O	The 8 bit data bus
12	DB5	I/O	The 8 bit data bus
13	DB6	I/O	The 8 bit data bus
14	DB7	I/O	The 8 bit data bus

Figure 8: LCD pins discription

LCD Background

Frequently, an 8051 program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an 8051 is an LCD display. Some of the most common LCDs connected to the 8051 are 16x2 and 20x2 displays. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip which receives data from an external source (in this case, the 8051) and communicates directly with the LCD.

44780 BACKGROUND

The 44780 standard requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).

The three control lines are referred to as **EN**, **RS**, and **RW**.

The **EN** line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring **EN** high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

The **RS** line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

The **RW** line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

4.1.5 AT89C52 (8-bit microcontroller)

Figure 9: Microcontroller

Introduction

A microcontroller is a computer with most of the necessary support chips onboard. All computers have several things in common, namely:

- A central processing unit (CPU) that 'executes' programs.
- Some random-access memory (RAM) where it can store data that is variable.
- Some read only memory (ROM) where programs to be executed can be stored.
- Input and output (I/O) devices that enable communication to be established with the outside world i.e. connection to devices such as keyboard, mouse, monitors and other peripherals.

There are a number of other common characteristics that define microcontrollers. If a computer matches a majority of these characteristics, then it can be classified as a 'microcontroller'. Microcontrollers may be:

- 'Embedded' inside some other device (often a consumer product) so that they can control the features or actions of the product. Another name for a microcontroller is therefore an 'embedded controller'.
- Dedicated to one task and run one specific program. The program is stored in ROM and generally does not change.
- A low-power device. A battery-operated microcontroller might consume as little as 50 mill watts.

Description

The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 instruction set and pin out.

The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications

PIN DIAGRAM

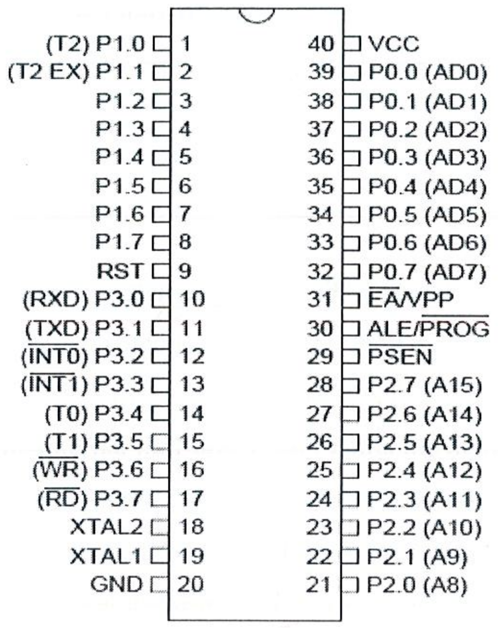


Figure 10: Pin diagram of 8051

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs. Port 0 can also be configured to be the multiplexed low order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pull-ups. In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses. In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current because of the pull-ups.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

ALE/PROG

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming.

PSEN

Program Store Enable is the read strobe to external program memory. When the AT89C52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

4.1.6 Opto-coupler (4N35)

Photodiode

A photodiode is a type of photo detector capable of converting light into either current or voltage, depending upon the mode of operation.

Photodiode Operation

A photodiode is a PN junction or PIN structure. When a photon of sufficient energy strikes the diode, it excites an electron, thereby creating a free electron (and a positively charged electron hole). This mechanism is also known as the photoelectric effect. If the absorption occurs in the junction's depletion region, or one diffusion length away from it, these carriers are swept from the junction by the built-in field of the depletion region. Thus holes move toward the anode, and electrons toward the cathode, and a photocurrent is produced. This photocurrent is the sum of both the dark current (without light) and the light current, so the dark current must be minimized to enhance the sensitivity of the device.

Photovoltaic mode

When used in zero bias or photovoltaic mode, the flow of photocurrent out of the device is restricted and a voltage builds up. This mode exploits the photovoltaic effect, which is the basis for solar cells – in fact; a traditional solar cell is just a large area photodiode.

Photoconductive mode

In this mode the diode is often reverse biased, dramatically reducing the response time at the expense of increased noise. This increases the width of the depletion layer, which decreases the junction's capacitance resulting in faster response times. The reverse bias induces only a small amount of current (known as saturation or back current) along its direction while the photocurrent remains virtually the same. For a given spectral distribution, the photocurrent is linearly proportional to the illuminance.

Although this mode is faster, the photoconductive mode tends to exhibit more electronic noise.

Opto-isolators

In electronics, an opto-isolator, also called an opt coupler, photo coupler, or optical isolator, is "an electronic device designed to transfer electrical signals by utilizing light waves to provide coupling with electrical isolation between its input and output". The main purpose of an opto-isolator is "to prevent high voltages or rapidly changing voltages on one side of the circuit from damaging components or distorting transmissions on the other side.

