PLEASE NOTE: THE MATTER PRODUCED BELOW IS REFERENCE MATERIAL ONLY FOR PREPARING THE PROJECT REPORT. IT IS NOT THE COMPLETE PROJECT REPORT.

A PROJECT REPORT ON

**IOT Early Flood Detection & Avoidance**

Submitted in partial fulfillment of the requirements

For the award of the degree

BACHELOR OF ENGINEERING

IN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ENGINEERING

SUBMITTED BY

-------------------- (--------------)

--------------------- (---------------)

--------------------- (---------------)

DEPARTMENT OF \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ENGINEERING

\_\_\_\_\_\_\_\_\_\_COLLEGE OF ENGINEERING

AFFILIATED TO \_\_\_\_\_\_\_\_\_\_\_ UNIVERSITY

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is result of studied blend of both research and knowledge.

We express our earnest gratitude to our internal guide, Assistant Professor \_\_\_\_\_\_\_\_\_\_\_\_\_\_, Department of ECE, our project guide, for his constant support, encouragement and guidance. We are grateful for his cooperation and his valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly for the completion of this project.

**DECLARATION**

We, the undersigned, declare that the project entitled **‘IOT Early Flood Detection & Avoidance’**, being submitted in partial fulfillment for the award of Bachelor of Engineering Degree in Electronics and Communication Engineering, affiliated to \_\_\_\_\_\_\_\_\_ University, is the work carried out by us.

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

**ABSTRACT**

# Flooding is usually brought on by an increased quantity of water in a water system, like a lake, river overflowing. On occasion a dam fractures, abruptly releasing a massive quantity of water. The outcome is that a number of the water travels into soil, and ‘flooding’ the region. Rivers are involving river banks, in a station. Aside from lack of products and house and office property, streets infrastructure flood water consists of bacteria and sewage flow of waste sites and chemical spillage which leads to a variety of diseases afterwards. Flood predictions need information like: The speed of change in river stage on a realtime basis, which may help indicate the seriousness and immediacy of this threat. Understanding of the form of storm generating the moisture, such as length, intensity and a real extent, which is valuable for discovering potential seriousness of the flood. In this system we make use of a Arduino Uno interfaced with 4 different sensors, named as Ultrasonic sensor for measuring water levels , float sensor detect full water, Flow sensor for knowing speed of water and humidity sensor. These combination of sensor are used to predict flood and alert respective authorities with help of IOT and sound instant alarm in nearby villages to instantly transmit information about possible floods. These sensors provide information over the IOT using Wifi module. On detection of conditions of flooding the system predicts the amount of time it would take to flood in a particular area and alerts the villages/areas that could be affected by it. The system also calculates the time it would take for flood to reach them and provides a time to people so that they can evacuate accordingly.

**2. INTRODUCTION TO EMBEDDED SYSTEMS**

**What is embedded system?**

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit

Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc .Lower end embedded systems - Generally 8,16 Bit Controllers used with an minimal operating systems and hardware layout designed for the specific purpose.

**SYSTEM DESIGN CALLS:**

Figure 3(a): Embedded system design calls

**EMBEDDED SYSTEM DESIGN CYCLE**

Figure 3(b) “V Diagram”

**Characteristics of Embedded System**

* An embedded system is any computer system hidden inside a product other than a computer.
* They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications.
  + Throughput – Our system may need to handle a lot of data in a short period of time.
  + Response–Our system may need to react to events quickly.
  + Testability–Setting up equipment to test embedded software can be difficult.
  + Debugability–Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem.
  + Reliability – embedded systems must be able to handle any situation without human intervention.
  + Memory space – Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists.
  + Program installation – you will need special tools to get your software into embedded systems.
  + Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power.
  + Processor hogs – computing that requires large amounts of CPU time can complicate the response problem.
  + Cost – Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.
* Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.

**APPLICATIONS**

1. Military and aerospace embedded software applications
2. Communication Applications
3. Industrial automation and process control software
4. Mastering the complexity of applications.
5. Reduction of product design time.
6. Real time processing of ever increasing amounts of data.
7. Intelligent, autonomous sensors.

**CLASSIFICATION**

* Real Time Systems.
* RTS is one which has to respond to events within a specified deadline.
* A right answer after the dead line is a wrong answer.

