ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

Design and Implementation of an Autonomous Robot

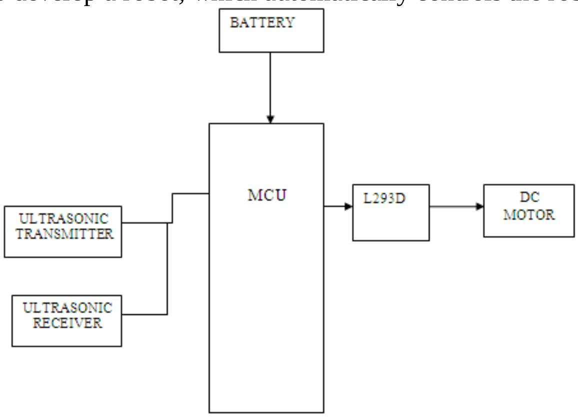
S. Sentil Kumar

Assistant professor, Department of Electronics and Instrumentation, Bharath University, Chennai, India

ABSTRACT : Automated systems have less manual operations, flexibility, reliability and accurate. Due to this demand every field prefers automated control systems. Especially in the field of electronics automated systems are giving good performance. In the present scenario of war situations, unmanned systems plays very important role to minimize human losses. So this robot is very useful to do operations like obstacle detection. This project aims at designing and executing the obstacle detection and avoidance robot. A robot obstacle detection system including a robot housing which navigates with respect to a surface and a sensor subsystem having a defined relationship with respect to the housing and aimed at the surface for detecting the surface. The ultrasonic sensor is a pair sensors has a receiver and a transmitter sensor. The transmitter sends the ultrasonic waves, and if the receiver senses any of the transmitted signal it indicates the presence of an obstacle. If the receiver doesn't sense any signal it indicates the absence of obstacle. If any obstacle is detected the directions of the robot will be automatically changed.

I. INTRODUCTION

The main aim of this project is to develop a robot, which automatically controls the robot using 8051 Micro controller.



An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular function. Since the embedded system is dedicated to specific tasks, design engineers can optimize it reducing the size and cost of the product and increasing the reliability and performance. Embedded systems are controlled by one or more main processing cores that is typically either a microcontroller or a digital signal processor (DSP). Embedded systems control many devices in common use today. The Keil C51 C Compiler for the 8051 microcontroller is the most popular 8051 C compiler in the world. It provides more features than any other 8051 C compiler available today. The C51 Compiler allows you to write 8051 microcontroller applications in C that, once compiled, have the efficiency and speed of assembly language. Language extensions in the C51 Compiler give you full access to all resources of the 8051.
The C51 Compiler translates C source files into relocatable object modules which contain full symbolic information for debugging with the µVision Debugger or an in-circuit emulator. In addition to the object file, the compiler generates a listing file which may optionally include symbol table and cross reference information.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

Embedded C is an extension for the programming language C to support embedded processors, enabling portable and efficient application programming for embedded systems. The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (EPROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pinout. 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 AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

This project aims at designing and executing the fire sensing, alcohol, bomb and obstacle detection and avoidance robot. A robot obstacle detection system including a robot housing which navigates with respect to a surface and a sensor subsystem having a defined relationship with respect to the housing and aimed at the surface for detecting the surface. The ultrasonic sensor is a pair sensors has a receiver and a transmitter sensor. The transmitter sends the ultrasonic waves, and if the receiver senses any of the transmitted signal it indicates the presence of an obstacle. If the receiver doesn't sense any signal it indicates the absence of obstacle. If any obstacle is detected the directions of the robot will be automatically changed. This robot is fitted with motors. A micro controller is used to control all operations. According to the motor operations the robot will operate as specified in program.

8051 Micro controller:

The first microprocessor introduced in 1981/1971, was made possible by high levels of integration of digital circuits. Continued integration of peripherals and memory on the same integrated circuit as the microprocessor core led to the creation of micro controllers. A micro controller is an integrated circuit composed of a CPU, various peripheral devices, and typically memory, all in one chip. Using one chip that contains all the necessary functions in place of a microprocessor and multiple peripheral chips has reduced the size and the power consumption of control oriented applications. A micro controller is different from a microprocessor both in hardware and software. In hardware it includes peripherals such as I/O, memory, and analog and digital interface. Micro controllers are more suited for small applications with specific control functions requiring specialized peripherals and interfaces.

They are designed for process control and are required to interface to the real world processes. Many of the peripheral devices integrated on a micro controller are for that specific purpose. Analog to digital converters perform the task of converting an analog signal to digital for use by the CPU, and digital to analog converters perform the task of converting digital data into analog value and waveforms to control analog functions. In addition to the analog interface, micro controllers contain peripheral devices that enable them to communicate to other digital components within a system or to monitor and control digital functions. Communication interfaces, digital I/O and interrupt controllers fall into this category of peripheral devices. Other peripheral devices often included on the same chip include clocks and timers.