An opto-isolator contains a source (emitter) of light, almost always a near infrared light-emitting diode (LED), that converts electrical input signal into light, a closed optical channel (also called dielectrical channel), and a photo sensor, which detects incoming light and either generates electric energy directly, or modulates electric current flowing from an external power supply. The sensor can be a photo resistor, a photodiode, a phototransistor. Because LEDs can sense light in addition to emitting it, construction of symmetrical, bidirectional opto-isolators is possible.

Photodiode Optoisolators

Diode opto-isolators employ LEDs as sources of light and silicon photodiodes as sensors. When the photodiode is reverse-biased with an external voltage source, incoming light increases the reverse current flowing through the diode. The diode itself does not generate energy; it modulates the flow of energy from an external source. This mode of operation is called photoconductive mode. Alternatively, in the absence of external bias the diode converts the energy of light into electric energy by charging its terminals to a voltage of up to 0.7 V. The rate of charge is proportional to the intensity of incoming light. This mode of operation is called photovoltaic mode.

Photodiode opto-isolators can be used for interfacing analog signals, although their non-linearity invariably distorts the signal. A special class of analog opto-isolators introduced by Burr-Brown uses two photodiodes and an input-side operational amplifier to compensate for diode non-linearity. One of two identical diodes is wired into the feedback loop of the amplifier, which maintains overall current transfer ratio at a constant level regardless of the non-linearity in the second (output) diode.

Opto-coupler :- It has one IR LED and a photo- transistor. One pin of the LED is connected to the MCU to get a signal (0 or 1) and the pin is given ground. When the signal from the MCU is 0, then LED emits light. This light will turn on the NPN transistor. Collector of the transistor is grounded. Emitter is connected to the PNP transistor whose collector is connected to Vcc and emitter to the relay.

P in NPN transistor means 1 and n means 0.

RELAY

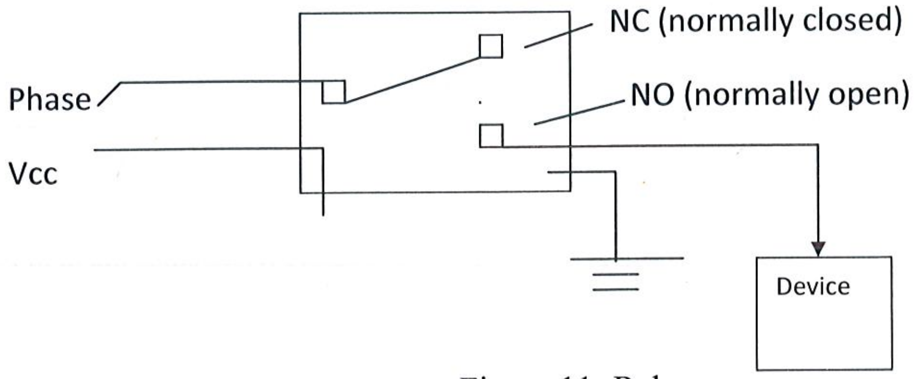


Figure 11: Relay

One point of the relay is given to phase and that point is connected to NC point. VCC is connected to the coil the relay. When vcc is given the coil gets magnetized and slowly phase is connected to NO point which is further connected to the device to be operated by the relay. Neutral must be given to the device.

4.1.7 MAX 202

The **MAX232** is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case.

The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.

The later MAX232A is backwards compatible with the original MAX232 but may operate at higher baud rates and can use smaller external capacitors – 0.1 μF in place of the 1.0 μF capacitors used with the original device.

The newer MAX3232 is also backwards compatible, but operates at a broader voltage range, from 3 to 5.5 V.

Pin to pin compatible: ICL232, ST232, ADM232, and HIN232

Voltage levels

It is helpful to understand what occurs to the voltage levels. When a MAX232 IC receives a TTL level to convert, it changes a TTL Logic 0 to between +3 and +15 V, and changes TTL Logic 1 to between -3 to -15 V, and vice versa for converting from RS232 to TTL. This can be confusing when you realize that the RS232 Data Transmission voltages at a certain logic state are opposite from the RS232 Control Line voltages at the same logic state. To clarify the matter, see the table below. For more information see RS-232 Voltage Levels

Pin description:

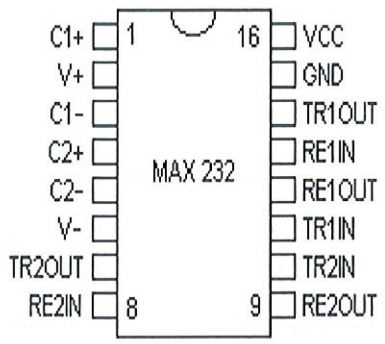


Figure 12(a): Pin diagram of MAX 232

UART (Universal Asynchronous Receiver Transmitter) or USART (Universal Synchronous Asynchronous Receiver Transmitter) are one of the basic interface which you will find in almost all the controllers available in the market till date. This interface provides a cost effective simple and reliable communication between one controller to another controller or between a controller and PC.