**RTS CLASSIFICATION**

* Hard Real Time Systems
* Soft Real Time System

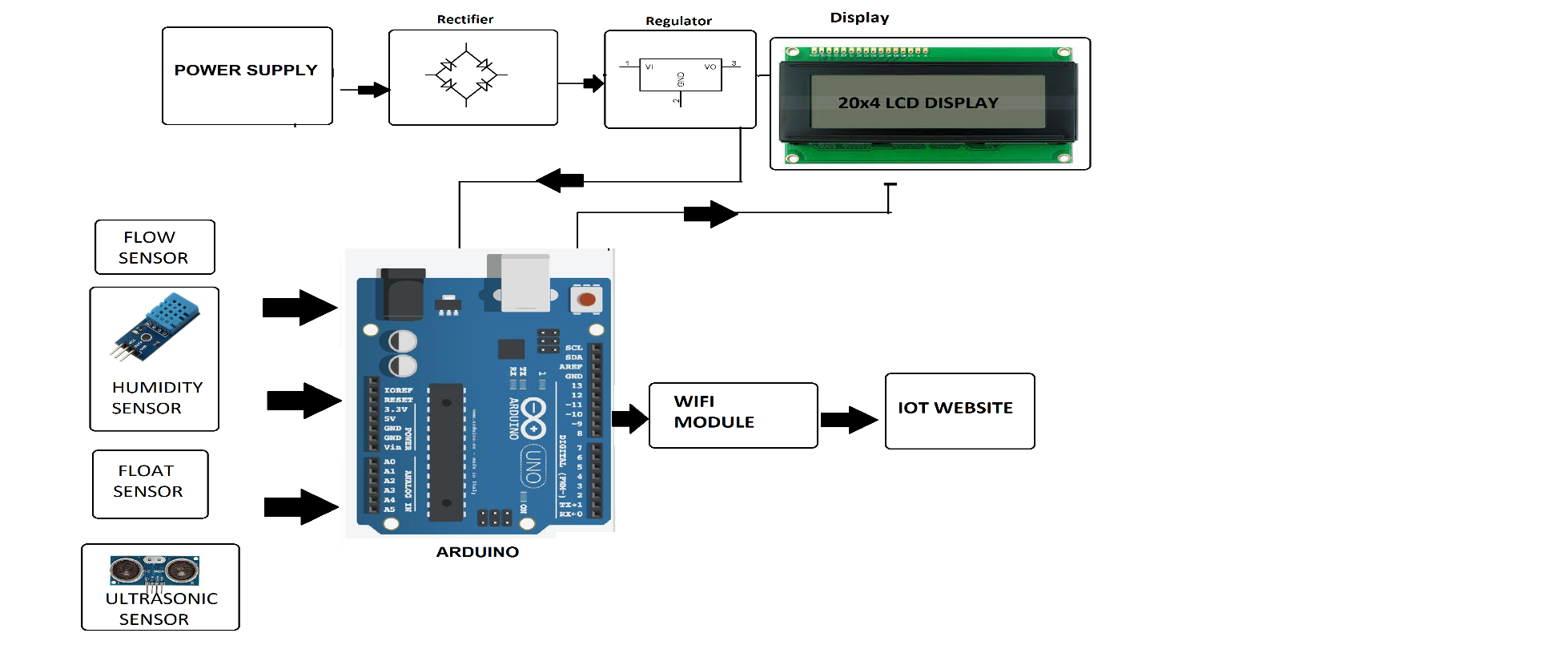
**HARD REAL TIME SYSTEM**

* "Hard" real-time systems have very narrow response time.
* Example: Nuclear power system, Cardiac pacemaker.

**SOFT REAL TIME SYSTEM**

* "Soft" real-time systems have reduced constrains on "lateness" but still must operate very quickly and repeatable.
* Example: Railway reservation system – takes a few extra seconds the data remains valid.

**3. BLOCK DIAGRAM**



**4. HARDWARE REQUIREMENT**

**HARDWARE COMPONENTS:**

* Arduino UNO
* Ultrasonic sensor
* Float sensor
* Humidity Sensor
* Flow sensor
* LCD 20x4
* Wifi module
* Resistors
* Capacitors
* Diodes

**BATTERY**

An electrical battery is a combination of one or more electrochemical cells, used to convert stored chemical energy into electrical energy. The battery has become a common power source for many household and industrial applications.

Batteries may be used once and discarded, or recharged for years as in standby power applications. Miniature cells are used to power devices such as hearing aids and wristwatches; larger batteries provide standby power for telephone exchanges or computer data centers.