In terms of the software, micro controllers have a more compact set of instructions with commands more suited to process control such as input and output from. Single bit operations such as set and reset, bit-wise logical functions or branching instructions that depend on a single bit are commonly available as part of the instruction set to allow for reading input switch status or on/off control of an external event. Since in a given application the micro controller is programmed for one task, it only has one control program. In a microprocessor based system various programs are stored in a mass storage device and then loaded into the RAM for execution. In contrast the micro controller program is typically stored in a ROM or PROM and RAM is used for temporary storage of data.

Compared with discrete implementation of a system, the micro controller based approach provides shorter system development time, reduced implementation cost, lower power consumption, and higher reliability. The only drawback, which is often not important, is the lower speed of execution. For example, for a micro controller system to perform a logical operation, several clock cycles are needed to read the inputs, perform the function and output the results. The same operation when implemented with discrete components will provide the results as soon as the signals have propagated through the logic gates.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

Micro-controllers are used in a variety of process control applications, replacing complex digital circuits and sometimes-analog functions while providing more flexibility due to their programmability. Portable electronic devices such as personal audio devices (CD players, MP3 players), mobile telephones, digital cameras and video camcorders rely heavily on the reduced size and low power consumption of micro controller based electronics. These features are crucial to applications like implantable medical devices such as pacemakers, or personal medical monitoring devices like glucometers (electronic devices used for the measurement of blood glucose). In other applications such as appliances, home audio and video, automotive, power management, and temperature control, using a micro controller results in reduced board level circuit complexity and consequently reduced cost. With the growing number of applications using micro controllers, it is not surprising that there are such a wide variety of these components. In addition to those commonly available, many manufacturers custom-design a micro controller to suit a specific application.

Architecture

Architecturally all micro controllers share certain features. They all contain a CPU, memory and I/O on the same chip. Another common feature is the interrupt handling capability. What sets them apart from one another is the choice of CPU, the structure of memory, and choice of peripheral devices, I/O and interrupts handling hardware. The major distinguishing architectural characteristic of micro controllers is the word size. Micro-controllers are available in 4, 8, 16, or 32 bit wide words. The width of the data path impacts several features of the micro controller. The complexity of the instruction set (number of available instructions and addressing modes), program efficiency (code generation and storage space), execution speed, as well as chip implementation and interfacing complexity are all influenced by the width of the data path.

For simple control tasks 4-bit, and for a vast number of control and measurement applications 8-bit micro controllers would be sufficient. For higher precision and speed applications like speech and video processing, or complex instrumentation, 16-bit and 32-bit micro controllers are more appropriate.

Another distinction between micro controllers is the instruction set. Micro-controllers with complex instruction set (CISC) provide capability to perform complex computations rapidly. The extensive set of instructions, allow complex operations to be performed with few instructions. On the other hand reduced instruction set computers (RISC) decrease program execution time by having fewer less complex instructions. Fewer available instructions results in faster execution due to smaller size of the op-code and less decoding time needed for each instruction. The trade-off depends on the complexity of operations needed for a specific application. In simple control applications a RISC based micro controller is more suitable because of its lower overhead for each instruction. In more complex applications, the availability of a more diverse instruction set results in a more efficient and faster executing code because fewer instructions are needed to accomplish a complicated task. For micro controller applications the instruction set should include common computational instructions plus instructions optimized for the specific application at hand.

Just as in microprocessors, micro controllers are also differentiated according to their memory structure. Von Neumann architecture maps the data and program to same memory address space. In the Harvard architecture the instructions are stored in a separate memory space than that used for data storage. Another memory related architectural characteristic of a processor is the addressing scheme. In linear addressing there is a one to one correspondence between an address and a memory location. So with an 8-bit address register, 2^8 distinct address locations can be accessed. In segmented addressing a separate register is used to point to a segment in memory, and the address register is used to point to an offset from that segment's start point. This way if all of the program or data are in the same segment, in order to access them, only the address register need to be used and the segment register can remain pointing to the start point of that segment.

Widely used group of micro controllers is Intel's MCS51 family. These micro controllers are also 8-bit processors, but with a separate 64Kbyte of data and 64Kbyte of program memory space. As implied by this statement, devices in the MCS51 utilize Harvard architecture. All of I/O addresses as well as CPU registers and various peripheral devices' registers are mapped in the same space as the data. The 8051, which is one of the options in this family, has 5 interrupt sources, 2 external, two timer interrupts and one serial port interrupt. Interrupt priority is resolved through a priority scheme and ranking in the polling sequence. The priority scheme allows each interrupt to be programmed to

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

one of two priority levels. Furthermore if two interrupts with the same priority occur simultaneously, they are serviced based on their rank in the polling sequence. Other manufacturers such as AMD, Dallas Semiconductor, Fujitsu and Philips also supply micro controllers in the MCS51 family.

