**A PROJECT REPORT**

**On**

ARDUINO BASED REMOTE CONTROLLED TEMPERATURE PROBING ROBOT USING BLUETOOTH AND TILT SENSOR

*Submitted by*

**KONATHALA PRIYANKA (Regd.no.314106816012)**

**CHINTALAPATI SANJITHA (Regd.no.314106816025)**

**VANTHINIBA RAJA (Regd.no.314106816027)**

**AKSHITH SUMANTH (Regd.no.314506816001)**

*In partial fulfillment for the award of the degree*

*Of*

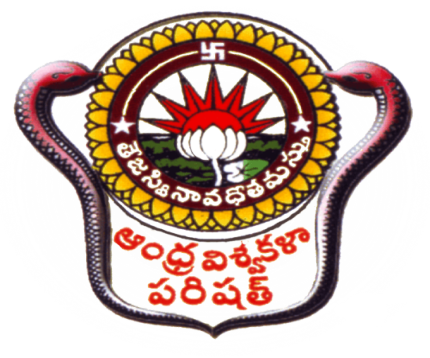
**BACHELOR OF TECHNOLOGY**

**IN**

**INSTRUMENTATION ENGINEERING**

***Under the guidance***

***Of***

***Dr. M. RAMESH PATNAIK***

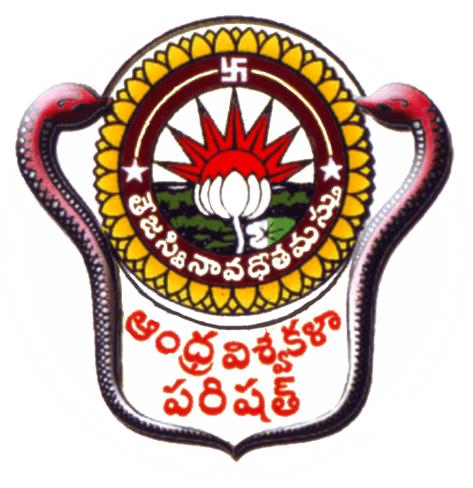
**DEPARTMENT OF INSTRUMENT TECHNOLOGY**

**ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)**

**2017-2018**

**DEPARTMENT OF INSTRUMENT TECHNOLOGY**

**ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)**

****

**Certificate**

This is to certify that this project report entitled **“ARDUINO BASED REMOTE CONTROLLED TEMPERATURE PROBING ROBOTUSING BLUETOOTH AND TILT SENSOR”** submitted by **K.PRIYANKA (314106816012), C.SANJITHA (314106816025), V.RAJA (314106816027)** and **K. AKSHITH SUMANTH (314506816001)** in the partial fulfillment of **BACHELOR OF TECHNOLOGY** in **INSTRUMENTATION ENGINEERING, ANDHRA UNIVERSITY, VISAKHAPATNAM,** during the academic year **2017-2018**, is a bonafied record of work carried under my guidance and supervision.

**Dr. M. RAMESH PATNAIK Prof D. RAMAKOTI REDDY**

Project guide Head of the Department

Dept. Of Instrument Technology Dept.Of Instrument Technology Andhra University College of Engg. Andhra University College of Engg.

Visakhapatnam. Visakhapatnam.

**Acknowledgment**

We would like to express our sincere gratitude to our project guide “**DR. RAMESH PATNAIK**” for giving the opportunity to work on this topic. We are thankful to him for encouraging us and giving us timely suggestions to complete this time bound project in time.

We also thank all teaching and nonteaching staff of the Department for cooperating with us to make our project a successful one.

**ABSTRACT**

Arduino based remote controlled temperature probing robot moves based on the commands received by Bluetooth module and it can be controlled by MEMS sensor in the mobile i.e., according to the movement of mobile the robot moves in forward, backward, Left and right directions.. It gives the continuous readings of temperature at remote locations .the robot has two D.C motors. These motors move in 2D direction with the help of driver IC L293D according its input signals.

In this project based on the tilt angle of the phone, the robot moves. So by mobile tilting positions, we can operate the robot.

The temperature sensor used in this sensor is LM35.this robot is used in industrial applications, military applications, and construction vehicles in civil side, medical applications, for transport and to measure the temperature in the inaccessible locations.

**INTRODUCTION**

A robot is a mechanical artificial agent in practice, it is usually an electro-mechanical machine which is guided by computer or electronic programming, and is thus able to do tasks on its own. Another common characteristic is that by its appearance or movements, a robot often conveys sense that it has intent or agency of its own. The robotic industries association define robot as follows: “a robot is programmable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.” Recently however, the industry’s current working definition of a robot has come to be understood as a piece of equipment that has three or more degrees of freedom. Robotics is a increasingly visible and important component of modern business, especially in certain industries. robotics oriented production processes are most obvious in factories and manufacturing facilities in fact, approximately 90 percent of all robots in operation today can be found in such facilities.

In many application of controlling gadget it becomes quite hard and complicated when there comes the part of controlling it with remote or any other switches mostly in military applications, industrial robotics, construction vehicles in civil side, medical application for surgery. in this field it is quite complicated to control the robot or particular machine with remote or switches. So a new concept is introduced to control the machine with the movement of mobile which will simultaneously control the movement of robot.

**APPLICATIONS:**

* Industrial applications
* Automatic control systems

**POWER SUPPLY BLOCK DIAGRAM:**

batteries

Filter

Regulator

Output

Bridge Rectifier

BLOCK DIAGRAM:

LM35

MOTOR2

MOTOR1

L

2

9

3

D

A0

A

T

M

E

G

A

3

2

8

Oscillator circuit

RESET CIRCUIT

BT HC-05

HC-

**THEORY**

**INTRODUCTION TO EMBEDDED SYSTEM**

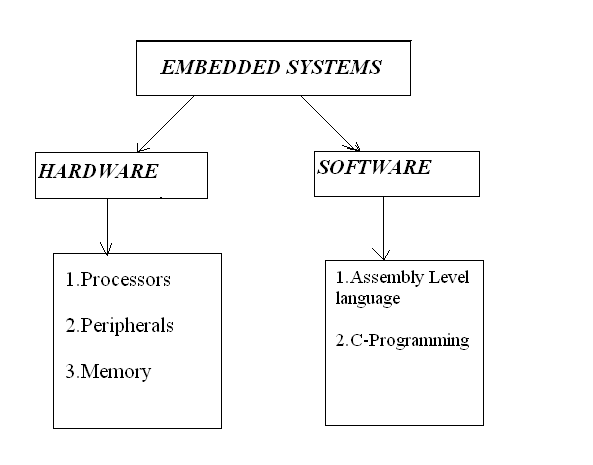
Embedded systems are a system is which it performs a specific or a pre-defined task. It is the combinations of hardware and software. It is nothing but a computer inside a product. It is a programmable hardware design nothing but an electronic chip.

A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. “Embedded” reflects the fact that they are an integral part of the system. In many cases their embeddedness may be such that their presence is far from obvious to the casual observer and even the equipment for sometime before being able to conclude that an embedded control system was involved in its functioning. At the other extreme a general-purpose computer may be used to control the operation of a large complex processing plant, and its presence will be obvious.

All embedded systems are or include computers or microprocessors. Some of these computers are however very simple systems as compared with a personal computer.

The very simplest embedded systems are capable of performing only single functions to meet single functions to meet a single predetermined purpose. In more complex systems an application program that enables the embedded system to be used for a particular purpose in a specific application determines the functioning of the embedded systems. The ability to have programs means that the same embedded system can be used for a variety of different purpose. In some cases a microprocessor may be designed in such way that application software for a particular purpose can be added to the basic software in a second process, after which it is not possible to make further changes. The applications software on such processors is sometimes referred to as firmware.

The simplest devices consist of a single microprocessor (often called a “chip”), which may itself be packaged with other chips in a hybrid systems or Application Specific Integrated Circuit (ASIC). Its input comes from a detector or sensor and its output goes to a switch or a activator which (for example) may start or stop the operation of a machine or, operating a value, may control the flow of fuel to an engine.



**PROCESSORS:**

Processors are the ones which performs some specific task or operation. These are divided in to several types like:

1. Digital Signal Processors.
2. Application Specific Integrated Circuits.
3. Micro Processors.
4. Micro Controllers.

**Digital Signal Processors:**

This is a one which performs scientific and mathematical

Calculations. The output of this type of processors will be in floating point values. Hence, we can get accurate values as outputs.

Best example of a Digital Signal Processors will be Weather Forecasting.

**Applications specific Integrated Circuits:**

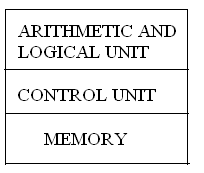
As the name itself specifies, this is integrated circuit designed for a specific application. IC designed for one specific application cannot be used in other applications.

Best Example for ASIC will be Cell phone card.

**Microprocessors:**

These are the ones which perform arithmetic and logical operations.

**Block Diagram of Microprocessor:**

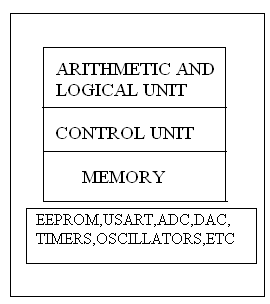


Arithmetic and logical unit performs arithmetical and logical calculations. Control unit controls all the peripheral devices connected to the microprocessors. Memory is a one which is used to store some data or information.

Best Example for a microprocessor will be our Personal Computer.

**Micro Controllers:**

These are the ones which are similar to that of a microprocessor which performs arithmetic and logical calculations. These have additional advantage to that it is having additional inbuilt features like:



1. Electrically Erasable Programmable Read Only Memory (EEPROM)
2. Universal Synchronous Asynchronous Receiver and Transmitter (USART).
3. Analog to Digital Converter (ADC).
4. Digital to Analog Converter (DAC).
5. Oscillators.
6. Timers.
7. Many others.

The output of a microcontroller is a always in integer format only. It cannot provide accurate values or floating point values.

Best Example for Micro Controller will be Traffic Light Controller.

Out of all these Processors, we are using Micro Controllers.

Coming to the software point of View, we have many software languages. Out of all, we are using Assembly Level Language to load the program in to the controller.

ALP coding is done and loaded into the microcontroller. These are different companies that manufacture Microcontrollers like Atmel, Motorola, Intel, Philips, PIC, etc.