**WORKING OF BATTERY:**

A battery is a device that converts chemical energy directly to electrical energy. It consists of a number of voltaic cells; each voltaic cell consists of two half cells connected in series by a conductive electrolyte containing anions and cat ions. One half-cell includes electrolyte and the electrode to which anions (negatively-charged ions) migrate, i.e. the anode or negative electrode; the other half-cell includes electrolyte and the electrode to which cat ions (positively-charged ions) migrate, i.e. the cathode or positive electrode. In the red ox reaction that powers the battery, reduction (addition of electrons) occurs to cat ions at the cathode, while oxidation (removal of electrons) occurs to anions at the anode. The electrodes do not touch each other but are electrically connected by the electrolyte. Many cells use two half-cells with different electrolytes. In that case each half-cell is enclosed in a container, and a separator that is porous to ions but not the bulk of the electrolytes prevents mixing.

Each half cell has an electromotive force (or emf), determined by its ability to drive electric current from the interior to the exterior of the cell. The net emf of the cell is the difference between the emfs of its half-cells. Therefore, if the electrodes have emfs and, in other words, the net emf is the difference between the reduction potentials of the half-reactions.

The electrical driving force or across the terminals of a cell is known as the terminal voltage (difference) and is measured in volts. The terminal voltage of a cell that is neither charging nor discharging is called the open-circuit voltage and equals the emf of the cell. Because of internal resistance, the terminal voltage of a cell that is discharging is smaller in magnitude than the open-circuit voltage and the terminal voltage of a cell that is charging exceeds the open-circuit voltage. An ideal cell has negligible internal resistance, so it would maintain a constant terminal voltage of until exhausted, then dropping to zero. If such a cell maintained 1.5 volts and stored a charge of one Coulomb then on complete discharge it would perform 1.5 Joule of work. In actual cells, the internal resistance increases under discharge, and the open circuit voltage also decreases under discharge. If the voltage and resistance are plotted against time, the resulting graphs typically are a curve; the shape of the curve varies according to the chemistry and internal arrangement employed.

An electrical battery is one or more [electrochemical cells](http://en.wikipedia.org/wiki/Electrochemical_cell) that convert stored chemical [energy](http://en.wikipedia.org/wiki/Energy) into electrical energy. Since the invention of the first battery (or "[voltaic pile](http://en.wikipedia.org/wiki/Voltaic_pile)") in 1800 by [Alessandro Volta](http://en.wikipedia.org/wiki/Alessandro_Volta), batteries have become a common power source for many household and industrial applications. According to a 2005 estimate, the worldwide battery industry generates [US$](http://en.wikipedia.org/wiki/United_States_dollar)48 [billion](http://en.wikipedia.org/wiki/1000000000_%28number%29) in sales each year, with 6% annual growth. There are two types of batteries: [primary batteries](http://en.wikipedia.org/wiki/Primary_battery) (disposable batteries), which are designed to be used once and discarded, and [secondary batteries](http://en.wikipedia.org/wiki/Secondary_battery) (rechargeable batteries), which are designed to be recharged and used multiple times. Miniature cells are used to power devices such as hearing aids and wristwatches; larger batteries provide standby power for telephone exchanges or computer data centers.

## Principle of operation

A battery is a device that converts chemical energy directly to electrical energy. It consists of a number of voltaic cells; each voltaic cell consists of two [half cells](http://en.wikipedia.org/wiki/Half_cell) connected in series by a conductive electrolyte containing anions and cations. One half-cell includes electrolyte and the electrode to which [anions](http://en.wikipedia.org/wiki/Ion#Ions) (negatively charged ions) migrate, i.e., the [anode](http://en.wikipedia.org/wiki/Anode) or negative electrode; the other half-cell includes electrolyte and the electrode to which [cations](http://en.wikipedia.org/wiki/Ion#Ions) (positively charged ions) migrate, i.e., the [cathode](http://en.wikipedia.org/wiki/Cathode) or positive electrode. In the [redox](http://en.wikipedia.org/wiki/Redox) reaction that powers the battery, cations are reduced (electrons are added) at the cathode, while anions are oxidized (electrons are removed) at the anode. The electrodes do not touch each other but are electrically connected by the [electrolyte](http://en.wikipedia.org/wiki/Electrolyte). Some cells use two half-cells with different electrolytes. A separator between half cells allows ions to flow, but prevents mixing of the electrolytes.