RS-232 (Recommended Standard 232) is a standard for serial binary data signals connecting between a DTE (Data terminal equipment) and a DCE (Data Circuit-terminating Equipment).

Voltage Levels:

The RS-232 standard defines the voltage levels that correspond to logical one and logical zero levels. Valid signals are plus or minus 3 to 25 volts. The range near zero volts is not a valid RS-232 level; logic one is defined as a negative voltage, the signal condition is called marking, and has the functional significance of OFF. Logic zero is positive, the signal condition is spacing, and has the function ON.

So a Logic Zero represented as +3V to +25V and Logic One represented as -3V to -25V.

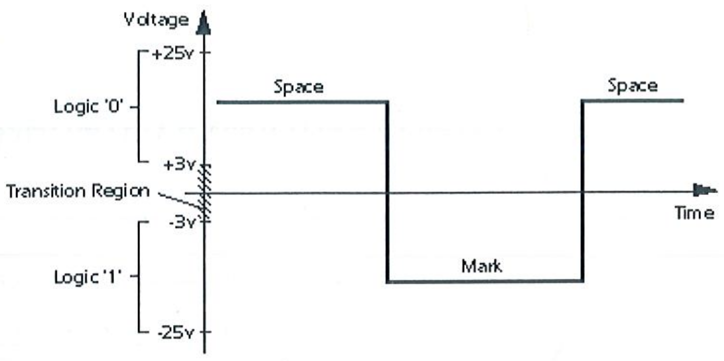


Figure 12(b): Waveform for logic transition

4.1.8 MOTOR

Speed Control of D.C. Series Motors:

Speed control of D.C. series motor may be accomplished through either armature control, field control.

1. ARMATURE CONTROL METHODS: Speed adjustment of D.C. series motors by armature control may be had by any one of the following methods:

(a)ARMATURE RESISTANCE CONTROL:

This is the most common method employed. It is obtained in the same way as D.C. shunt motor with the exception that the control resistance may be connected directly in series with the supply to the complete motor. The drawbacks of armature resistance method for machines with shunt fields are not as important in the speed control of D.C. series motor. This method of speed control is most economical for constant torque drives and is employed chiefly for D.C. series motors driving cranes, hoists, trains etc. because such devices operate on intermittent duty. The power load in the control resistance for many applications of the series D.C motors is not too serious, since in these applications the control is utilized for a larger portion of time for reducing the speed under light load conditions and is used intermittently when the motor is carrying full-load. The maximum range of speed control of about 3:1 will be available depending on the load.

(b)SHUNTED ARMATURE CONTROL:

The combination of a rheostat shunting the armature and a rheostat in series with the armature is used to give slow speeds at light loads. Such a scheme accomplishes the speed control both by lowering the voltage applied to the motor armature and by varying the flux. The voltage applied to the armature terminals is varied by varying series rheostat R1. The exciting current can be varied by varying the armature shunting resistance R2 for same armature current Ia. For a given constant load torque , if armature Ia is reduced due to armature then the flux must increase because torque developed by the armature T is proportional to product of flux and armature current. This causes increase in current drawn from the supply mains, so increase in flux and decrease in speed. In this method speed control can be obtained over a wide range but below normal speed. The limit of speed control range depends upon ratio existing between resistances R1and R2 and also on the current at which there is saturation of the magnetic circuit. No speed can effectively be adjusted to any desired low speed .This method is not economical due to considerable power losses in the speed controlling resistances.

(c)ARMATURE TERMINAL VOLTAGE CONTROL:

The speed control of D.C series motor can be accomplished by supplying the power to the motor from a separate variable voltage supply. This method is seldom used because of high cost of control equipment.

2. FIELD CONTROL METHODS:

The speed of D.C series motor can be controlled by varying the flux in any of the following manners:

(a)FIELD DIVERTOR METHOD:

The field flux can be reduced by shunting a portion of motor current around the series field thus reducing the excitation mmf and weakening of field. This method gives speed above normal because flux is reduced by this method. Lesser the divertor resistance, less the field current, less speed and hence more speed. This method is convenient as well as economical and provides the speed control range usually not exceeding 2:1.This method is used in drives in which the speed should rise sharply as soon as the load falls.