We are using Atmel manufactured microcontrollers. Atmel Company manufactures different series of microcontrollers like

AT89C, AT90S, ATMega, ATCAN. Here we are using ATMega Microcontrollers. Again in ATMega we are having different ATMega 8, ATMega 8515, ATMega 16, ATMega 32,

AT Mega 162v, etc.Here, in this project we are using ATMega 328 Microcontroller.

**Introduction to applications of Embedded System:**

Embedded Controllers may be found in many different kinds of system and are used for many different applications. The list, which follows, is a indicative rather than exhaustive. An item in the list may be relevant to a particular company because either

1. It is or involves a core process or product,
2. It is or involves an ancillary function or services performed by the company or

(c) It refers to a product or services provided by a contractor under some form of agreement and the vulnerability of the supplier may need to be considered.

**List of Applications of Embedded Systems:**

1. **Manufacturing and process control:**

* Manufacturing plants.
* Water and sewage systems.
* Power stations.
* Power grid systems.
* Oil refineries and related storage facilities.
* Bottling plants.
* Automated factories.
* Simulators.
* Test equipment for control system development, maintenance and testing.

1. **Construction industry:**

* Surveying and location Equipment.
* Construction Plant.

1. **Transport:**

* Aero planes
* Trains and Buses.
* Marine craft (known cases include: radar mapping; ballast monitoring; cargo loading; ship main control system)
* Automobiles
* Fuel services.
* Air Traffic Control Systems.
* Signaling system.
* Radar Systems.
* Traffic Lights.
* Ticketing systems and Machines.
* Car Parking and other meters.

1. **Buildings and premises:**

* Electrically supply- supply, measurement, control, protection.
* Backing lighting and generators.
* Fire control systems.
* Heating and ventilating systems.
* Lifts Elevators, escalators.
* Security Systems.
* Security Cameras.
* Door locks.

1. **Domestic service:**

* Catering.
* Cleaning.

1. **Communications:**

* Telephone.
* Cable systems.
* Telephone switches.
* Satellites.

1. **Office systems and mobile management:**

* Telephone systems
* Faxes and Copier.
* Time recording systems.
* Mobile telephones
* Still and Video Cameras.

1. **Banking, Finance and Commercial:**

* Automated teller systems.
* Credit card Systems.
* Point of scale systems including scanner / cash systems

Embedded systems are electronic devices that incorporate microprocessors with in Their implementations. The main purposes of the microprocessors are to simplify the system design and provide flexibility. Having a microprocessor in the device means that removing the bugs, making modifications, or adding new features are only matters of rewriting the software that controls the device. Or in other words embedded computer systems are electronic systems that include a microcomputer to perform a specific dedicated application. The computer is hidden inside these products. Embedded systems are ubiquitous. Every week millions of tiny computer chips come pouring out of factories finding their way into our everyday products.

**Embedded systems** are self-contained programs that are embedded within a piece of hardware. Whereas a regular computer has many different applications and software that can be applied to various tasks, embedded systems are usually set to a specific task that cannot be altered without physically manipulating the circuitry. Another way to **think of an embedded** **system** is as a computer system that is created with optimal efficiency, thereby allowing it to complete specific functions as quickly as possible.

**Embedded systems designers** usually have a significant grasp of hardware technologies. They used specific programming languages and software to **develop embedded systems** and manipulate the equipment. When searching online, companies offer embedded systems development kits and other **embedded systems tools** for use by engineers and businesses.

**Embedded systems technologies** are usually fairly expensive due to the necessary development time and built in efficiencies, but they are also highly valued in specific industries. Smaller businesses may wish to hire a consultant to determine what sort of **embedded systems** will add value to your organization.

## Characteristics

Two major areas of differences are cost and power consumption. Since many embedded systems are produced in the tens of thousands to millions of units range, reducing cost is a major concern. Embedded systems often use a (relatively) slow processor and small memory size to minimize costs.

The slowness is not just clock speed. The whole architecture of the computer is often intentionally simplified to lower costs. For example, embedded systems often use peripherals controlled by synchronous serial interfaces, which are ten to hundreds of times slower than comparable peripherals used in PCs.

Programs on an embedded system often must run with real-time constraints with limited hardware resources: often there is no disk drive, operating system, keyboard or screen.

A flash drive may replace rotating media, and a small keypad and LCD screen may be used instead of a PC's keyboard and screen.

Firmware is the name for software that is embedded in hardware devices, e.g. in one or more ROM/Flash memory IC chips.

Embedded systems are routinely expected to maintain 100% reliability while running continuously for long periods, sometimes measured in years. Firmware is usually developed and tested too much stricter requirements than is general-purpose software, which can usually be easily restarted if a problem occurs.

### Platform

There are many different CPU architectures used in embedded designs. This in contrast to the desktop computer market, which as of this writing (2003) is limited to just a few competing architectures, mainly the Intel/AMD x86, and the Apple/Motorola/IBM PowerPC, used in the Apple Macintosh.

One common configuration for embedded systems is the *system on a chip*, an application-specific integrated circuit, for which the CPU was purchased as intellectual property to add to the IC's design.

### Tools

Like a typical computer programmer, embedded system designers use compilers, assemblers and debuggers to develop an embedded system.

Those software tools can come from several sources:

Software companies that specialize in the embedded market Ported from the GNU software development tools.

Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor.

Embedded system designers also use a few software tools rarely used by typical computer programmers.

Some designers keep a utility program to turn data files into code, so that they can include any kind of data in a program.

Most designers also have utility programs to add a checksum or CRC to a program, so it can check its program data before executing it.

### Operating system

They often have no operating system, or a specialized embedded operating system (often a real-time operating system), or the programmer is assigned to port one of these to the new system.

### Debugging

Debugging is usually performed with an in-circuit emulator, or some type of debugger that can interrupt the micro controller’s internal microcode.

The microcode interrupt lets the debugger operate in hardware in which only the CPU works. The CPU-based debugger can be used to test and debug the electronics of the computer from the viewpoint of the CPU. This feature was pioneered on the PDP-11.

Developers should insist on debugging which shows the high-level language, with breakpoints and single stepping, because these features are widely available. Also, developers should write and use simple logging facilities to debug sequences of real-time events.

PC or mainframe programmers first encountering this sort of programming often become confused about design priorities and acceptable methods. Mentoring, code-reviews and ego less programming are recommended.

### Design of embedded systems

The electronics usually uses either a microprocessor or a micro controller. Some large or old systems use general-purpose mainframes computers or minicomputers.

### Start-up

All embedded systems have start-up code. Usually it disables interrupts, sets up the electronics, tests the computer (RAM, CPU and software), and then starts the application code. Many embedded systems recover from short-term power failures by restarting (without recent self-tests). Restart times under a tenth of a second are common.

Many designers have found one of more hardware plus software-controlled LEDs useful to indicate errors during development (and in some instances, after product release, to produce troubleshooting diagnostics). A common scheme is to have the electronics turn off the LED(s) at reset, whereupon the software turns it on at the first opportunity, to prove that the hardware and start-up software have performed their job so far. After that, the software blinks the LED(s) or sets up light patterns during normal operation, to indicate program execution progress and/or errors. This serves to reassure most technicians/engineers and some users.

### The control loop

In this design, the software simply has a loop. The loop calls subroutines. Each subroutine manages a part of the hardware or software. Interrupts generally set flags, or update counters that are read by the rest of the software.

A simple API disables and enables interrupts. Done right, it handles nested calls in nested subroutines, and restores the preceding interrupt state in the outermost enable. This is one of the simplest methods of creating an exokernel.

Typically, there's some sort of subroutine in the loop to manage a list of software timers, using a periodic real time interrupt. When a timer expires, an associated subroutine is run, or flag is set.

Any expected hardware event should be backed-up with a software timer. Hardware events fail about once in a trillion times. That's about once a year with modern hardware. With a million mass-produced devices, leaving out a software timer is a business disaster.

State machines may be implemented with a function-pointer per state-machine (in C++, C or assembly, anyway). A change of state stores a different function into the pointer. The function pointer is executed every time the loop runs.

Many designers recommend reading each IO device once per loop, and storing the result so the logic acts on consistent values.

Many designers prefer to design their state machines to check only one or two things per state. Usually this is a hardware event, and a software timer.

Designers recommend that hierarchical state machines should run the lower-level state machines before the higher, so the higher run with accurate information.

Complex functions like internal combustion controls are often handled with multi-dimensional tables. Instead of complex calculations, the code looks up the values. The software can interpolate between entries, to keep the tables small and cheap.

One major weakness of this system is that it does not guarantee a time to respond to any particular hardware event. Careful coding can easily assure that nothing disables interrupts for long. Thus interrupt code can run at very precise timings. Another major weakness of this system is that it can become complex to add new features. Algorithms that take a long time to run must be carefully broken down so only a little piece gets done each time through the main loop.

This system's strength is its simplicity, and on small pieces of software the loop is usually so fast that nobody cares that it is not predictable.

Another advantage is that this system guarantees that the software will run. There is no mysterious operating system to blame for bad behavior.

###### EMBEDDED SYSTEMS:

An **embedded system** is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few pre-defined tasks, usually with very specific requirements. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass-produced, benefiting from economies of scale.

Personal digital assistants (PDAs) or handheld computers are generally considered embedded devices because of the nature of their hardware design, even though they are more expandable in software terms. This line of definition continues to blur as devices expand.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. In terms of complexity embedded systems can range from very simple with a single microcontroller chip, to very complex with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

## EXAMPLES OF EMBEDDED SYSTEMS

* Automatic teller machines (ATMs)
* Avionics, such as inertial guidance systems, flight control hardware/software and other integrated systems in aircraft and missiles
* Cellular telephones and telephone switches
* Engine controllers and antilock brake controllers for automobiles
* Home automation products, such as thermostats, air conditioners, sprinklers, and security monitoring systems
* Handheld calculators, Handheld computers
* Household appliances, including microwave ovens, washing machines, television sets, DVD players and recorders
* Medical equipment
* Personal digital assistant, Videogame console.