II. BLOCK DIAGRAM

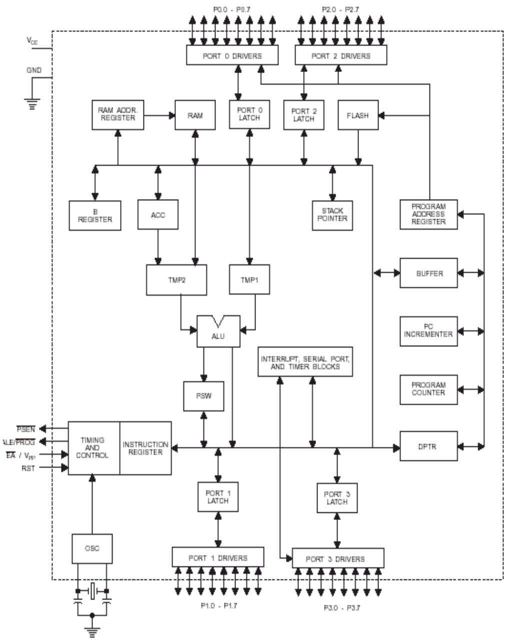


Fig Block Diagram of AT89C51 Microcontroller

Dallas Semiconductor's DC87C550 provides increased performance over Intel's 8051 while maintaining instruction set compatibility. Many instructions that execute in 12 CPU clock cycles in an 8051, will execute in only 4 clocks for the DC87C550 therefore resulting in increased execution speeds of up to three times. Additionally, the DC87C550 has a power management mode that allows slowing of the processor in order to reduce power consumption. This mode can be utilized in battery operated or otherwise low power applications. The architecture of the instruction set varies greatly from one micro controller to another. The choices made in designing the instruction set impact program memory space usage, code execution speed, and ease of programming.

Basic Registers

The Accumulator

The Accumulator, as its name suggests, is used as a general register to accumulate the results of a large number of instructions. It can hold an 8-bit (1-byte) value and is the most versatile register the 8051 has due to the shear number of instructions that make use of the accumulator. More than half of the 8051's 255 instructions manipulate or use the accumulator in some way. For example, if we add the number 10 and 20, the resulting 30 will be stored in the accumulator.

The "R" registers

The "R" registers are a set of eight registers that are named R0, R1, etc. up to and including R7. These registers are used as auxiliary registers in many operations. To continue with the above example, perhaps you are adding 10 and 20. The original number 10 may be stored in the Accumulator whereas the value 20 may be store
in, say, register R4. To process the addition you would execute the command:
ADD A,R4 After executing this instruction the Accumulator will contain the value 30.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

The "R" registers as very important auxiliary, or "helper", registers. The Accumulator alone would not be very useful if it were not for these "R" registers. The "R" registers are also used to temporarily store values.
MOV A, R3; Move the value of R3 into the accumulator
ADD A, R4; Add the value of R4
MOV R5, A; Store the resulting value temporarily in R5
MOV A, R; Move the value of R1 into the accumulator
ADD A,R2 ;Add the value of R2
SUBB A,R5 ;Subtract the value of R5 (which now contains R3 + R4)
In the above example we used R5 to temporarily hold the sum of R3 and R4. Of course, this isn't the most efficient way to calculate (R1+R2) **-** (R3 +R4) but it does illustrate the use of the "R" registers as a way to store values temporarily.

The "B" Register
The "B" register is very similar to the Accumulator in the sense that it may hold an 8-bit (1-byte) value. The "B" register is only used by two 8051 instructions: MUL AB and DIV AB. Thus, if you want to quickly and easily multiply or divide A by another number, you may store the other number in "B" and make use of these two instructions.
Aside from the MUL and DIV an instruction, the "B" register is often used as yet another temporary storage register much like a ninth "R" register.

The Data Pointer (DPTR)
The Data Pointer (DPTR) is the 8051's only user-accessible 16-bit (2-byte) register. The Accumulator, "R" registers, and "B" register are all 1-byte values. DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory. When the 8051 accesses external memory it will access external memory at the address indicated by DPTR. While DPTR is most often used to point to data in external memory, many programmers often take advantage of the fact that it's the only true 16-bit register available. It is often used to store 2-byte values which have nothing to do with memory locations.
The Program Counter (PC)
The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to execute is found in memory.
When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed. It is important to note that PC isn't always incremented by one. Since some instructions require 2 or 3 bytes the PC will be incremented by 2 or 3 in these cases. The Program Counter is special in that there is no way to directly modify its value. That is to say, you can't do something like PC=2430h. On the other hand, if you execute LJMP 2340h you've effectively accomplished the same thing. It is also interesting to note that while you may change the value of PC (by executing a jump instruction, etc.) there is no way to read the value of PC. That is to say, there is no way to ask the 8051.
Events that Trigger Interrupts
The 8051 can be configured so that any of the following events will cause an interrupt:
• Timer 0 Overflow.
• Timer 1 Overflow.
• Reception/Transmission of Serial Character.
• External Event 0.
• External Event 1.
In other words, we can configure the 8051 so that when Timer 0 overflows or when a character is sent/received, the appropriate interrupt handler routines are called. Obviously we need to be able to distinguish between various interrupts and executing different code depending on what interrupt was triggered. This is accomplished by jumping to a fixed address when a given interrupt occurs.