Each half cell has an electromotive force (or emf), determined by its ability to drive electric current from the interior to the exterior of the cell. The net emf of the cell is the difference between the emfs of its half-cells, as first recognized by Volta. Therefore, if the electrodes have emfs \mathcal{E}_1and \mathcal{E}_2, then the net emf is \mathcal{E}_{2}-\mathcal{E}_{1}; in other words, the net emf is the difference between the [reduction potentials](http://en.wikipedia.org/wiki/Reduction_potential) of the [half-reactions](http://en.wikipedia.org/wiki/Half-reaction). The electrical driving force or \displaystyle{\Delta V_{bat}}across the terminals of a cell is known as the terminal voltage (difference) and is measured in [volts](http://en.wikipedia.org/wiki/Volt). The terminal voltage of a cell that is neither charging nor discharging is called the [open-circuit voltage](http://en.wikipedia.org/wiki/Open-circuit_voltage) and equals the emf of the cell. Because of internal resistance, the terminal voltage of a cell that is discharging is smaller in magnitude than the open-circuit voltage and the terminal voltage of a cell that is charging exceeds the open-circuit voltage. An ideal cell has negligible internal resistance, so it would maintain a constant terminal voltage of \mathcal{E}until exhausted, then dropping to zero. If such a cell maintained 1.5 volts and stored a charge of one [coulomb](http://en.wikipedia.org/wiki/Coulomb) then on complete discharge it would perform 1.5 [joule](http://en.wikipedia.org/wiki/Joule) of work. In actual cells, the internal resistance increases under discharge, and the open circuit voltage also decreases under discharge. If the voltage and resistance are plotted against time, the resulting graphs typically are a curve; the shape of the curve varies according to the chemistry and internal arrangement employed.

As stated above, the voltage developed across a cell's terminals depends on the energy release of the chemical reactions of its electrodes and electrolyte. Alkaline and [carbon-zinc](http://en.wikipedia.org/wiki/Zinc-carbon_battery) cells have different chemistries but approximately the same emf of 1.5 volts; likewise [NiCd](http://en.wikipedia.org/wiki/Nickel-cadmium_battery) and [NiMH](http://en.wikipedia.org/wiki/Nickel-metal_hydride_battery) cells have different chemistries, but approximately the same emf of 1.2 volts. On the other hand the high electrochemical potential changes in the reactions of [lithium](http://en.wikipedia.org/wiki/Lithium) compounds give lithium cells emfs of 3 volts or more.

**Lead-acid**

Tried, tested, and trusted, lead-acid batteries have been with us since the middle of the 19th century. With an overall rating of 12 volts, they have six separate cells, each producing 2 volts. Crudely reduced to its basic components, each cell has a "spongy" [lead](http://www.explainthatstuff.com/lead.html) metal electrode (negative), a lead dioxide electrode (positive), and a sulfuric acid electrolyte. As the battery discharges, both electrodes become coated with lead sulfate and the sulfuric acid is largely converted into water, while electrons flow out around the external circuit to provide power.

Lead-acid batteries made it possible to start cars without the help of a dangerous and dirty hand crank. Normally, you never have to recharge them—because your car does that automatically. The battery discharges (gives up a little of its energy) to help the car's [gasoline engine](http://www.explainthatstuff.com/carengines.html) start up, and recharges (gets energy back again) when the engine begins generating electrical energy through a device called an alternator. As for disadvantages, lead-acid batteries are relatively big, surprisingly heavy (try lifting one!), expensive, and can't be fully charged and discharged too many times. Another problem is their use of toxic lead metal, which can cause environmental problems when they're dumped in landfills.

**Nickel-cadmium**

Nickel-cadmium (NiCd, pronounced "nicad") are widely used as replacements for disposable 1.5-volt batteries in things like toys, flashlights, and power tools. They're relatively cheap, can be charged and discharged hundreds of times, and, properly treated, will last about a decade.

Although very dependable, it's often said that NiCd batteries need to be discharged fully before you charge them up or the amount of charge they will store (and their effective lifespan) can be greatly reduced. Opinions vary on whether this is true and, if so, why it happens, but as a rule of thumb, regularly discharging batteries completely and then recharging them is a good practice. Another problem with NiCd batteries is the toxic cadmium metal they contain. If they are buried in a landfill, instead of properly recycled, the cadmium can escape into the soil and could potentially [pollute](http://www.explainthatstuff.com/waterpollution.html) watercourses nearby.