# TECHNICAL SPECIFICATIONS:

**HARDWARE:**

Micro controller : ATMega 328

Crystal : 11.0592 MHz

Bluetooth module : BT HC-05

Temperature sensor : LM35

DC gear motor

Driver IC : L293D

TILT sensor

**Power supply:**

Batteries :Rechargeable sealed lead acid battery 4V,1.0AH

Filter : 1000uf/25V

Voltage Regulator : 7805, 7812

**SOFTWARE:**

* AURDINO

**INTRODUCTION TO BLUETOOTH**

**Description: HC-05** is a class-2**bluetooth module** with Serial Port Profile , which can configure as either Master or slave. a Drop-in replacement for wired serial connections, transparent usage. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project and etc.



**HC-05 Specification:**

* Bluetooth protocol: Bluetooth Specification v2.0+EDR
* Frequency : 2.4GHz ISM band
* Modulation : GFSK(Gaussian Frequency Shift Keying)
* Emission power : ≤4dBm, Class 2
* Sensitivity : ≤-84dBm at 0.1% BER
* Speed:

Asynchronous : 2.1Mbps(Max) / 160 kbps,

Synchronou s : 1Mbps/1Mbps

* Security : Authentication and encryption
* Profiles : Bluetooth serial port
* Power supply : +3.3VDC 50mA
* Working temperature : -20 ~ +75Centigrade
* Dimension : 26.9mm x 13mm x 2.2 mm.
* **POWER SUPPLY**

Any invention of latest technology cannot be activated without the source of power. So in this fast moving world we deliberately need a proper power source which will be apt for a particular requirement. All the electronic components starting from diode to is only work with a DC supply ranging from 5V to 12V.We are utilizing for the same, the cheapest and commonly available energy source of 230V-50Hz and stepping down, rectifying, filtering and regulating the voltage.

100

uf

12v

230V/12

AC step-down transformer

7812

7805

1000uf

1

2

3

1

2

3

330

uf

5v

Bridge

Rectifier

* **Batteries:**
* A bridge rectifier coupled with a step down transformer is used for our design. The voltage rating of transformer used is 0-12V and the current rating is 500mA. When AC voltage of 230V is applied across the primary winding an output AC voltage of 12V is obtained. One alteration of input causes the top of transformer to be positive and the bottom negative. The next alteration will temporarily cause the reverse.
* **Rectifier:**
* In the power supply unit, rectification is normally achieved using a solid state diode. Diode has the property that will let the electron flow easily at one direction at proper biasing condition. Bridge rectifiers of 4 diodes are used to achieve Bridge wave rectification. Two diodes will conduct during the negative cycle and the other two will conduct during the positive half cycle.
* **Filtering unit:**
* Filter circuit which is usually a capacitor acts as a surge arrester always follows the rectifier unit. This capacitor is also called as a decoupling capacitor or a bypass capacitor, is used not only to short the ripple with frequency to ground but also leave the frequency of the DC to appear at the output.
* **Regulators:**
* The voltage regulators play an important role in any power supply unit. The primary purpose of a regulator is to aid the rectifier and filter circuit in providing a constant DC voltage to the device. Power supplies without regulators have an inherent problem of changing DC voltage values due to variations in the load or due to fluctuations in the AC line voltage. With a regulator connected to DC output, the voltage can be maintained within a close tolerant region of the desired output. IC 7805 and 7812 regulators are used in this project for providing a DC voltage of +5V and +12V respectively.

# ARDUINO

**Arduino interface boards** provide the engineers,artists,designers,hobbyists and anyone who tinker with technology with a low-cost,easy-to-use technology to create their creative,interactive objects,useful projects etc.,A whole new breed of projects can now be built that can be controlled from a computer.

**WHAT IS ARDUINO?**



ARDUINO UNO

Arduino is a open source electronics prototyping platform based on flexible, easy-to-use hardware and software. It’s intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. It’s an open-source physical computing platform based on a microcontroller board, and a development environment for writing software for the board.

In simple words, Arduino is a small microcontroller board with a USB plug to connect to your computer and a number of connection sockets that can be wired up to external electronics, such as motors, relays, light sensors, laser diodes, loudspeakers, microphones, etc., They can either be powered through the USB connection from the computer or from a 9V battery. They can be controlled from the computer or programmed by the computer and then disconnected and allowed to work independently.

**Microcontroller**

Microcontroller can be described as a computer embedded on a rather small circuit board. To describe the function of a microcontroller more precisely, it is a single chip that can perform various calculations and tasks, and send/receive signals from other devices via the available pins. Precisely what tasks and communication with the world it does, is what is governed by what instructions we give to the Microcontroller. It is this job of telling the chip what to do, is what we refer to as programming on it.

However, the uC by itself, cannot accomplish much; it needs several external inputs: power, for one; a steady clock signal, for another. Also, the job of programming it has to be accomplished by an external circuit. So typically, a uC is used along with a circuit which provides these things to it; this combination is called a microcontroller board. The Arduino Uno that you have recieved, is one such microcontroller board. The actual microcontroller at its heart is the chip called **Atmega328**. The advantages that Arduino offers over other microcontroller boards are largely in terms of reliability of the circuit hardware as well as the ease of programming and using it.

**Open-source hardware**

Open-source hardware shares much of the principles and approach of free and open-source software. The founders of Arduino wanted people to study their hardware, to understand how it works, make changes to it, and share those changes with the world. To facilitate this, they release all of the original design files (Eagle CAD) for the Arduino hardware. These files are licensed under a Creative Common Attribution Share-Alike license, which allows for both personal and commercial derivative works, as long as they(people) credit Arduino and release their designs under the same license.

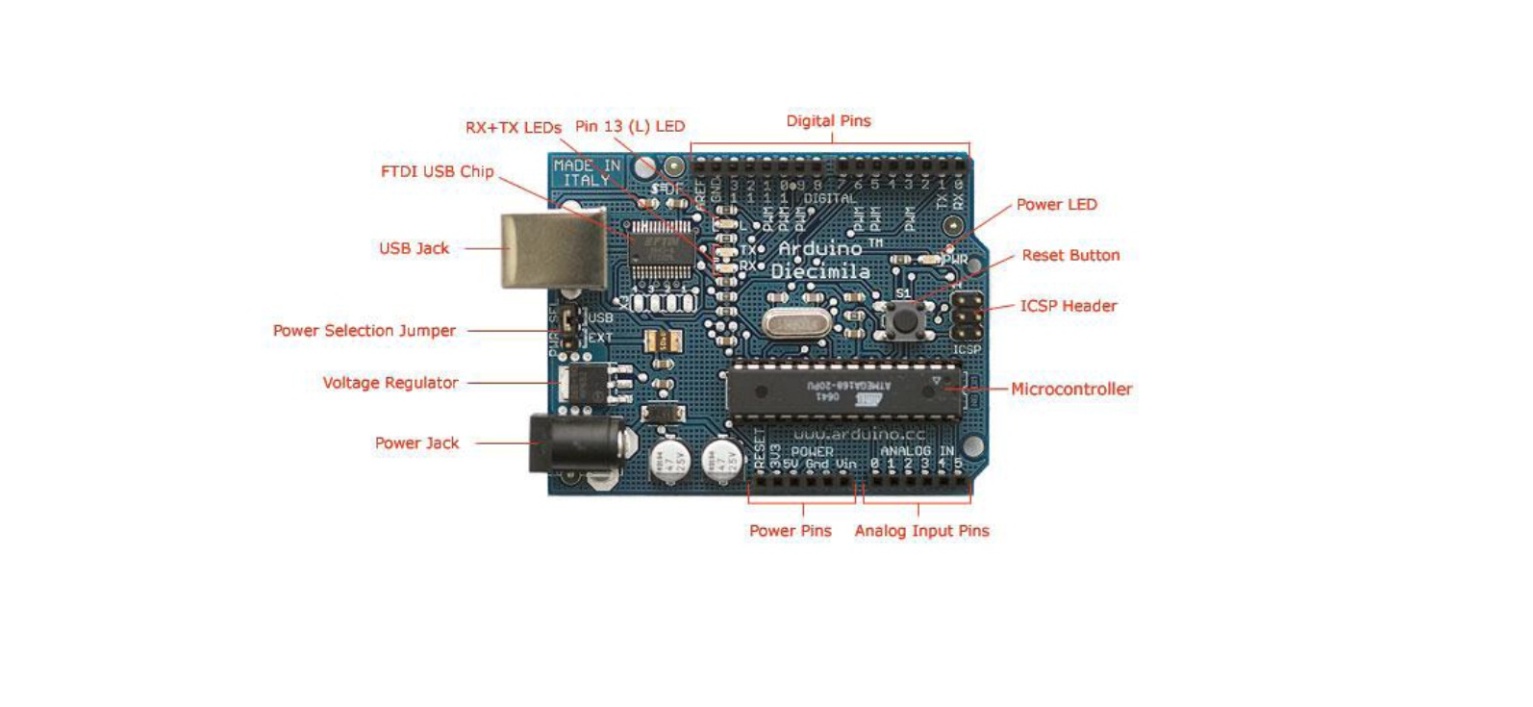
The Arduino software is also open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL

# ARDUINO Board Layout

# board layout 2.jpg

ARDUINO board layout

**ARDUINO pin diagram**

****

ARDUINO pin diagram

ATmega8(Microcontroller)

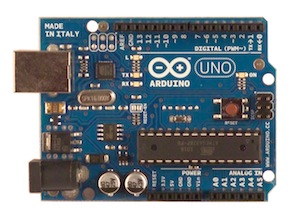
* **16 MHz**
* **8 Kbyte Flash RAM(**1K taken by the boot loader)
* **1 Kbyte RAM(**eg.for auto/local variables and stack)
* **14 digital Input/Output Ports**

**Single chip USB to async. Serial data transfer interface**

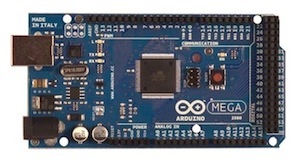
* **USB 2.0 compatible**
* **Transmit and receive LED frive signals**
* **256 Byte receive,128 Byte transmit buffer**
* **Data transfer rate from 300bits/sec to 2 Mb/sec**

**ARDUINO flavors!!**

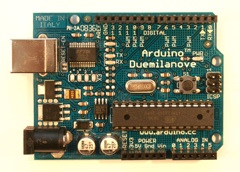
There have been many revisions of the USB Arduino.some of them are

1. **Arduino UNO:**

This is the latest revision of the basic Arduino USB board. It connects to the computer with a standard USB cable and contains everything else you need to program and use the board. It can be extended with a variety of shields: custom daughter-boards with specific features. It is similar to the Duemilanove, but has a different USB-to-serial chip the ATMega8U2, and newly designed labeling to make inputs and outputs easier to identify.