(b) TAPPED FIELD CONTROL:

This is a method of increasing the speed by reducing flux and is accomplished by reducing the number of turns of the field winding through which the current flows. In this method a number of tapping form the field winding are brought outside. A number of series field turns can be short-circuited according to the requirement. When all field turns are in circuit, the motor runs at its lowest speed and speed increases with cutting out some of the series field turns. This method is usually employed in electric traction.

4.1.9 Resistor

A resistor is a two-terminal passive electronic component which implements electrical resistance as a circuit element. When a voltage V is applied across the terminals of a resistor, a current I will flow through the resistor in direct proportion to that voltage. Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment.

Figure 13: Resistor

CAPACITOR

A capacitor (formerly known as condenser) is a device for storing electric charge. The forms of practical capacitors vary widely, but all contain at least two conductors separated by a non-conductor. Capacitors used as parts of electrical systems, for example, consist of metal foils separated by a layer of insulating film.

Figure 14: Capacitor

A capacitor is a passive electronic component consisting of a pair of conductors separated by a dielectric (insulator). When there is a potential difference (voltage) across the conductors, a static electric field develops across the dielectric, causing positive charge to collect on one plate and negative charge on the other plate. Energy is stored in the electrostatic field. An ideal capacitor is characterized by a single constant value, capacitance, measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them.

Capacitors are widely used in electronic circuits for blocking direct current while allowing alternating current to pass, in filter networks, for smoothing the output of power supplies, in the resonant circuits that tune radios to particular frequencies and for many other purposes.

4.1.10 LED

A light-emitting diode is a semiconductor light source. LEDs are used as indicator lamps in many devices and are increasingly used for other lighting. Introduced as a practical electronic component in 1962,[2] early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness.

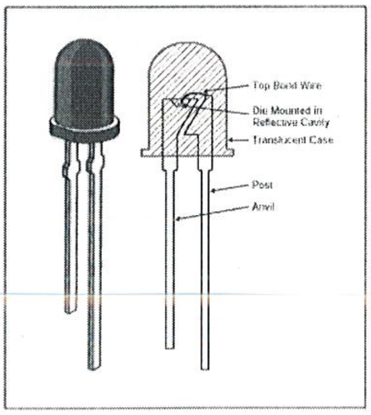


Figure 15: LED

When a light-emitting diode is forward biased (switched on), electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is often small in area (less than 1 mm^2), and integrated optical components may be used to shape its radiation pattern.[3] LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability. LEDs powerful enough for room lighting are relatively expensive and require more precise current and heat management than compact fluorescent lamp sources of comparable output.

4.1.11 DIODE

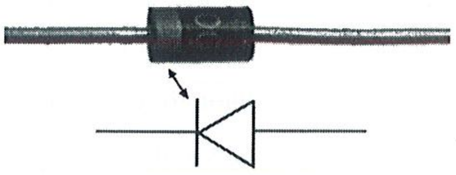


Figure 16: Diode

In electronics, a diode is a two-terminal electronic component that conducts electric current in only one direction. The term usually refers to a semiconductor diode, the most common type today. This is a crystalline piece of semiconductor material connected to two electrical terminals.[1] A vacuum tube diode (now little used except in some high-power technologies) is a vacuum tube with two electrodes: a plate and a cathode.

The most common function of a diode is to allow an electric current to pass in one direction (called the diode's forward direction), while blocking current in the opposite direction (the reverse direction). Thus, the diode can be thought of as an electronic version of a check valve. This unidirectional behavior is called rectification, and is used to convert alternating current to direct current,

Chapter 5

Visual Studio

5.1 Introduction

History

After the release of Internet Information Services 4.0 in 1997, Microsoft began researching possibilities for a new web application model that would solve common complaints about ASP, especially with regard to separation of presentation and content and being able to write "clean" code. Mark Anders, a manager on the IIS team, and Scott Guthrie, who had joined Microsoft in 1997 after graduating from Duke University, were tasked with determining what that model would look like. The initial design was developed over the course of two months by Anders and Guthrie, and Guthrie coded the initial prototypes during the Christmas holidays in 1997.

Scott Guthrie (Microsoft Developer Division VP) in 2007

The initial prototype was called "XSP"; Guthrie explained in a 2007 interview that, "People would always ask what the X stood for. At the time it really didn't stand for anything. XML started with that; XSLT started with that. Everything cool seemed to start with an X, so that's what we originally named it." The initial prototype of XSP was done using Java, but it was soon decided to build the new platform on top of the Common Language Runtime (CLR), as it offered an object-oriented programming environment, garbage collection and other features that were seen as desirable features that Microsoft's Component Object Model platform didn't support. Guthrie described this decision as a "huge risk", as the success of their new web development platform would be tied to the success of the CLR, which, like XSP, was still in the early stages of development, so much so that the XSP team was the first team at Microsoft to target the CLR.