**Nickel-metal-hydride (NiMH)**

Nickel metal hydride batteries work in a similar way, but suffer less from the so-called "memory effect." They became a popular alternative to NiCd batteries in the 1990s, partly because of environmental concerns about cadmium. NiMH batteries work more effectively in gadgets like cellphones, which are often "topped-up" with a quick recharge instead of a complete discharge and recharge (which is more typical with something like power tools).

**Lithium-ion**

Lithium-ion batteries are the fastest-growing type of rechargeables; there are probably lithium-ion batteries in your [cellphone](http://www.explainthatstuff.com/cellphones.html), [MP3 player](http://www.explainthatstuff.com/how-mp3players-work.html), and laptop [computer](http://www.explainthatstuff.com/howcomputerswork.html). What's so good about lithium? It's a lightweight metal that easily forms ions, so it's excellent for making batteries. The latest lithium-ion batteries can store about twice as much energy as traditional NiCd rechargeables, work at higher voltages, and are more environmentally friendly, but don't last as long. Even so, they can be charged and discharged hundreds of times and typically last several years, so they're great for everyday use in electronic gadgets that aren't meant to last that long.

How do they work? When you plug a cellphone or laptop into the power supply, the lithium-ion battery inside starts buzzing with chemical activity. The battery's job is to store as much electricity as possible, as fast as possible. It does this through a chemical reaction that shunts lithium ions (lithium atoms that have lost an electron to become positively charged) from one part of the battery to another. When you unplug the power and use your laptop or phone, the battery switches into reverse: the ions move the opposite way and the battery gradually loses its charge. Read more in our main article on [how lithium-ion batteries work](http://www.explainthatstuff.com/how-lithium-ion-batteries-work.html).

**4.2 VOLTAGE REGULATOR 7805**

**Features**

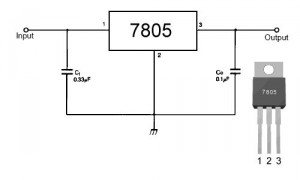
• Output Current up to 1A.

• Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V.

• Thermal Overload Protection.

• Short Circuit Protection.

• Output Transistor Safe Operating Area Protection.



**Description**

The LM78XX/LM78XXA series of three-terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a Wide range of applications. Each type employs internal current limiting, thermal shutdown and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output Current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

**Internal Block Diagram**

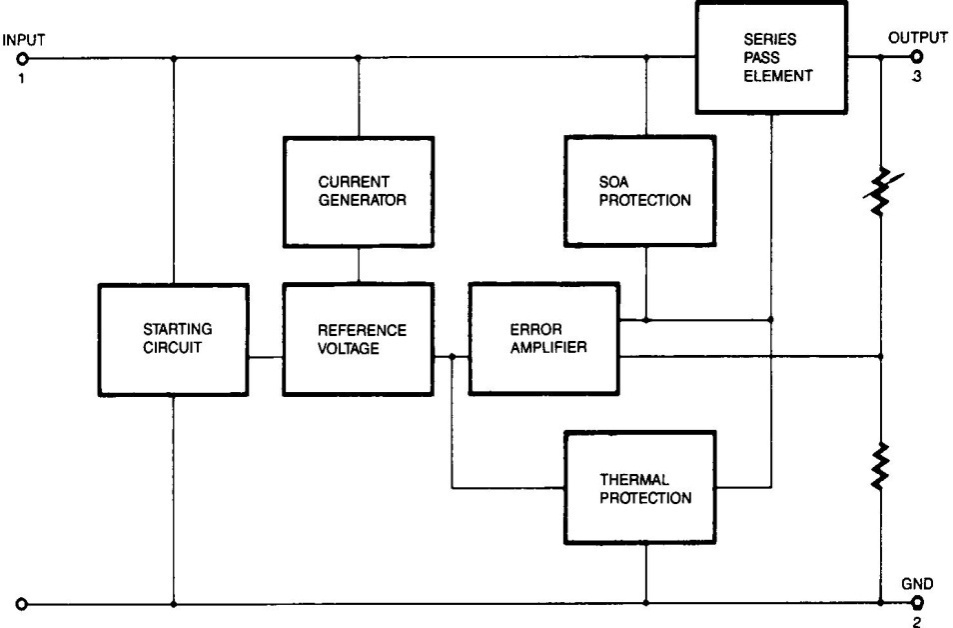
****

FIG 4.2(a): BLOCK DIAGRAM OF VOLTAGE REGULATOR

**Absolute Maximum Ratings**

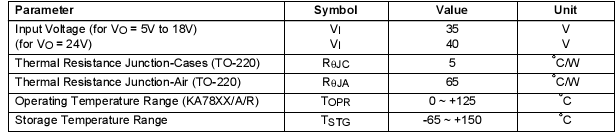
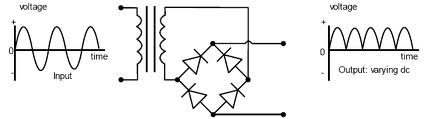
****