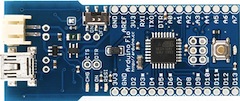
1. **Arduino Mega 2560:**

A larger, more powerful Arduino board. Has extra digital pins, PWM pins, analog inputs, serial ports, etc. The version of the Mega released with the Uno, this version features the Atmega2560, which has twice the memory, and uses the ATMega 8U2 for USB-to-serial communication.

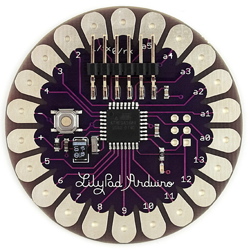
1. **Arduino Duemilanove:**

The Duemilanove automatically selects the appropriate power supply (USB or external power), eliminating the need for the power selection jumper found on previous boards. It also adds an easiest to cut trace for disabling the auto-reset, along with a solder jumper for re-enabling it.

Note: around March 1st, 2009, the Duemilanove started to ship with the ATmega328p instead of theATmega168.

1. **Arduino Fio:**

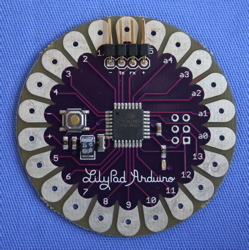
An Arduino intended for use as a wireless node. Has a header for an*XBee*radio, a connector for a*LiPo*battery, and a battery chargingcircuit.

1. LilyPad Arduino:

A stripped-down, circular Arduino board designed for stitching into clothing and other fabric/flexible applications. Needs an additional adapter to communicate with a computer.

1. Arduino Diecimila:

The main change in the Arduino Diecimila is that it can be reset from the computer, without the need to physically press the reset button on the board. The Diecimila uses a low dropout voltage regulator which lowers the board's power consumption when powered by an external supply (AC/DC adapter or battery). A resettable polyfuse protects your computer's USB ports from shorts and surges. It also provides pin headers for the reset line and for 3.3V. There is a built-in LED on pin 13. Some blue Diecimila boards say "Prototype - Limited Edition" but are in fact fully-tested production boards (the actual prototypes are red).

1. Lilypad Arduino 03

This revision has a 6-pin programming header that's compatible with FTDI USB cables and the Sparkfun FTDI Basic Breakout. It adds support for automatic reset, allowing sketches to be uploaded without pressing the reset button on the board. The header is surface mounted, meaning that the board has no pokey bits sticking out the back.

1. **Arduino NG Rev.C**

Revision C of the Arduino NG does not have a built-in LED on pin 13 - instead you'll see two small unused solder pads near the labels "GND" and "13". There is, however, about 1000 ohms of resistance on pin 13, so you can connect an LED without external resistor.

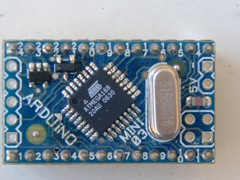


1. **Arduino Extreme**

The Arduino Extreme uses many more surface mount components than previous USB Arduino boards and comes with female pin headers. It also has RX and TX LEDs that indicate when data is being sent to or from the board.

1. **Arduino Mini 04**

On this version of the Arduino Mini, two of the pins changed. The third pin became reset (instead of ground) and fourth pin became ground (instead of being unconnected). These boards are labelled "Mini 04".

Still there are ,Arduino Serial,Arduino Serial v2.0,Arduino Nano 3.0,Arduino Nano 2.x,Serverino(S3V3),Arduino Stamp 02,Mini USB adapter 03,Mini USB Adapter,Arduino Bluetooth.

**Basic Terminologies in ARDUINO:**

**1.Analog to digital converter(ADC)**

The process of Analog to digital conversion is shown in figure.

The Arduino has 10 bits of Resolution when reading analog signals.

2 power 10=1024 increments

Influence also by how fast you sample

**2.Pulse width modulation (PWM)**

The Arduino has 8bit of resolution,when outputting a signal using PWM.The range of output voltage is from 0 to 5 Volts

2power 8=255 Increments

Average of on/off(digital signals to make an average voltage),Duty cycle in 100% of 5Volts.

**LANGUAGE REFERENCES:**

The Microcontroller on the board is programmed using the **Arduino programming language**(based on **wiring**) and the **arduino development environment**(based on **processing**).

**Arduino Programming Language(APL)(based on wiring)**

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

* **Wiring**

Wiring is an open-source programming framework for microcontrollers. Wiring allows writing cross-platform software to control devices attached to a wide range of microcontroller boards to create all kinds of creative coding, interactive objects, spaces or physical experiences. The framework is thoughtfully created with designers and artists in mind to encourage a community where beginners through experts from around the world share ideas, knowledge and their collective experience. There are thousands of students, artists, designers, researchers, and hobbyists who use Wiring for learning, prototyping, and finished professional work production.

**Arduino development environment(based on processing)**

* **Processing**

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production. **3 Software**

The software used by the arduino is Arduino IDE.

he Arduino IDE is a cross-platform application written in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and is derived from the IDE for the [Processing programming language](http://en.wikipedia.org/wiki/Processing_(programming_language)) and the Wiringproject. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as [syntax highlighting](http://en.wikipedia.org/wiki/Syntax_highlighting), [brace matching](http://en.wikipedia.org/wiki/Brace_matching), and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit [makefiles](http://en.wikipedia.org/wiki/Makefiles) or run programs on a[command-line interface](http://en.wikipedia.org/wiki/Command-line_interface). Although building on command-line is possible if required with some third-party tools such as [Ino](http://inotool.org/).

The Arduino IDE comes with a [C](http://en.wikipedia.org/wiki/C_(programming_language))/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:

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

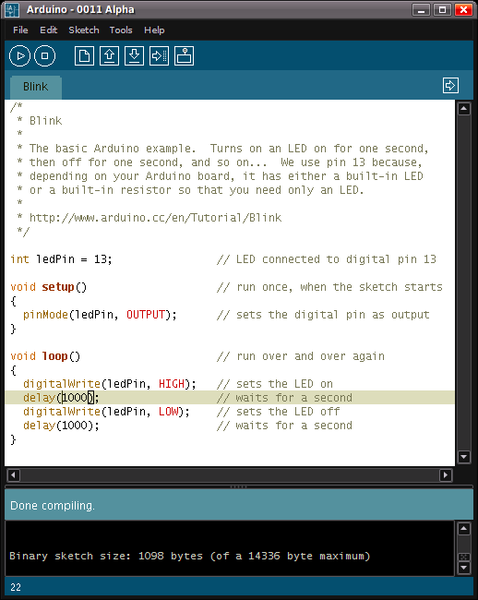


Figure 10 A screenshot of the Arduino IDE showing the "Blink"program,a simple biginner program

A typical first program for a microcontroller simply blinks a [LED](http://en.wikipedia.org/wiki/Light-emitting_diode) on and off. In the Arduino environment, the user might write a program like this:

#define LED\_PIN 13

void setup () {

pinMode (LED\_PIN, OUTPUT); *// enable pin 13 for digital output*

}

void loop () {

digitalWrite (LED\_PIN, HIGH); *// turn on the LED*

delay (1000); *// wait one second (1000 milliseconds)*

digitalWrite (LED\_PIN, LOW); *// turn off the LED*

delay (1000); *// wait one second*

}

For the above code to work correctly, the positive side of the LED must be connected to pin 13 and the negative side of the LED must be connected to ground. The above code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

The Arduino IDE uses the [GNU toolchain](http://en.wikipedia.org/wiki/GNU_toolchain) and [AVR Libc](http://en.wikipedia.org/w/index.php?title=AVR_Libc&action=edit&redlink=1) to compile programs, and uses [avrdude](http://en.wikipedia.org/w/index.php?title=Avrdude&action=edit&redlink=1) to upload programs to the board.

For educational purposes there is third party graphical development environment called [Minibloq](http://en.wikipedia.org/wiki/Minibloq) available under a different open source license.

**language reference**

Arduino programs can be divided in three main parts: structure, values (variables and constants), and functions.

Available datatypes in ARDUINO IDE are

•void

•boolean

•char **( 0 – 255)**

•byte - **8 bit data ( 0 – 255)**

•int - **16-bit data (32,767 - -32,768)**

•long – **32 bit data (2,147,483,647 to -2,147,483,648**)

•float

•double

•string - char array

•String - object

•array

**Arithmetic operators**

Arithmetic operators include addition,subtraction,multiplication and division.For math that requires fractions,you can use float variables,if you can bear large size and slow computation speeds in your microcontroller.

**e.g. ,**

y = y + 3;

x = x – 7;

i = j \* 6;

r = r / 5;

**Comparision operators**

Comparisons of one variable or constant against another are often used in if statements to test if a specified condition is true.

**e.g. ,**

x == y // x is equal to y

x != y // x is not equal to y

x < y // x is less than y

x > y // x is greater than y

x <= y // x is less than or equal to y

x >= y // x is greater than or equal to y

**Logical operators**

Logical operators are usually a way to logically combine two expressions and return a **TRUE** or **FALSE** depending on the operator.

There are three logical operators, **AND, OR, and NOT.**

**e.g. ,**

Logical AND:

if (x > 0 && x < 5) // true only if both expressions are true

Logical OR:

if (x > 0 || y > 0) // true if either expression is true

Logical NOT:

if (!x > 0) // true only if expression

**TRUE/FALSE**

These are Boolean constants that define logic levels of the arduino.

**FALSE** is easily defined as 0 (zero)

**TRUE** is often defined as 1, but can also be anything else except zero. So in a Boolean sense, -1, 2, and -200 are all also defined as TRUE.

**e.g. ,**

if (abcd== TRUE);

{

DoSomethingNice;

}

else

{

DoSomethingHorrible;

}

**HIGH/LOW**

These constants define pin levels as HIGH or LOW and are used when reading or writing to digital pins.

HIGH is defined as logic level 1, ON, or 5 volts

LOW is logic level 0, OFF, or 0 volts.

**e.g. ,**

digitalWrite(13, HIGH);

**INPUT/OUTPUT**

These constants define pin levels as HIGH or LOW and are used when reading or writing to digital pins.