With the move to the Common Language Runtime, XSP was re-implemented in C# (known internally as "Project Cool" but kept secret from the public), and the name changed to ASP+, as by this point the new platform was seen as being the successor to Active Server Pages, and the intention was to provide an easy migration path for ASP developers.

Mark Anders first demonstrated ASP+ at the ASP Connections conference in Phoenix, Arizona on May 2, 2000. Demonstrations to the wide public and initial beta release of ASP+ (and the rest of the .NET Framework) came at the 2000 Professional Developers Conference on July 11, 2000 in Orlando, Florida. During Bill Gates' keynote presentation, Fujitsu demonstrated ASP+ being used in conjunction with COBOL, and support for a variety of other languages was announced, including Microsoft's new Visual Basic .NET and C# languages, as well as Python and Perl support by way of interoperability tools created by Active State.

Once the ".NET" branding was decided on in the second half of 2000, it was decided to rename ASP+ to ASP.NET. Mark Anders explained on an appearance on The MSDN Show that year that, "The .NET initiative is really about a number of factors, it's about delivering software as a service, it's about XML and web services and really enhancing the Internet in terms of what it can do ... we really wanted to bring its name more in line with the rest of the platform pieces that make up the .NET framework."

After four years of development, and a series of beta releases in 2000 and 2001, ASP.NET 1.0 was released on January 5, 2002 as part of version 1.0 of the .NET Framework. Even prior to the release, dozens of books had been written about ASP.NET, and Microsoft promoted it heavily as part of their platform for web services. Guthrie became the product unit manager for ASP.NET, and development continued apace, with version 1.1 being released on April 24, 2003 as a part of Windows Server 2003. This release focused on improving ASP.NET's support for mobile devices.

Characteristics

ASP.NET web pages, known officially as "web forms", are the main building block for application development.[8] Web forms are contained in files with an ".aspx" extension; these files typically contain static (X)HTML markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the required static and dynamic content for the web page. Additionally, dynamic code which runs on the server can be placed in a page within a block <% -- dynamic code -- %>, which is similar to other web development technologies such as PHP, JSP, and ASP. With ASP.NET Framework 2.0, Microsoft introduced a new code-behind model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs file (depending on the programming language used).

Code-behind model

Microsoft recommends dealing with dynamic program code by using the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files typically have names like MyPage.aspx.cs or MyPage.aspx.vb while the page files isMyPage.aspx (same filename as the page file (ASPX), but with the final extension denoting the page language). This practice is automatic in Microsoft Visual Studio and other IDEs. When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through of the document.

ASP.NET's code-behind model marks a departure from Classic ASP in that it encourages developers to build applications with separation of presentation and content in mind. In theory, this would allow a web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it.

Examples

Note that this sample uses code "inline", as opposed to code-behind.

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
 protected void Page_Load(object sender, EventArgs e)
  {
   lbl1.Text = DateTime.Now.ToLongTimeString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
 <title>Sample page</title>
</head>
<body>
 <form id="form1" runat="server">
  <div>
   The current time is: <asp:Label runat="server" id="lbl1" />
  </div>
 </form>
</body>
</html>
```

The above page renders with the Text "The current time is: " and the <asp:Label> Text is set with the current time, upon render.

Code-behind solutions

```
<%@ Page Language="C#" CodeFile="SampleCodeBehind.aspx.cs"
Inherits="Website.SampleCodeBehind"
AutoEventWireup="true" %>
```

The above tag is placed at the beginning of the ASPX file. The CodeFile property of the @ Page directive specifies the file (.cs or .vb) acting as the code-behind while the Inherits property specifies the Class the Page derives from. In this example, the @ Page directive is included in SampleCodeBehind.aspx, then SampleCodeBehind.aspx.cs acts as the code-behind for this page:

```
using System;
namespace Website
{
 public partial class SampleCodeBehind : System.Web.UI.Page
 {
  protected void Page_Load(object sender, EventArgs e)
  {
   Response.Write("Hello, world");
```