TABLE 4.2(b): RATINGS OF THE VOLTAGE REGULATOR

[](http://www.google.co.in/imgres?imgurl=http://www.evl.uic.edu/sjames/dvl/images/51262.jpg&imgrefurl=http://vinosysabores.com/voltage-regulator-7805&page=7&usg=__8M5wBB3ts7AF3gXgXNgYUtUKog4=&h=300&w=300&sz=7&hl=en&start=14&zoom=1&tbnid=gdGbqNw28x40UM:&tbnh=116&tbnw=116&ei=MRHeTdXKDo-8vgO735nABQ&prev=/search?q=voltage+regulator+7805&hl=en&sa=X&biw=1003&bih=539&tbm=isch&itbs=1)

**4.3 RECTIFIER**

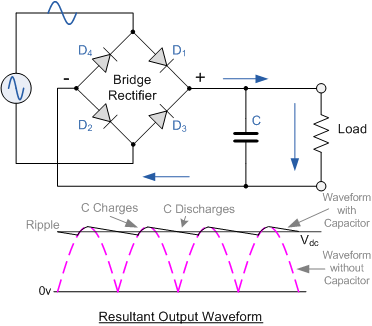
A rectifier is an electrical device that converts [alternating current](http://en.wikipedia.org/wiki/Alternating_current) (AC), which periodically reverses direction, to [direct current](http://en.wikipedia.org/wiki/Direct_current) (DC), current that flows in only one direction, a process known as rectification. Rectifiers have many uses including as components of [power supplies](http://en.wikipedia.org/wiki/Power_supply) and as [detectors](http://en.wikipedia.org/wiki/Detector_(radio)) of [radio](http://en.wikipedia.org/wiki/Radio) signals. Rectifiers may be made of [solid state](http://en.wikipedia.org/wiki/Solid_state_(electronics)) [diodes](http://en.wikipedia.org/wiki/Diode), [vacuum tube](http://en.wikipedia.org/wiki/Vacuum_tube) diodes, [mercury arc valves](http://en.wikipedia.org/wiki/Mercury_arc_valve), and other components. The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification. In positive half cycleonly two diodes( 1 set of parallel diodes) will conduct, in negative half cycle remaining two diodes will conduct and they will conduct only in forward bias only.



**4.4 FILTER**

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.

The simple capacitor filter is the most basic type of power supply filter. The use of this filter is very limited. It is sometimes used on extremely high-voltage, low-current power supplies for cathode-ray and similar electron tubes that require very little load current from the supply. This filter is also used in circuits where the power-supply ripple frequency is not critical and can be relatively high.below figure can show how the capacitor chages and discharges.



**Atmega 328**



ATmega328 is an eight bit Microcontroller. It can handle the data sized of up to eight bits. It is an AVR based miniaturized scale controller. Its built in interior memory is around 32KB. It works extending from 3.3V to 5V. It has a capacity to store the information notwithstanding when the electrical supply is expelled from its biasing terminals. Its brilliant highlights incorporate the cost effectiveness, low power dispersal, programming lock for security purposes, genuine clock counter with isolated oscillator. It's ordinarily utilized as a part of Embedded Systems applications.

The computer on one hand is intended to play out all the universally useful errands on a solitary machine like you can utilize a computer to run a product to perform estimations or you can utilize a computer to store some media record or to get to web through the program, while the microcontrollers are intended to perform just the particular undertakings, for e.g., turning the AC off naturally when room temperature drops to a specific characterized confine and again turning it ON when temperature transcends as far as possible.

There are number of mainstream groups of microcontrollers which are utilized as a part of various applications according to their capacity and attainability to play out the coveted errand, most normal of these are 8051, AVR and PIC microcontrollers. In this we will present you with AVR group of microcontroller.