HIGH is defined as logic level 1, ON, or 5 volts

LOW is logic level 0, OFF, or 0 volts.

**e.g. ,**

pinmode(13, OUTPUT);

**Arduino/processing language Comparision**

The Arduino language (based on Wiring) is implemented in C/C++, and therefore has some differences from the Processing language, which is based on Java.

Figure Comparision between Arduino and Processing language

**SIMULATOR for ARDUINO:**

The Arduino Simulator app gives the user the freedom to work without the basic setup of hardware and software. It is designed to be used by beginners and also, experienced developers, who want to quickly develop Arduino projects.

The developer can make the necessary changes in the code - delay, pin number, and state - 0 (low) 1 (high) - and check it immediately. The app shows the breadboard, complete with 14 LED pins.

You can drag and place the wires in the correct positions to connect to Arduino. If the wires are placed according to the code, then it will show the expected results.  Once satisfied, you can save it and email it. The code can be copied and used in an actual project just as easily.

This app is an easy way to work through Arduino projects. With customisable codes, and a simple to use interface,  this Arduino Simulator app from Schogini Systems  is a convenient app for Arduino developers.

**A** screenshot of Arduino simulator is shown in the figure below.

**APPLICATIONS OF ARDUINO**

Arduino was basically designs to make the process of using electronics in multidisciplinary projects more accessible. It is intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. because of these features, arduino finds extensive application in various fields. Arduino projects can be stand-alone or they can communicate with software running on a computer.

**Why Arduino is popular?**

Here are five reasons why the Arduino is more popular than beagleboard:

* **Starter Projects:** Editing and rewriting is often easier than writing from scratch. It’s the same with electronics. It’s easier to mod an idea than start with a blank slate.That’s where the BeagleBoard falls short. “It has virtually no example application that you can just copy and hack to learn from,” says Massimo Banzi, one of the co-founders of the Arduino project. It’s a chicken-and-egg problem for the BeagleBoard. Unless there are more example codes out there, it is difficult to draw in the audience. And without the audience it is challenging to get enough sample projects into the community.
* **Cost and Durability:** At $30 a piece, an Arduino is an inexpensive investment for someone who wants to try it out. “It’s the price of a few sandwiches,” says Torrone. Compare that to the BeagleBoard-xM, which costs $180. One reason why the Arduino is so cheap is because it is easy to clone. The microcontroller is completely open source so the “components are all commodity,” says Torrone. With the BeagleBoard, hobbyists don’t have the same amount of freedom. They have to work closely with Texas Instruments or its partners, says Torrone. Arduino is also very resilient. Drop it, smash it and it still stays alive. Add to that its low-power requirement, and the product becomes a must-have for DIYers. An Arduino can run on a 9V-battery for days. “The BeagleBoard is fast and powerful but that also means lots of energy is needed, which makes it difficult for simple projects,” says Torrone.
* **A Thriving Community:** Arduino’s popularity means it’s easy to get started. Companies such as Adafruit, SparkFun and Liquidware not only sell chips, but they also host blogs that suggest ideas on how to use your Arduino while providing extensive project plans to guide you in completing your creations. Will Chellman, a student who has played with Arduino for years, says he’s now experimenting with the BeagleBoard. But finding documentation and information to work off is not easy, he says. The lack of well-documented projects done with the BeagleBoard can be intimidating to new users as well, says Banzi. “There’s lots of of interesting stuff (about the BeagleBoard) but it is very technical,” he [wrote in a comment](http://www.wired.com/gadgetlab/2010/06/beagleboard/) recently on Gadget Lab in response to the launch of BeagleBoard-xM. Banzi says BeagleBoard documentation is also scattered and fragmented. “Parts of it have aged and you spend quite a bit of time jumping from wikis to mailing list to track which specific bit of documentation applies to your board, bootloader etc.,” he says.
* **Maturity is the key:** Arduino has had a head start on the BeagleBoard. By October 2008, about 50,000 Arduino boards had already been shipped. That year, the first BeagleBoards started making their way into the hands of hardware enthusiasts. The BeagleBoard is just two years old. Since it hasn’t been around long enough, there’s not enough people building apps based on it,” says Chellman. That’s not to say that BeagleBoard isn’t catching up. Earlier this month, we showed five projects ranging from a [videowall to the iPad of ham radios](http://www.wired.com/gadgetlab/2010/06/beagleboard/) that use the BeagleBoard. There’s also a build-your-own tablet kit that is based off the BeagleBoard.
* **Simple is attractive**: With its single-board computer configuration, 1-GHz processing power and the choice of accessories, the BeagleBoard is a creative engineer’s dream come true. But the same reasons make it intimidating to those who want to geek out on a DIY project but don’t have the technical know-how. Arduino users point out that it is simple to connect external sensors to the board, and the example codes out there make it easy to get started quickly. Arduino is a simple system designed for creative people with little or “no prior knowledge of electronics,” says Banzi. “It’s cheap and open source with lots of documentation written in a not too technical language. Above all, it has a very welcoming attitude towards beginners and tries not to scare them too much.”

**3.2 Conclusion and future scope**

Over the years,Arduino has went out to become a huge success and a common name among students.With google deploying it,people’s imagination has went out to much higher level than before.A developer in the annual GOOGLE IO conference said “when Arduino and Android coming together,this really proves “INFINITY EXISTS” in the future”.I think a study on arduino and practical experiments on arduino must be added for UG courses of engineering,to help students to leverage their talents,and imagination.

**L293D HALF H-DRIVERS**

6.1 DESCRIPTION:

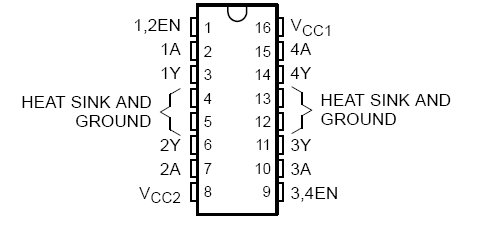
The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V.

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. In this project we provide two drivers, one for two motors which move the robot forward and backward and an other for a motor which controls the arm of the robot.

6.2 Features of L293 and L293D

* Wide Supply-Voltage Range: 4.5 V to 36 V
* Separate Input-Logic Supply
* Internal ESD Protection
* Thermal Shutdown
* High-Noise-Immunity Inputs
* Functional Replacements for SGS L293 and SGS L293D
* Output Current 1 A Per Channel (600 mA for L293D)
* Peak Output Current 2 A Per Channel (1.2 A for L293D)
* Output Clamp Diodes for Inductive Transient Suppression (L293D)

6.3 PIN DIAGRAM

****

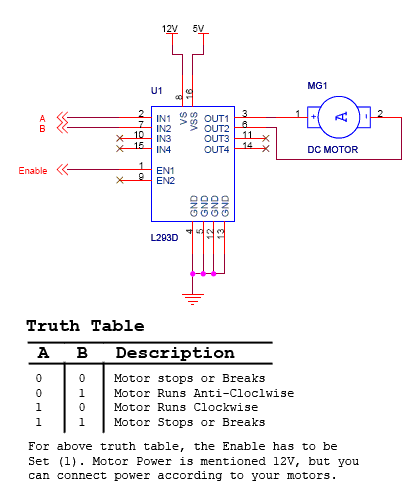
FIGUR L293D Dual H-Bridge Motor Driver

L293D is a dual H-Bridge motor driver, so with one IC we can interface two DC motors which can be controlled in both clockwise and counter clockwise direction and if you have motor with fix direction of motion you can make use of all the four I/Os to connect up to four DC motors.

L293D has output current of 600mA and peak output current of 1.2A per channel. Moreover for protection of circuit from back EMF output diodes are included within the IC.

The output supply (VCC2) has a wide range from 4.5V to 36V, which has made L293D a best choice for DC motor driver.

A simple schematic for interfacing a DC motor using L293D is shown below.



As you can see in the circuit, three pins are needed for interfacing a DC motor (A, B, Enable). If you want the o/p to be enabled completely then you can connect Enable to VCC and only 2 pins needed from controller to make the motor work.

As per the truth mentioned in the image above its fairly simple to program the microcontroller. Its also clear from the truth table of BJT circuit and L293D the programming will be same for both of them, just keeping in mind the allowed combinations of A and B. We will discuss about programming in C as well as assembly for running motor with the help of a microcontroller.

* It is a 16 pin DIP IC
* It is a dual H- Bridge motor driver.
* L293d able to drive with output current of 600mA for each channel
* It consists of four inputs, which accepts TTL logic voltage level and outputs that gives Vcc2 (8th pin) voltage.
* 16 pin is connected to +5V, operating voltage of IC
* 8 pin is operating voltage of Motor
* There are two more TTL inputs (EN), which stands for enable. This means that pin1 (1,2EN) enables outputs 1Y & 2Y.
* Motor Driver IC for 2 DC or 1 Stepper Motor. Gives 1A current per channel. Same pin configuration as per L293D. Available in DIP16 package.
* L293NE Dual Full Bridge Driver, for 2 DC motors or a Stepper motor in bipolar mode.

**Features:**

* Wide Supply-Voltage Range: 4.5 V to 36 V
* Separate Input-Logic Supply
* Pin compatible with L293D
* Internal ESD Protection
* Thermal Shutdown
* High-Noise-Immunity Inputs
* Output Current 1 A Per Channel
* Peak Output Current 2 A Per Channel
* Output Clamp Diodes for Inductive Transient Suppression

**Description:**

The L293 and L293D are quadruple high-current half-H drivers. The L293NE is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. This device is 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.

DC MOTOR

INTRODUCTION TO MOTORS

Electric motors are extremely important to modern-day life, being used in many different places, e.g., vacuum cleaners, dishwashers, computer printers, fax machines, video cassette recorders, machine tools, printing presses, automobiles, subway systems, sewage treatment plants and water pumping stations. Electric motor is a machine used to convert electrical energy to mechanical energy. In this project we use three DC motors, two which control the direction of the robot and an other which controls the arm of the robot.

The major physical principles behind the operation of an electric motor are known as Ampere’s law and Faraday's law. The first states that an electrical conductor sitting in a magnetic field will experience a force if any current flowing through the conductor has a component at right angles to that field. Reversal of either the current or the magnetic field will produce force acting in the opposite direction. The second principle states that if a conductor is moved through a magnetic field, then any component of motion perpendicular to that field will generate a potential difference between the ends of the conductor.