```
    }
  }
}
```

In this case, the Page_Load() method is called every time the ASPX page is requested. The programmer can implement event handler at several stages of the page execution process to perform processing.

User controls

User controls are encapsulations of sections of pages which are registered and used as controls in ASP.NET. User controls are created as ASCX markup files. These files usually contain static (X)HTML markup, as well as markup defining server-side web controls. These are the locations where the developer can place the required static and dynamic content. A user control is compiled when its containing page is requested and is stored in memory for subsequent requests. User controls have their own events which are handled during the life of ASP.NET requests. An event bubbling mechanism provides the ability to pass an event fired by a user control up to its containing page. Unlike an ASP.NET page, a user control cannot be requested independently; one of its containing pages is requested instead.

Custom controls

Programmers can also build custom controls for ASP.NET applications. Unlike user controls, these controls don't have an ASCX markup file, having all their code compiled into a dynamic link library (DLL) file. Such custom controls can be used across multiple web applications and Visual Studio project

Rendering technique

ASP.NET uses a visited composites rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting HTML output is sent to the client.

After the request has been processed, the instance of the page class is discarded and with it the entire control tree. This is a source of confusion among novice ASP.NET programmers who rely on class instance members that are lost with every page request/response cycle.

State management

ASP.NET applications are hosted by a web server and are accessed using
the stateless HTTP protocol. As such, if an application uses stateful interaction, it has to implement state management on its own. ASP.NET provides various functions for state management. Conceptually, Microsoft treats "state" as GUI state. Problems may arise if an application needs to keep track of "data state"; for example, a finite state machine which may be in a transient state between requests (lazy evaluation) or which takes a long time to initialize. State management in ASP.NET pages with authentication can make Web scraping difficult or impossible.

Application State

Application state is held by a collection of shared user-defined variables. These are set and initialized when the Application_OnStartevent fires on the loading of the first instance of the application and are available until the last instance exits. Application state variables are accessed using the Applications collection, which provides a wrapper for the application state variables. Application state variables are identified by name.

Session state

Server-side session state is held by a collection of user-defined session variables that are persistent during a user session. These variables, accessed using the Session collection, are unique to each session instance. The variables can be set to be automatically destroyed after a defined time of inactivity even if the session does not end. Client-side user session is maintained by either a cookie or by encoding the session ID in the URL itself.

ASP.NET supports three modes of persistence for server-side session variables:

In-Process Mode

The session variables are maintained within the ASP.NET process. This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down.

ASPState Mode

ASP.NET runs a separate Windows service that maintains the state variables, because state management happens outside the ASP.NET process, and because the ASP.NET engine accesses data using .NET Remoting, ASPState is slower than In-Process. This mode allows an ASP.NET application to be load-balanced and scaled across multiple servers. Because the state management service runs independently of ASP.NET, the session variables can persist across ASP.NET process shutdowns. However, since session state server runs as one instance, it is still one point of failure for session state. The session-state service cannot be load-balanced, and there are restrictions on types that can be stored in a session variable.

SqlServer Mode

State variables are stored in a database, allowing session variables to be persisted across ASP.NET process shutdowns. The main advantage of this mode is that it allows the application to balance load on a server cluster, sharing sessions between servers. This is the slowest method of session state management in ASP.NET.

View state

View state refers to the page-level state management mechanism, utilized by the HTML pages emitted by ASP.NET applications to maintain the state of the web form controls and widgets.

The state of the controls is encoded and sent to the server at every form submission in a hidden field known as __VIEWSTATE. The server sends back the variable so that when the page is re-rendered, the controls render at their last state. At the server side, the application may change the viewstate, if the processing requires a change of state of any control. The states of individual controls are decoded at the server, and are available for use in ASP.NET pages using the ViewState collection.

The main use for this is to preserve form information across postbacks. View state is turned on by default and normally serializes the data in every control on the page regardless of whether it is actually used during a postback. This behavior can (and should) be modified, however, as View state can be disabled on a per-control, per-page, or server-wide basis.

Developers need to be wary of storing sensitive or private information in the View state of a page or control, as the base64 string containing the view state data can easily be de-serialized. By default, View state does not encrypt the __VIEWSTATE value. Encryption can be enabled on a server-wide (and server-specific) basis, allowing for a certain level of security to be maintained.

Server-side caching

ASP.NET offers a "Cache" object that is shared across the application and can also be used to store various objects. The "Cache" object holds the data only for a specified amount of time and is automatically cleaned after the session time-limit elapses.

Template engine

When first released, ASP.NET lacked a template engine. Because the .NET Framework is object-oriented and allows for inheritance, many developers would define a new base class that inherits from "System.Web.UI.Page", write methods there that render HTML, and then make the pages in their application inherit from this new class. While this allows for common elements to be reused across a site, it adds complexity and mixes source code with markup. Furthermore, this method can only be visually tested by running the application - not while designing it. Other developers have used include files and other tricks to avoid having to implement the same navigation and other elements in every page.

ASP.NET 2.0 introduced the concept of "master pages", which allow for template-based page development. A web application can have one or more master pages, which, beginning with

ASP.NET 2.0, can be nested. Master templates have place-holder controls, called Content Place Holders to denote where the dynamic content goes, as well as HTML and JavaScript shared across child pages.