Interrupt	Flag	Interrupt Handler Address
External 0	IE0	0003h
Timer 0	TF0	000Bh
External 1	IE1	0013h
Timer 1	TF1	001Bh
Serial	RI/TI	0023h

Tab Interrupt Handler Address and the Interrupts associated to them

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

By consulting the above chart we see that whenever Timer 0 overflows (i.e., the TF0 bit is set), the main program will be temporarily suspended and control will jump to 00BH. It is assumed that we have code at address 0003H that handles the situation of Timer 0 overflowing.

Setting up Interrupts

By default at power up, all interrupts are disabled. This means that even if, for example, the TF0 bit is set, the 8051 will not execute the interrupt. Your program must specifically tell the 8051 that it wishes to enable interrupts and specifically which interrupts it wishes to enable. Your program may enable and disable interrupts by modifying the IE SFR (A8h):

Bit	Name	Bit Address	Explanation of Function
7	EA	AFh	Global Interrupt Enable/Disable
6	-	AEh	Undefined
5	-	ADh	Undefined
4	ES	ACh	Enable Serial Interrupt
3	ET1	ABh	Enable Timer 1 Interrupt
2	EX1	AAh	Enable External 1 Interrupt
1	ET0	A9h	Enable Timer 0 Interrupt
0	EX0	A8h	Enable External 0 Interrupt

Tab Setting up Interrupts

For example, to enable Timer 1 Interrupt, you would execute either:
MOV IE, #08h || SETB ET1
Both of the above instructions set bit 3 of IE, thus enabling Timer 1 Interrupt. Once Timer 1 Interrupt is enabled, whenever the TF1 bit is set, the 8051 will automatically put "on hold" the main program and execute the Timer 1 Interrupt Handler at address 001Bh. However, before Timer 1 Interrupt (or any other interrupt) is truly enabled, you must also set bit 7 of IE. Bit 7, the Global Interrupt Enable/Disable, enables or disables all interrupts simultaneously. That is to say, if bit 7 is cleared then no interrupts ill occur, even if all the other bits of IE are set. Setting bit 7 will enable all the interrupts that have been selected by setting other bits in IE. This is useful in program execution if you have time-critical code that needs to execute.

In this case, you may need the code to execute from start to finish without any interrupt getting in the way. Accomplish this you can simply clear bit 7 of IE (CLR EA) and then set it after your timecriticial code is done. So, to sum up what has been stated in this section, to enable the Timer 1 Interrupt the most common approach is to execute the following two Instructions:
SETB ET1
SETB EA
Thereafter, the Timer 1 Interrupt Handler at 01Bh will automatically be called whenever the TF1 bit is set (upon Timer 1 overflow).

Interrupt Priorities

The 8051 offer two levels of interrupt priority: high and low. By using interrupt priorities you may assign higher priority to certain interrupt conditions. For example, you may have enabled Timer 1 Interrupt, which is automatically called every time Timer 1 overflows. Additionally, we may have enabled the Serial Interrupt, which is called every time a character is received via the serial port. However, you may consider that receiving a character is much more important than the timer interrupt. In this case, if Timer 1 Interrupt is already executing you may wish that the serial interrupt itself interrupts the Timer 1 Interrupt.

When the serial interrupt is complete, control passes back to Timer 1 Interrupt and finally back to the main program. You may accomplish this by assigning a high priority to the Serial Interrupt and a low priority to the Timer 1 Interrupt. Interrupt priorities are controlled by the **IP** SFR (B8h).