An electric motor consists of two essential elements. The first, a static component which consists of magnetic materials and electrical conductors to generate magnetic fields of a desired shape, is known as the stator. The second, which also is made from magnetic and electrical conductors to generate shaped magnetic fields which interact with the fields generated by the stator, is known as the rotor. The rotor comprises the moving component of the motor, having a rotating shaft to connect to the machine being driven and some means of maintaining an electrical contact between the rotor and the motor housing (typically, carbon brushes pushed against slip rings). In operation, the electrical current supplied to the motor is used to generate magnetic fields in both the rotor and the stator. These fields push against each other with the result that the rotor experiences a torque and consequently rotates.Electrical motors fall into two broad categories, depending on the type of electrical power applied-direct current (DC) and alternating current (AC) motors.

The first DC electrical motor was demonstrated by Michael Faraday in England in 1821. Since the only available electrical sources were DC, the first commercially available motors were of the DC type, becoming popular in the 1880s. These motors were used for both low power and high power applications, such as electric street railways. It was not until the 1890s, with the availability of AC electrical power that the AC motor was developed, primarily by the Westinghouse and General Electric corporations. Throughout this decade, most of the problems concerned with single and multi-phase AC motors were solved. Consequently, the principal features of electric motors were all developed by 1900.

MOTOR CHARACTERSTICS

There are a few characteristics common to all motors:

**Voltage**

The rated voltage of a motor is the voltage at which it operates at peak efficiency. Most DC motors can be operated somewhat above or below their range, but it's best to plan to operate them at their rated voltage. Dropping below rated voltage reduces the motor's power, and operating above the rated voltage may burn the motor out. It is better if the motor's top speed is at rated voltage, and slowest speed is at no more than 50% less than the rated voltage.

**Current**

Motors draw current depending on the load they're pulling. Usually more load means more current. Every motor has a stall current, which is the current it draws when it's stopped by an opposing force. This stall current is much greater than the runningcurrent, or current that it draws under no load. Your power supply for a motor should be able to handle the stall current with extra amperage to spare. Motors may draw near the stall current for a brief period of time when starting up, to overcome their inertia.

**Speed**

Motor speed is given in revolutions per minute (RPM's).

**Torque**

Torque is the measure of a motor's pulling force. It's measured by the force a motor can pull when the opposing force is attached to a shaft attached to its center rod. If the shaft sticks out a foot from the motor's center, and the motor can pull one pound on that shaft, the motor's torque is one foot-pound. Motor manufacturers haven't standardized this measurement, so sometimes you will see it as ft.-lb., lb-ft., oz.-in, in.-oz., g-cm (gram-centimeter), and any other weight to length variation you can think of.

**Resistance:**

Often you'll see a motor rated in ohms. This just gives you the resistance that the motor's coil offers. Using Ohm's Law (voltage = current x resistance), you can calculate the motor's current draw if you know the rated voltage and the coil resistance.

TYPES OF MOTORS:

Industrial motors come in a variety of basic types. These variations are suitable for many different applications. Naturally, some types of motors are more suited for certain applications than other motor types are:

 AC Motors

 DC Motors

 Brushless DC Motors

 Servo Motors

 Brushed DC Servo Motors

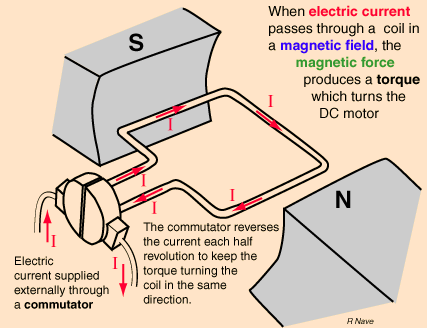
 Brushless AC Servo Motors

 Stepper Motors

 Linear Motors

DC MOTOR

The operation of a DC motor is dependent on the workings of the poles of the stator with a part of the rotor, or armature. The stator contains an even number of poles of alternating magnetic polarity, each pole consisting of an electromagnet formed from a pole winding wrapped around a pole core. When a DC current flows through the winding, a magnetic field is formed. At its center is the rotor, a coil wound around an iron armature, which spins within the poles of the magnet that can be seen on the inside of the casing. It also contains a winding, in which the current flows in the direction illustrated in figure below. This armature current interacts with the magnetic field in accordance with Ampere’s law, producing a torque which turns the armature.



If the armature windings were to rotate round to the next pole piece of opposite polarity, the torque would operate in the opposite direction, thus stopping the armature. In order to prevent this, the rotor contains a commutator which changes the direction of the armature current for each pole piece that the armature rotates past, thus ensuring that the windings passing, for example, a pole of north polarity will all have current flowing in the same direction, while the windings passing south poles will have oppositely flowing current to produce a torque in the same direction as that produced by the north poles. The commutator generally consists of a split contact ring against which the brushes applying the DC current ride.

The rotation of the armature windings through the stator field generates a voltage across the armature which is known as the counter EMF (electromotive force) since it opposes the applied voltage: this is the consequence of Faraday's law. The magnitude of the counter EMF is dependent on the magnetic field strength and the speed of the rotation of the armature. When the DC motor is initially turned on, there is no counter EMF and the armature starts to rotate. The counter EMF increases with the rotation. The effective voltage across the armature windings is the applied voltage minus the counter EMF.

TYPES OF DC MOTOR

[DC motors](http://science.jrank.org/pages/2312/Electric-Motor-Types-DC-motor.html) are more common than we may think. A [car](http://science.jrank.org/pages/2312/Electric-Motor-Types-DC-motor.html) may have as many as 20 DC motors to drive fans, seats, and windows. They come in three different types, classified according to the electrical circuit used. In the shunt motor, the armature and field windings are connected in parallel, and so the currents through each are relatively independent. The current through the field winding can be controlled with a field rheostat (variable resistor), thus allowing a wide variation in the motor speed over a large range of load conditions. This type of motor is used for driving machine tools or fans, which require a wide range of speeds.

In the series motor, the field winding is connected in series with the armature winding, resulting in a very high starting torque since both the armature current and field strength run at their maximum. However, once the armature starts to rotate, the counter EMF reduces the current in the circuit, thus reducing the field strength. The series motor is used where a large starting torque is required, such as in automobile [starter motors](http://science.jrank.org/pages/2312/Electric-Motor-Types-DC-motor.html), cranes, and hoists.

The compound motor is a combination of the series and shunt motors, having parallel and series field windings. This type of motor has a high starting torque and the ability to vary the speed and is used in situations requiring both these properties such as punch presses, conveyors and elevators.

## *Brushed*

The brushed DC motor generates torque directly from DC power supplied to the motor by using internal commutation, stationary permanent magnets, and rotating electrical magnets. Advantages of a brushed DC motor include low initial cost, high reliability, and simple control of motor speed. Disadvantages are high maintenance and low life-span for high intensity uses. Maintenance involves regularly replacing the brushes and springs which carry the electric current, as well as cleaning or replacing the commutator. These components are necessary for transferring electrical power from outside the motor to the spinning wire windings of the rotor inside the motor.

## *Synchronous*

Synchronous DC motors, such as the brushless DC motor and the stepper motor, require external commutation to generate torque. They lock up if driven directly by DC power.

## *Brushless*

Brushless DC motors use a rotating permanent magnet in the rotor, and stationary electrical magnets on the motor housing. A motor controller converts DC to AC. This design is simpler than that of brushed motors because it eliminates the complication of transferring power from outside the motor to the spinning rotor. Advantages of brushless motors include long life span, little or no maintenance, and high efficiency. Disadvantages include high initial cost, and more complicated motor speed controllers.

SELECTION OF DC MOTOR:

Choosing DC motor and associated equipment for a given application requires consideration of several factors.

**Speed range:** If field control is to be used, and a large speed range is required, the base speed must be proportionately lower and the motor size must be larger. If speed range is much over 3:1, armature voltage control should be considered for at least part of the range. Very wide dynamic speed range can be obtained with armature voltage control. However, below about 60% of base speed, the motor should be derated or used for only short periods.

**Speed variation with torque:** Applications requiring constant speed at all torque demands should use a shunt-wound dc motor. If speed change with load must be minimized, a dc motor regulator, such as one employing feedback from a tachometer, must be used.

When the dc motor speed must decrease as the load increases, compound or series-wound dc motors may be used. Or, a dc motor power supply with a drooping volt-ampere curve could be used with a shunt-wound dc motor.

**Reversing:** This operation affects power supply and control, and may affect the dc motor's brush adjustment, if the dc motor cannot be stopped for switching before reverse operation. In this case, compound and stabilizing dc motor windings should not be used, and a suitable armature-voltage control system should supply power to the dc motor.

**Duty cycle:** Direct current motors are seldom used on drives that run continuously at one speed and load. Motor size needed may be determined by either the peak torque requirement or heating.

**Peak torque*:*** The peak torque that a dc motor delivers is limited by that load at which damaging commutation begins. Dc motor brush and commutator damage depends on sparking severity and duration. Therefore, the dc motor's peak torque depends on the duration and frequency of occurrence of the overload. Dc motor peak torque is often limited by the maximum current that the power supply can deliver.

Dc motors can commutate greater loads at low speed without damage. NEMA standards specify that machines powered by dc motors must deliver at least 150% rated current for 1 min at any speed within rated range, but most dc motors do much better.

**Heating:** Dc motor temperature is a function of ventilation and electrical/mechanical losses in the machine. Some dc motors feature losses, such as core, shunt-field, and brush-friction losses, which are independent of load, but vary with speed and excitation.

The best method to predict a given dc motor's operating temperature is to use thermal capability curves available from the dc motor manufacturer. If curves are not available, dc motor temperature can be estimated by the power-loss method. This method requires total losses versus load curve or an efficiency curve.

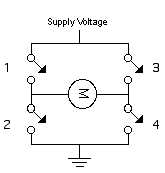
For each portion of the duty cycle, power loss is obtained and multiplied by the duration of that portion of the cycle. The summation of these products divided by the total cycle time gives the dc motor's average power loss. The ratio of this value to the power loss at the motor rating is multiplied by the dc motor's rated temperature rise to give the approximate temperature rise of the dc motor when operated on that duty cycle.