Child pages use those Content Place Holder controls, which must be mapped to the place-holder of the master page that the content page is populating. The rest of the page is defined by the shared parts of the master page, much like a mail merge in a word processor. All markup and server controls in the content page must be placed within the Content Place Holder control.

When a request is made for a content page, ASP.NET merges the output of the content page with the output of the master page, and sends the output to the user.

The master page remains fully accessible to the content page. This means that the content page may still manipulate headers, change title, configure caching etc. If the master page exposes public properties or methods (e.g. for setting copyright notices) the content page can use these as well.

Other files

Other file extensions associated with different versions of ASP.NET include:

App_Code

This is the "raw code" directory. The ASP.NET server automatically compiles files (and subdirectories) in this folder into an assembly which is accessible in the code of every page of the site. App_Code will typically be used for data access abstraction code, model code and business code. Also any site-specific http handlers and modules and web service implementation go in this directory. As an alternative to using App_Code the developer may opt to provide a separate assembly with precompiled code.

App_Data

Default directory for databases, such as Access mdb files and SQL Server mdf files. This directory is usually the only one with write access for the application.

App_LocalResources

E.g. a file called CheckOut.aspx.fr-FR.resx holds localized resources for the French version of the CheckOut.aspx page. When the UI culture is set to french, ASP.NET will automatically find and use this file for localization.

App_GlobalResources

Holds resx files with localized resources available to every page of the site. This is where the ASP.NET developer will typically store localized messages etc. which are used on more than one page.

App_Themes

Adds a folder that holds files related to themes which is a new ASP.NET feature that helps ensure a consistent appearance throughout a Web site and makes it easier to change the Web site's appearance when necessary.

Performance

ASP.NET aims for performance benefits over other script-based technologies (including Classic ASP) by compiling the server-side code to one or more DLL files on the web server. This compilation happens automatically the first time a page is requested (which means the developer need not perform a separate compilation step for pages). This feature provides the ease of development offered by scripting languages with the performance benefits of a compiled binary. However, the compilation might cause a noticeable but short delay to the web user when the newly-edited page is first requested from the web server, but won't again unless the page requested is updated further.

The ASPX and other resource files are placed in a virtual host on an Internet Information Services server (or other compatible ASP.NET servers; see Other implementations, below). The first time a client requests a page, the .NET Framework parses and compiles the file(s) into a .NET assembly and sends the response; subsequent requests are served from the DLL files. By default ASP.NET will compile the entire site in batches of 1000 files upon first request. If the compilation delay is causing problems, the batch size or the compilation strategy may be tweaked.

Developers can also choose to pre-compile their "code behind" files before deployment, using MS Visual Studio, eliminating the need for just-in-time compilation in a production environment. This also eliminates the need of having the source code on the web server.

5.2 Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Speech;
using System.Speech.Synthesis;
using System.Speech.Recognition;

namespace signal_sender
{
  public partial class Form1 : Form
  {
    SpeechRecognizer speechreco = new SpeechRecognizer();
    SpeechSynthesizer sp_sz = new SpeechSynthesizer();
    String opr = "";
    List<String> grammarList = new List<string>();
    public Form1()
    {
      InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
      try
      {
        serialPort1.Open();

      }
      catch (Exception err)
      {
        MessageBox.Show("port busy");
      }
      sp_sz.SpeakAsync("WELCOME TO SIGNAL SENDER");
      sp_sz.SpeakAsync("SPEAK ON TO START THE DEVICE");
      sp_sz.SpeakAsync("SPEAK STOP TO STOP THE DEVICE");
    try
    {
        Grammar CustomGrammar = CreateCustomGrammar();
        speechreco.UnloadAllGrammars();
        speechreco.LoadGrammar(CustomGrammar);
        speechreco.Enabled = true;
        speechreco.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(speechreco_SpeechRecognized);
    }
    catch (Exception ex)
    {
        sp_sz.SpeakAsync(ex.Message.ToString());
    }


}
void speechreco_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    String str = e.Result.Text;
    if (str == "stop")
    {
        try
        {
            sp_sz.Speak("PROGRAM STOPPED");
            sp_sz.Dispose();
            serialPort1.Write("S");
            speechreco.Dispose();
            //this.Close();
        }
        catch
        {
            sp_sz.SpeakAsync("PROBLEM OCCURED");
        }
    }
    else if (str == "one")
    {
        try
        {
            sp_sz.SpeakAsync("program started");
            serialPort1.Write("D");
            label4.Text = "ONE";
            speechreco.Dispose();
            SpeechRecognizer speechreco1 = new SpeechRecognizer();
            SpeechSynthesizer sp_sz1 = new SpeechSynthesizer();
            Grammar CustomGrammar1 = CreateCustomGrammar1();
            speechreco1.UnloadAllGrammars();
            speechreco1.LoadGrammar(CustomGrammar1);
            speechreco1.Enabled = true;
            speechreco1.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(speechreco1_SpeechRecognized);
          }
          catch
          {
            sp_sz.SpeakAsync("PROBLEM OCCURED");
          }

        }



      }
      private Grammar CreateCustomGrammar()
      {
        GrammarBuilder grammarBuilder = new GrammarBuilder();
        grammarBuilder.Append(new Choices("one" , "stop"));
        return new Grammar(grammarBuilder);
      }

      private void label1_Click(object sender, EventArgs e)
      {

      }
      private Grammar CreateCustomGrammar1()
      {
        GrammarBuilder grammarBuilder1 = new GrammarBuilder();
        grammarBuilder1.Append(new Choices( "one","two","three","stop","start"));
        return new Grammar(grammarBuilder1);
      }
      void speechreco1_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
      {
        String v_rs = e.Result.Text;
        if (v_rs =="one")
        {
          serialPort1.Write("D");
          label4.Text = "1";
        }
        if (v_rs =="two")
        {
          serialPort1.Write("C");
          label4.Text = "2";
        }
       if (v_rs =="three")
```

```
            {
              serialPort1.Write("B");
              label4.Text = "3";
            }
           if (v_rs =="stop")
           {
             serialPort1.Write("S");
             label4.Text = "STOP";
           }


        }
     }
  }
```