The IP SFR has the following format:

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

Bit	Name	Bit Address	Explanation of Function
7	-	-	Undefined
6	-	-	Undefined
5	-	-	Undefined
4	PS	BCh	Serial Interrupt Priority
3	PT1	BBh	Timer 1 Interrupt Priority
2	PX1	BAh	External 1 Interrupt Priority
1	PT0	B9h	Timer 0 Interrupt Priority
0	PX0	B8h	External 0 Interrupt Priority

Tab Interrupt Priorities

Bit Name Bit Address Explanation of Function
7 - - Undefined
6 - - Undefined
5 - - Undefined
4 PS BCh Serial Interrupt Priority
3 PT1 BBh Timer 1 Interrupt Priority
2 PX1 BAh External 1 Interrupt Priorities
1 PT0 B9h Timer 0 Interrupt Priority
0 PX0 B8h External 0 Interrupt Priority
When considering interrupt priorities, the following rules apply:
• Nothing can interrupt a high-priority interrupt-- not even another high priority interrupt.
• A high-priority interrupt may interrupt a low priority interrupt.
• A low-priority interrupt may only occur if no other interrupt is already executing.
• If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority the interrupt, which is serviced first by polling sequence, will be executed first.
When an interrupt is triggered, the following actions are taken automatically by the Micro controller:
• The current Program Counter is saved on the stack, low-byte first.
• Interrupts of the same and lower priority are blocked.
• In the case of Timer and External interrupts, the corresponding interrupt flag is set.
• Program execution transfers to the corresponding interrupt handler vector address.
• The Interrupt Handler Routine executes. Take special note of the third step: If the Interrupt being handled is a Timer or External interrupt; the micro controller automatically clears the interrupt flag before passing control to your interrupt handler routine.
An interrupt ends when your program executes the RETI instruction. When the RETI Instruction is executed the micro controller takes the following actions:
• Two bytes are popped off the stack into the Program Counter to restore normal program Execution.
• Interrupt status is restored to its pre-interrupt status.

Serial Interrupts
Serial Interrupts are slightly different than the rest of the interrupts. This is due to the fact that there is two interrupt flags: RI and TI. If either flag is set, a serial interrupt is triggered. As you will recall from the section on the serial port, the RI bit is set when a byte is received by the serial port and the TI bit is set when a byte has been sent. This means that when your serial interrupt is executed, it may have been triggered because the RI flag was set or because the TI flag was set-- or because both flags were set.
Thus, the routine must check the status of these flags to determine that action is appropriate. Also, since the 8051 does not automatically clear the RI and TI flags
You must clear these bits in your interrupt handler.
INT_SERIAL: JNB RI, CHECK_TI; if the RI flag is not set, we jump to check TI
MOV A, SBUF; If we got to this line, it's because the RI bit *was* set
CLR RI; Clear the RI bit after we've processed it
CHECK_TI: JNB TI, EXIT_INT; if the TI flag is not set, we jump to the exit point
CLR TI; Clear the TI bit**) Universal Asynchronous Receive Transmit (UART)**

The UART provides means of asynchronous serial communication between devices or systems. It is essentially a parallel to serial and serial to parallel converter that conforms to a certain protocol for coding the data and interface specifications.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

b) Serial Peripheral Interface (SPI)

SPI is used for synchronous serial communication. Because of its synchronous nature it uses a separate connection for clock. Additionally it requires a transmit data, receive data, and enable. SPI interfaces run as fast as 10MHz,which is why high density EEPROM are increasingly using this serial interface method.

c) I2C

I^2C uses a bi-directional 3-wire (including ground) bus for communication between multiple devices. Communication protocol is based on a master/slave relationship. The maximum number of devices is limited by the 16K address space of the protocol and the maximum allowable capacitance on the lines (400pF). The original standard had a 100 kHz maximum clock speed. The low pin count associated with the I^2C has made it the industry standard for serial interface to EEPROM chips. The drawback with I^2C interface is its inherent intolerance to noise. Enhanced I^2C schemes extend the address space to about 512K and the maximum clock speed to about 400kHz.

e) Other serial interface standards

Other serial interfaces have been developed that specialize in certain functionality. Controller Area Network (CAN) was designed to operate in noisy environments such as in automobiles and industrial applications. Universal Serial Bus (USB) and IEEE 1394 are two serial interface standards that address interface speed issues. USB 2.0 supports data rates as high as 480 Mbps and IEEE 1394b supports data rates greater than 1Gbps.

6**. Timers and Clocks**

a) **General Purpose Timer** : A free running timer can be used to keep track of time of day and the date. A timer can also be used for precise measurement of time. For example, using an onboard timer and digital I/O, A dual slope ADC can be externally implemented using few analog components where the timer is used to set the integration period and measure the de-integration time.

b) **Watch Dog Timer** (WDT) : A timer can be used to verify proper operation of the CPU. This is typically done using a WDT. WDT operates by continuously incrementing a count value stored in the WDT register. If the value in the WDT register reaches a preset final count, an interrupt is generated which indicates a fault condition. During normal operation, the software should prevent the WDT register from reaching its final count by periodically resetting it to zero. For development or testing activities the WDT should be disabled.

c) **Real Time Clock** (RTC) : RTC is a programmable timer that is used to perform a certain task at regular intervals. For example to sample an analog waveform at regular intervals with an Analog to Digital Converter, an RTC can be programmed to generate interrupts at the sampling rate. The interrupt service routine will then activate the ADC and after completion of each conversion stores the result in an array.