*CONTROLLING DC MOTORS*

There are two easily controllable parameters of a DC motor, direction and speed. To control the direction, the polarity of the motor is reversed. To control the speed, the input voltage is varied using pulse width modulation.

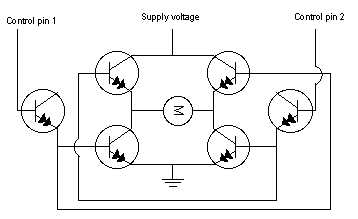
**Direction Control**

To control a DC motor from a microcontroller, you use switching arrangement known as an H bridge. It looks like this:



When switches 1 and 4 are closed and 2 and 3 are open, voltage flows from the supply to 1 to the motor to 4 to ground. When 2 and 3 are closed and 1 and 4 are open, polarity is reversed, and voltage flows from the supply to 3 to the motor to 2 to ground.

An H-bridge can be built from transistors, so that a microcontroller can switch the motor, like this:



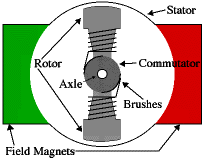
You can see that there are six transistors here; the outer two are used to switch the inner four, in pairs, so that the proper two transistors always switch together. If you were using this circuit, you'd want to make sure that control pins 1 and 2 were always reversed; when one is high, the other is low.

Although you can make your own H-bridges, it's usually easier to use a controller manufactured specifically for the job. A pre-manufactured H-bridge chip will include diodes to protect the transistors from back voltage, sometimes a current sensing pin to sense the current the motor is drawing, and much more. There are many motor drivers available from various electronics suppliers. Look around to find one that suits your needs and price range.

**5.4 DC Motors:-**

Industrial applications use dc motors because the speed-torque relationship can be varied to almost any useful form -- for both dc motor and regeneration applications in either direction of rotation. Continuous operation of dc motors is commonly available over a speed range of 8:1. Infinite range (smooth control down to zero speed) for short durations or reduced load is also common. Dc motors are often applied where they momentarily deliver three or more times their rated torque. In emergency situations, dc motors can supply over five times rated torque without stalling (power supply permitting). Dynamic braking (dc motor-generated energy is fed to a resistor grid) or regenerative braking (dc motor-generated energy is fed back into the dc motor supply) can be obtained with dc motors on applications requiring quick stops, thus eliminating the need for, or reducing the size of, a mechanical brake.

Dc motors feature a speed, which can be controlled smoothly down to zero, immediately followed by acceleration in the opposite direction -- without power circuit switching. And dc motors respond quickly to changes in control signals due to the dc motor's high ratio of torque to inertia. In any electric motor, operation is based on simple electromagnetism. A [current](http://encyclobeamia.solarbotics.net/articles/current.html)-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](http://encyclobeamia.solarbotics.net/articles/current.html) 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](http://encyclobeamia.solarbotics.net/articles/dc.html) motor is designed to harness the magnetic interaction between a [current](http://encyclobeamia.solarbotics.net/articles/current.html)-carrying conductor and an external magnetic field to generate rotational motion. Let's start by looking at a simple 2-pole [DC](http://encyclobeamia.solarbotics.net/articles/dc.html) electric motor (here red represents a magnet or winding with a "North" polarization, while green represents a magnet or winding with a "South" polarization).



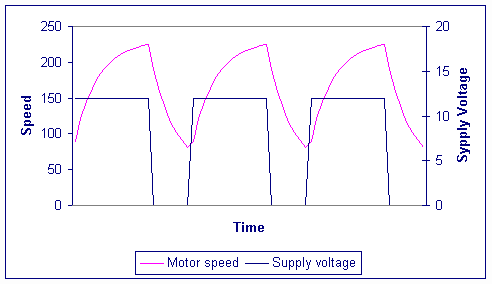
Every [DC](http://encyclobeamia.solarbotics.net/articles/dc.html) motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that [BEAM](http://encyclobeamia.solarbotics.net/articles/beam.html)ers will see), the external magnetic field is produced by high-strength permanent magnets1. 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 rotors (together with the axle and attached commutator) rotate 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.

**5.5Theory of DC motor speed control**:

The speed controller works by varying the average voltage sent to the motor. It could do this by simply adjusting the voltage sent to the motor, but this is quite inefficient to do. A better way is to switch the motor's supply on and off very quickly. If the switching is fast enough, the motor doesn't notice it, it only notices the average effect. When you watch a film in the cinema, or the television, what you are actually seeing is a series of fixed pictures, which change rapidly enough that your eyes just see the average effect - movement. Your brain fills in the gaps to give an average effect. Now imagine a light bulb with a switch. When you close the switch, the bulb goes on and is at full brightness, say 100 Watts. When you open the switch it goes off (0 Watts). Now if you close the switch for a fraction of a second, then open it for the same amount of time, the filament won't have time to cool down and heat up, and you will just get an average glow of 50 Watts. This is how lamp dimmers work, and the same principle is used by speed controllers to drive a motor. When the switch is closed, the motor sees 12 Volts, and when it is open it sees 0 Volts. If the switch is open for the same amount of time as it is closed, the motor will see an average of 6 Volts, and will run more slowly accordingly. As the amount of time that the voltage is on increases compared with the amount of time that it is off, the average speed of the motor increases.

This on-off switching is performed by power MOSFETs. A MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) is a device that can turn very large currents on and off under the control of a low signal level voltage. For more detailed information, see the dedicated chapter on [MOSFETs](http://homepages.which.net/~paul.hills/SpeedControl/Mosfets.html)) .The time that it takes a motor to speed up and slow down under switching conditions is dependent on the inertia of the rotor (basically how heavy it is), and how much friction and load torque there is. The graph below shows the speed of a motor that is being turned on and off fairly slowly:

**5.6 2pin switch**:

****

These small, two-pin, tactile buttons are intended for mounting to PCBs, but they can also be plugged into standard 0.1" breadboards. They work well as reset buttons and can be connected to microcontroller inputs to serve as user-interface components.

## Overview

These small, two-pin, SPST momentary pushbuttons are intended for mounting to PCBs, but they can also be plugged into standard [0.1" breadboards](http://www.pololu.com/catalog/category/28) as shown in the picture to the right. We use them as reset buttons and user pushbuttons in several of our products, including the [3pi robot](http://www.pololu.com/catalog/product/975) and most of our [Orangutan robot controllers](http://www.pololu.com/catalog/category/8). Note that this button should not be used with voltages above 12 V, and it should not be used to switch currents greater than 50 mA. This product is sold as a **5-pack**.

## Specs

* Activation force: 6 oz
* Maximum rating: DC 12 V / 50 mA
* On resistance: ≤ 50 mΩ
* Off resistance: > 100 MΩ
* Life: > 100,000 cycles

|  |
| --- |
|  |
|  |

**INTRODUCTION TO SERIAL COMMUNICATION**

In [telecommunication](http://en.wikipedia.org/wiki/Telecommunication) and [computer science](http://en.wikipedia.org/wiki/Computer_science), the concept of serial communication is the process of sending [data](http://en.wikipedia.org/wiki/Data) one [bit](http://en.wikipedia.org/wiki/Bit) at a time, sequentially, over a [communication channel](http://en.wikipedia.org/wiki/Communication_channel) or [computer bus](http://en.wikipedia.org/wiki/Computer_bus). This is in contrast to [parallel communication](http://en.wikipedia.org/wiki/Parallel_communication), where several bits are sent as a whole, on a link with several parallel channels. Serial communication is used for all long-haul communication and most [computer networks](http://en.wikipedia.org/wiki/Computer_network), where the cost of [cable](http://en.wikipedia.org/wiki/Cable) and [synchronization](http://en.wikipedia.org/wiki/Synchronization) difficulties make parallel communication impractical. Serial computer buses are becoming more common even at shorter distances, as improved [signal integrity](http://en.wikipedia.org/wiki/Signal_integrity) and transmission speeds in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity (no need for serializer and deserializer, or [SerDes](http://en.wikipedia.org/wiki/SerDes)) and to outstrip its disadvantages ([clock skew](http://en.wikipedia.org/wiki/Clock_skew), interconnect density).

## Serial buses:

[Integrated circuits](http://en.wikipedia.org/wiki/Integrated_circuit) are more expensive when they have more pins. To reduce the number of pins in a package, many ICs use a serial bus to transfer data when speed is not important. Some examples of such low-cost serial buses include [SPI](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus), [I²C](http://en.wikipedia.org/wiki/I%C2%B2C), [UNI/O](http://en.wikipedia.org/wiki/UNI/O), and [1-Wire](http://en.wikipedia.org/wiki/1-Wire).

# Asynchronous serial communication

Asynchronous serial communication describes an [asynchronous](http://en.wikipedia.org/wiki/Asynchronous_communication), [serial transmission protocol](http://en.wikipedia.org/wiki/Serial_communication) in which a start signal is sent prior to each byte, character or code word and a stop signal is sent after each code word. The start signal serves to prepare the receiving mechanism for the reception and registration of a symbol and the [stop signal](http://en.wikipedia.org/wiki/Stop_signal) serves to bring the receiving mechanism to rest in preparation for the reception of the next symbol. A common kind of start-stop transmission is [ASCII](http://en.wikipedia.org/wiki/ASCII) over [RS-232](http://en.wikipedia.org/wiki/RS-232), for example for use in [teletypewriter](http://en.wikipedia.org/wiki/Teletypewriter) operation.

[Puerto serie Rs232.png](http://en.wikipedia.org/wiki/File:Puerto_serie_Rs232.png)

In the diagram, two [bytes](http://en.wikipedia.org/wiki/Byte) are sent, each consisting of a start bit, followed by seven data bits (bits 0-6), a [parity bit](http://en.wikipedia.org/wiki/Parity_bit) (bit 7), and one stop bit, for a 10-bit character frame. The number of data and formatting bits, the [order of data bits](http://en.wikipedia.org/wiki/Endianness), and the transmission speed must be pre-agreed by the communicating parties. The "stop bit" is actually a "stop period"; the stop period of the transmitter may be arbitrarily long. It cannot be shorter than a specified amount, usually 1 to 2 bit times. The receiver requires a shorter stop period than the transmitter. At the end of each character, the receiver stops briefly to wait for the next start bit. It is this difference which keeps the transmitter and receiver synchronized.