Chapter 6

6.1 WORKING

After switching on the power supply the LCD will display a message "VOICE BASED MOTOR CONTROL" and simultaneously a welcome message can be heard on the headphones.

After that u can speak "ONE", "TWO", or "THREE" in the microphone and if the voice is recognized by the system it will give a command to the microcontroller and it will turn on the motor attached to it. Different rpm has been set to rotate the motor at different speeds which can control by speaking one, two or three. In order to stop the motor speak "STOP" in the microphone and the motor will stop.

6.2 ALGORITHM

STEP 1: Reset pin of microcontroller goes high for initiating the execution of program and accept data.

STEP 2: Clear the display of LCD.

STEP 3: Check the input given by the computer on the input pin of microcontroller and drive the motor according to the given input.

Code:-

```
#define RS P35

#define RW P36

#define E P37

#define DATA P1

#define motor P: 3


unsigned char Ton=0;
```

```
#include<lcdrout.h>

void control_motor(void)

            {

             while(1)

            {

                        motor=0;                //motor is on

                        ms_delay(Ton);

                        motor=1;                //motor is off

                        ms_delay(100-Ton);

                }

}

void main()

{

unsigned char in=0;

lcd_initialize();

lcd_display("  SPEECH BASED       ",16);

ACC=0xc0;                                            //go to the second line of LCD

Lcd_cmd();

Lcd_display("MOTOR CONTROL       ",16);

Secdelay(3);

ACC=0x01;                                            //clear LCD
```

```
Lcd_cmd();

Secdelay(3);

While(1)

{

in=SBUF;

ACC=0x01;                                  //go to first line of LCD

Lcd_cmd();

If(in==’B’)                        //3

{

        ACC=0x80;                          //go to first line of LCD

        Lcd_cmd();

        Lcd_display(“Duty Cycle 075 “,16);

        Lcd_cmd1(0xc2);

        Lcd_puts(“RPM  1800”);             //diplay on LCD

        Ton=75;

}

If(in==’C’)                        //2

{

        ACC=0x80;

        Lcd_cmd();

        Lcd_display(“Duty Cycle 025 “,16)
```

```
            Lcd_cmd1(0xc2);

            Lcd_puts("RPM  600");

            Ton=40;

}

If(in=='D')                              //1

{

            ACC=0x80;

            Lcd_cmd();

            Lcd_display("Duty Cycle 015 ",16);

            Lcd_cmd1(0xc2);

            Lcd_puts("RPM  360");

            Ton=15;

}

If(in=='S')                              //stop

{

            ACC=0x82;

            Lcd_cmd();

            Lcd_display("STOP            ",16);

            ACC=0xc6;

            Lcd_cmd();

            Ton=0;
```

```
}
control_motor();
}
}
```

CONCLUSION

Finally we conclude that overall circuitry is able to take our voice command and according to that command any electrical appliance or electronic item which can be actuated is connected at output pin of microcontroller can be operated. Thus it can be applied in various devices example in household appliances, in industrial appliances, in robotics, etc.

BIBLOGRAPHY

- www.wikipedia.com
- www.8051.com
- www.datasheet.com
- www.webopedia.com
- www.fifthgen.com

BOOKS

- The 8051 microcontroller and embedded systems by Muhammad Ali Mazidi
- Power Electronic by P.S Bimbhra
- Electronics circuits and Devices by D.C Kulshreshtha