7. **Memory :** Most often all the memory required for the operation of a micro controller is included on board. Program is usually stored in non-volatile memory such as ROM. In that situation the program has to be fully tested before committing it to silicon. Micro controllers are usually equipped with an emulation mode that enables access to external memory. This mode of operation can be used for program development or debugging. Other forms of memory used in micro controllers include EEPROM and RAM. EEPROM is used for non-volatile storage of variables such as calibration data and system settings. RAM is used for temporary storage of variables.

External Device Drivers

a**) LCD Interface**: Liquid Crystal Display drivers consisting of logic, signal level generation and row and column drivers may be included on the micro controller chip. LCD interface usually involves a large number of pins for the LCD row and column drivers. Including LCD driver on the chip results in a significant increase in the package pin count.

b**) LED Interface**: LED's are used for status indicator or signal transmission. Special high current drivers are needed to handle the large current required by the LED. Integrating the driver on the micro controller simplifies system level design but the large currents can complicate the design of the chip.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

DC Motor:

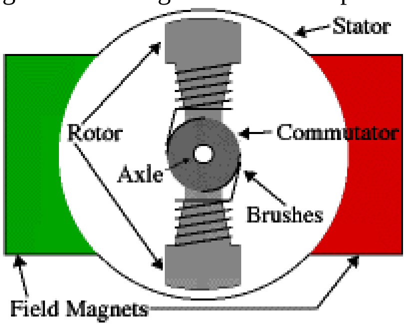
DC motors are configured in many types and sizes, including brush less, servo, and gear motor types. A motor consists of a rotor and a permanent magnetic field stator. The magnetic field is maintained using either permanent magnets or electromagnetic windings. DC motors are most commonly used in variable speed and torque.

Motion and controls cover a wide range of components that in some way are used to generate and/or control motion. Areas within this category include bearings and bushings, clutches and brakes, controls and drives, drive components, encoders and resolves, Integrated motion control, limit switches, linear actuators, linear and rotary motion components, linear position sensing, motors (both AC and DC motors), orientation position sensing, pneumatics and pneumatic components, positioning stages, slides and guides, power transmission (mechanical), seals, slip rings, solenoids, springs.

Motors are the devices that provide the actual speed and torque in a drive system. This family includes AC motor types (single and multiphase motors, universal, servo motors, induction, synchronous, and gear motor) and DC motors (brush less, servo motor, and gear motor) as well as linear, stepper and air motors, and motor contactors and starters.

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

Let's start by looking at a simple 2-pole DC electric motor (here red represents a magnet or winding with a "North" polarization, while green represents a magnet or winding with a "South" polarization).



Every DC motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that Beamers will see), the external magnetic field is produced by high-strength permanent magnets[1]. The stator is the stationary part of the motor -- this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor (together with the axle and attached commutator) rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. The above diagram shows a common motor layout -- with the rotor inside the stator (field) magnets.

The geometry of the brushes, commutator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commutator contacts, and energize the next winding. Given our example two-pole motor, the rotation reverses the direction of current through the rotor winding, leading to a "flip" of the rotor's magnetic field, and driving it to continue rotating.

In real life, though, DC motors will always have more than two poles (three is a very common number). In particular, this avoids "dead spots" in the commutator. You can imagine how with our example two-pole motor, if the rotor is exactly at the middle of its rotation (perfectly aligned with the field magnets), it will get "stuck" there. Meanwhile, with a two-pole motor, there is a moment where the commutator shorts out the power supply (i.e., both brushes touch both

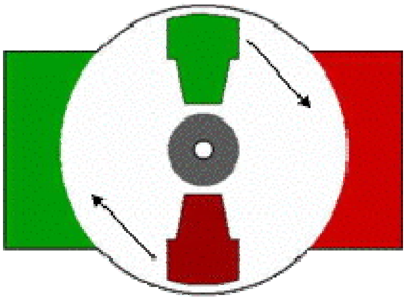
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

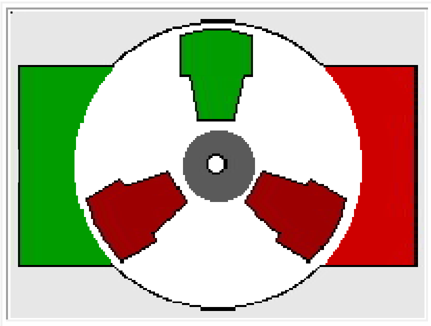
(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