**MAX 232 PIN DIAGRAM:**

****

Max232 Pin

**SPECIFICATIONS:**

* Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
* Operates from a Single 5-V Power Supply With 1.0-\_F Charge-Pump Capacitors
* Operates up To 120 Kbit/s
* Two Drivers and Two Receivers
* 30-V Input Levels
* Low Supply Current 8 mA Typical
* ESD Protection Exceeds JESD 22
* 2000-V Human-Body Model (A114-A)
* Upgrade with Improved ESD (15-kV HBM) and 0.1-\_F Charge-Pump Capacitor

**DESCRIPTION:**

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F Voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept 30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells.

**LOGIC DIAGRAM:**



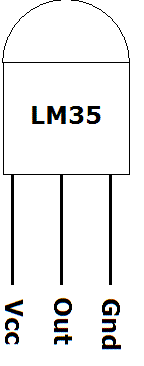
**DB-9 Connector:**

The RS-232c standard is based on a low/false voltage between +3 to +15V, and an high/true voltage between -3 to -15V (+/-12V is commonly used). Figure 27.4 shows some of the common connection schemes. In all methods the txd and rxd lines are crossed so that the sending txd outputs are into the listening rxd inputs when communicating between computers. When communicating with a communication device (modem), these lines are not crossed. In the modem connection the dsr and dtr lines are used to control the flow of data. In the computer the cts and rts lines are connected. These lines are all used for handshaking, to control the flow of data from sender to receiver. The null-modem configuration simplifies the handshaking between computers. The three wire configuration is a crude way to connect to devices, and data can be lost.

Circuit diagram:



**LM35 Temperature sensor:**

* Transducers convert physical data such as temp, light intensity, speed etc to electrical signal.
* Depending on the transducer, the output produced in the form of voltage, current, resistance or capacitance.
* For example: Temperature is converted in to electrical signals using a transducer called Thermister. A thermister responds to the temperature change by changing resistance, but its response is not linear.
* That’s why we are using linear temperature sensors like LM34 or LM35 series.
* The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature.
* Its output is 10MilliVolts per degree centigrade.
* So if the output is 310 mV then temperature is 31 degree C.
* It has a range of −55 to +150°C temperature range. 
* LM35 is a popular and low cost temperature sensor.
* It has three pins as follows.
* To use the sensor simply connect the Vcc to 5V, GND to Gnd and the Out to one of the ADC (analog to digital converter channel). The output linearly varies with temperature.

PROGRAM CODE:

int val;

float TEMP;

int M1 = 4;

int M2 = 5;

int M3 = 6;

int M4 = 7;

int sw = 8;

int temperature, humidity, temp,Temp;

int state;

int flag=0;

void setup()

{

pinMode(M1, OUTPUT);

pinMode(M2, OUTPUT);

pinMode(M3, OUTPUT);

pinMode(M4, OUTPUT);

pinMode(sw, INPUT);

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

digitalWrite(M3, LOW);

digitalWrite(M4, LOW);

Serial.begin(9600);

}

void loop() {

if (digitalRead(sw)==0)

{

val = analogRead(0);float T=( val/1024.0)\*5000;float TEMP= T/10;

Serial.print(TEMP); //printing the numbers

Serial.print("C"); //and the unit

Serial.println(" ");

delay(500);

}

if(Serial.available() > 0)

{

state = Serial.read();

flag=0;

}

if (state == 'A'||state == '1')

{

digitalWrite(M1, LOW);

digitalWrite(M2, HIGH);

digitalWrite(M3, LOW);

digitalWrite(M4, HIGH);

flag=1;

delay(20);

}

else if (state == 'B'||state == '2')

{

digitalWrite(M1, HIGH);

digitalWrite(M2, LOW);

digitalWrite(M3, HIGH);

digitalWrite(M4, LOW);

flag=1;

delay(20);

}

else if (state == 'D'||state == '3')

{

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

digitalWrite(M4, HIGH);

digitalWrite(M3, LOW);

flag=1;

delay(20);

}

else if (state == 'C'||state == '4')

{

digitalWrite(M2, HIGH);

digitalWrite(M1, LOW);

digitalWrite(M3, LOW);

digitalWrite(M4, LOW);

flag=1;

delay(20);

}

else if (state == 'E'||state == '5')

{

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

digitalWrite(M3, LOW);

digitalWrite(M4, LOW);

flag=1;

delay(20);

}

}

**POWER SUPPLY DESCRIPTION**

A variable regulated power supply, also called a variable bench power supply, is one where you can continuously adjust the output voltage to your requirements. Varying the output of the power supply is the recommended way to test a project after having double checked parts placement against circuit drawings and the parts placement guide. This type of regulation is ideal for having a simple variable bench power supply. Actually this is quite important because one of the first projects a hobbyist should undertake is the construction of a variable regulated power supply. While a dedicated supply is quite handy e.g. 5V or 12V, it's much handier to have a variable supply on hand, especially for testing. Most digital logic circuits and processors need a 5 volt power supply. To use these parts we need to build a regulated 5 volt source. Usually you start with an unregulated power supply ranging from 9 volts to 24 volts DC (A 12 volt power supply is included with the [Beginner Kit](http://www.iguanalabs.com/1stled.htm) and the [Microcontroller Beginner Kit](http://www.iguanalabs.com/mbkit.htm).). To make a 5 volt power supply, we use a LM7805 voltage regulator IC (Integrated Circuit). The IC is shown below.

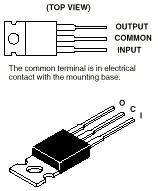


Fig 7.1 LM7805

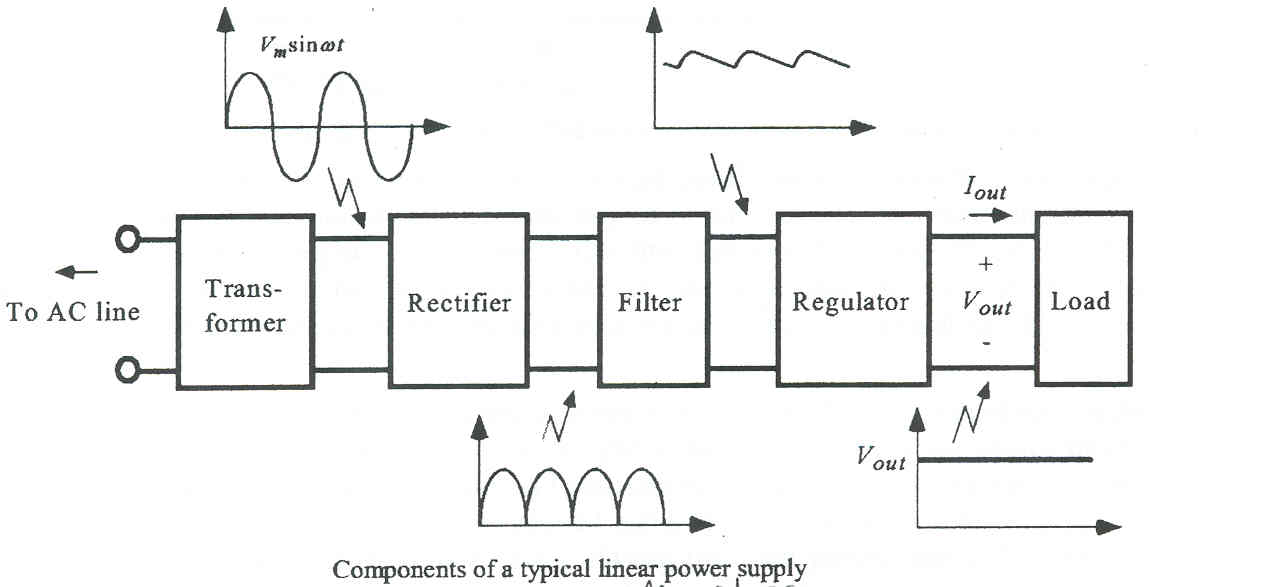
The LM7805 is simple to use. You simply connect the positive lead of your unregulated DC power supply (anything from 9VDC to 24VDC) to the Input pin, connect the negative lead to the Common pin and then when you turn on the power, you get a 5 volt supply from the Output pin.

**7.1 CIRCUIT FEATURES:**

Brief description of operation: Gives out well regulated +5V output, output current capability of 100 mA .

* Circuit protection: Built-in overheating protection shuts down output when regulator IC gets too hot
* Circuit complexity: Very simple and easy to build
* Circuit performance: Very stable +5V output voltage, reliable operation
* Availability of components: Easy to get, uses only very common basic components
* Design testing: Based on datasheet example circuit, I have used this circuit successfully as part of many electronics projects
* Applications: Part of electronics devices, small laboratory power supply
* Power supply voltage: Unregulated DC 8-18V power supply
* Power supply current: Needed output current + 5 mA
* Component costs: Few dollars for the electronics components + the input transformer cost

**7.2 BLOCK DIAGRAM:**



**7.3 CIRCUIT DIAGRAM:**

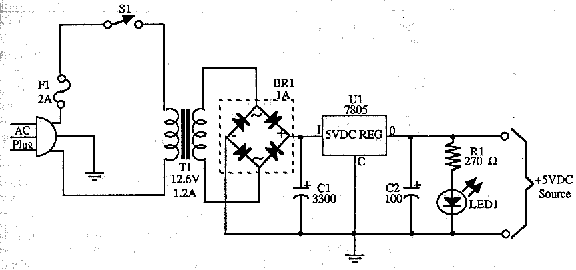


Fig 7.2 Regulated power supply

****

**BIBLIOGRAPHY**

**Websites viewed:**

* [www.philips.com](http://www.philips.com)
* [www.howstuffworks.com](http://www.howstuffworks.com)

[www.google.com](http://www.google.com)

[www.wiznet.com](http://www.wiznet.com)

* [www.wikipedia.com](http://www.wikipedia.com)
* [www.maxim-ic.com](http://www.maxim-ic.com/)
* [Jonathan Westhues's Proximity Website](http://cq.cx/prox.pl)
* Ethernet
* Sunroom technologies
* [IEEE Code of Ethics](http://www.ieee.org/portal/pages/about/whatis/code.html;jsessionid=Ghj4pYGn0TM2FK21vl4LLhycvcvCj1YftpW0yk6KmwB9HyvzHmq8%21-323916855)