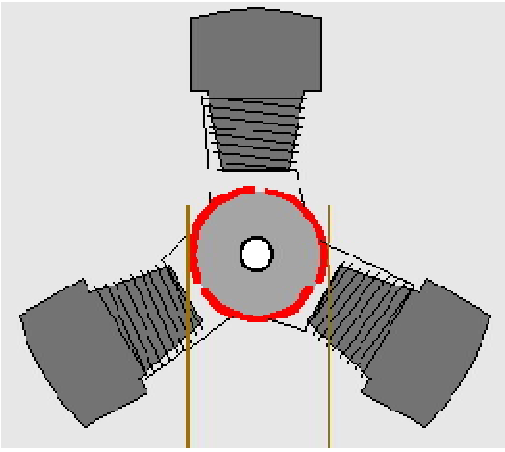
commutator contacts simultaneously). This would be bad for the power supply, waste energy, and damage motor components as well.



So since most small DC motors are of a three-pole design, let's tinker with the workings of one via an interactive animation (JavaScript required):



You'll notice a few things from this -- namely, one pole is fully energized at a time (but two others are "partially" energized). As each brush transitions from one commutator contact to the next, one coil's field will rapidly collapse, as the next coil's field will rapidly charge up (this occurs within a few microsecond). We'll see more about the effects of this later, but in the meantime you can see that this is a direct result of the coil windings' series wiring:



There's probably no better way to see how an average dc motor is put together, than by just opening one up. Unfortunately this is tedious work, as well as requiring the destruction of a perfectly good motor.
This is a basic 3-pole dcmotor, with 2 brushes and three commutator contacts.

PWM technique:

A pulse width modulator (PWM) is a device that may be used as an efficient light dimmer or DC motor speed controller. A PWM works by making a square wave with a variable on-to-off ratio; the average on time may be varied from 0 to 100 percent. In this manner, a variable amount of power is transferred to the load. The main advantage of a

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

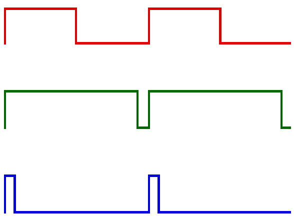
International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

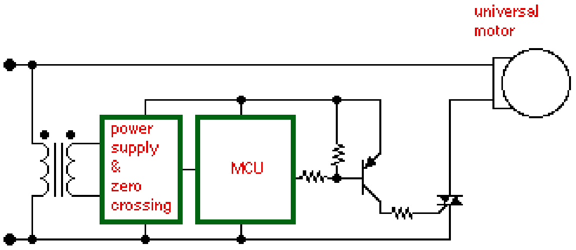
Vol. 4, Issue 2, February 2015

PWM circuit over a resistive power controller is the efficiency, at a 50% level, the PWM will use about 50% of full power, almost all of which is transferred to the load, a resistive controller at 50% load power would consume about 71% of full power, 50% of the power goes to the load and the other 21% is wasted heating the series resistor. Load efficiency is almost always a critical factor in solar powered and other alternative energy systems. One additional advantage of pulse width modulation is that the pulses reach the full supply voltage and will produce more torque in a motor by being able to overcome the internal motor resistances more easily. Finally, in a PWM circuits, common small potentiometers may be used to control a wide variety of loads whereas large and expensive high power variable resistors are needed for resistive controllers.

Pulse width modulation consists of three signals, which are modulated by a square wave. The duty cycle or high time is proportional to the amplitude of the square wave. The effective average voltage over one cycle is the duty cycle times the peak-to-peak voltage. Thus, the average voltage follows a square wave. In fact, this method depends on the motor inductance to integrate out the PWM frequency.



A very simply off line motor drive can be built using a TRIAC and a control IC. This circuit can control the speed of a universal motor. A universal motor is a series wound DC motor. The circuit uses phase angle control to vary the effective motor voltage.



A micro controller can also be used to control a triac. A PNP of transistor may be used to drive the triac. As shown, the MCU ground is connected to the AC line. The gate trigger current is lower if instead the MCU 5V supply is connected to the AC line. The MCU must have some means of detecting zero crossing and a timer, which can control the triac firing. A general-purpose timer with one input capture and one output compare makes an ideal phase angle control.

L293D IC (DC MOTOR DRIVER)

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

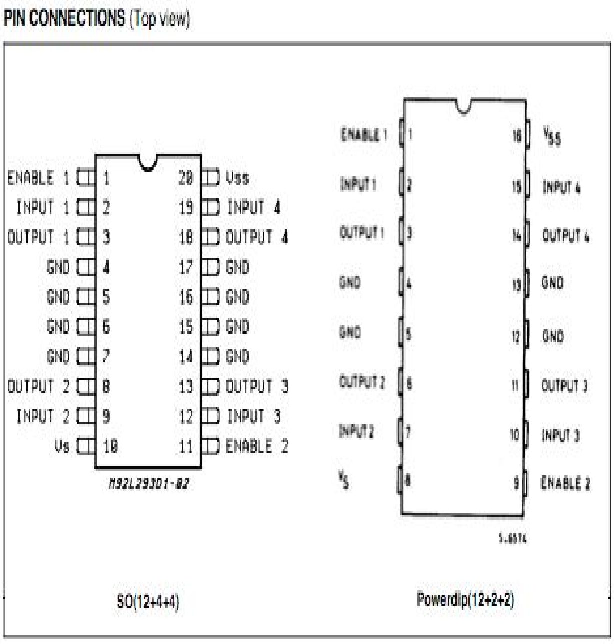
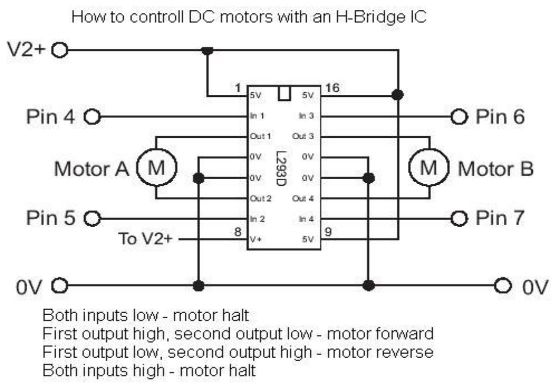


FIGURE: L293 & L293D Driver ICs

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications. All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications. On the L293, external high-speed output clamp diodes should be used for inductive transient suppression. A VCC1 terminal, separate from VCC2, is provided for the logic inputs to minimize device power dissipation. The L293and L293D are characterized for operation from 0°C to 70°C.

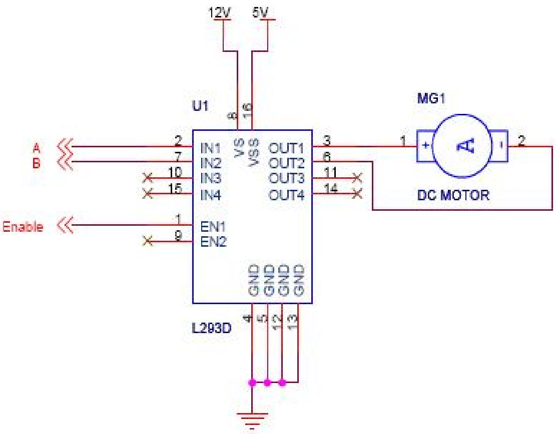


ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015



Truth Table

A	B	Description
0	0	Motor stops or Breaks
0	1	Motor Runs Anti-Cloclwise
1	0	Motor Runs Clockwise
1	1	Motor Stops or Breaks

For above truth table, the Enable has to be Set (1). Motor Power is mentioned 12V, but you can connect power according to your motors.

LED (Light Emitting Diodes):

As its name implies it is a diode, which emits light when forward biased. Charge carrier recombination takes place when electrons from the N-side cross the junction and recombine with the holes on the P side. Electrons are in the higher conduction band on the N side whereas holes are in the lower valence band on the P side. During recombination, some of the energy is given up in the form of heat and light. In the case of semiconductor materials like Gallium arsenide (GaAs), Gallium phoshide (Gap) and Gallium arsenide phoshide (GaAsP) a greater percentage of energy is released during recombination and is given out in the form of light. LED emits no light when junction is reverse biased.

III. ULTRASONIC SENSOR

Specifications:
Power Voltage: DC 6-12V
Quiescent Current: less than 2mA
Output Level: high 5V
Output Level: Low 0V
Sensing Angle: no greater than 15°

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2015

Sensing Distance: 2mm-3m
Note: The sensing distance varies by smoothness of different surfaces.
Ultrasonic sensors (also known as **tranceivers** when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.
This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms and non-destructive testing.
Systems typically use a transducer which generates sound waves in the ultrasonic range, above 20,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed

IV. CONCLUSION

The project "**AUTONOMOUS ROBOT**" has been successfully designed and tested. It has been developed by integrating features of all the hardware components used. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced IC's and with the help of growing technology the project has been successfully implemented. Finally we conclude that embedded system is an emerging field and there is a huge scope for research and development.

REFERENCES

1. The 8051 Micro controller and Embedded Systems -Muhammad Ali Mazidi, Janice Gillispie Mazidi
2. The 8051 Micro controller Architecture, Programming & Applications -Kenneth J.Ayala
3. Fundamentals Of Micro processors and Micro computers - B.Ram Micro processor Architecture, Programming & Applications -Ramesh S.Gaonkar Electronic Components - D.V.Prasad
4. Wireless Communications - Theodore S. Rappaport
5. Mobile Tele Communications - William C.Y. Lee