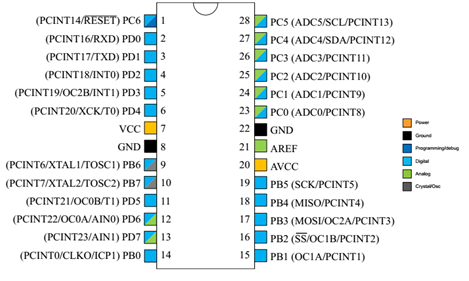
There are number of well known groups of microcontrollers which are utilized as a part of various applications according to their capacity and achievability to play out the coveted undertaking, most normal of these are 8051, AVR and PIC microcontrollers. In this we will present you with AVR group of microcontrollers.

The present microcontrollers are very different from what it were in the underlying stage, and the quantity of producers are considerably more in check than it was 10 years or two prior. At current portion of the real producers are Microchip (production: PIC microcontrollers), Atmel (distribution: AVR microcontrollers), Hitachi, Phillips, Maxim, NXP, Intel and so forth. Our intrest is upon ATmega32. It has a place with Atmel's AVR arrangement small scale controller family. How about we see the highlights.

**Features include:**

* High Performance, Low Power Design
* 8-Bit Microcontroller Atmel® AVR® advanced RISC architecture
  + 131 Instructions most of which are executed in a single clock cycle
  + Up to 20 MIPS throughput at 20 MHz
  + 32 x 8 working registers
  + 2 cycle multiplier
* Memory Includes
  + 32KB of of programmable FLASH
  + 1KB of EEPROM
  + 2KB SRAM
  + 10,000 Write and Erase Cycles for Flash and 100,000 for EEPROM
  + Data retention for 20 years at 85°C and 100 years at 25°C
  + Optional boot loader with lock bits
    - In System Programming (ISP) by via boot loader
    - True Read-While-Write operation
  + Programming lock available for software security
* Features Include
  + 2 x 8-bit Timers/Counters each with independent prescaler and compare modes
  + A single 16-bit Timer/Counter with an idependent prescaler, compare and capture modes
  + Real time counter with independent oscillator
  + 10 bit, 6 channel analog to digital Converter
  + 6 pulse width modulation channels
  + Internal temperature sensor
  + Serial USART (Programmable)
  + Master/Slave SPI Serial Interface - (Philips I2C compatible)
  + Programmable watchdog timer with independent internal oscillator
  + Internal analog comparator
  + Interrupt and wake up on pin change
* Additional Features Features
  + Internal calibrated oscillator
  + Power on reset and programmable brown out detection
  + External and internal interrupts
  + 6 sleep modes including idle, ADC noise reduction, power save, power down, standby, and extended standby
* I/O and Package
  + 23 programmable I/O lines
  + 28 pin PDIP package
* Operating voltage:
  + 1.8 - 5.5V
* Operating temperature range:
  + 40°C to 85°C
* Speed Grades:
  + 0-4 MHz at 1.8-5.5V
  + 0-10 MHz at 2.7-5.5V
  + 0-20 MHz at 4.5-5.5V
* Low power consumption mode at 1.8V, 1 MHz and 25°C:
  + Active Mode: 0.3 mA
  + Power-down Mode: 0.1 μA
  + Power-save Mode: 0.8 μA (Including 32 kHz RTC)

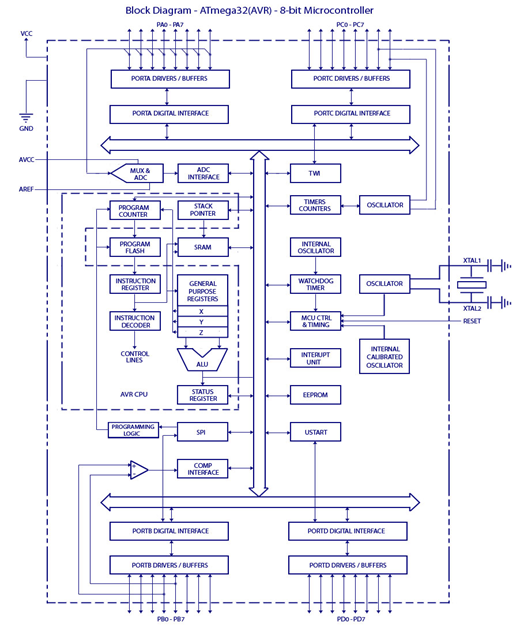
|  |  |
| --- | --- |
| Flash: | 32 KBytes |
| EEPROM: | 1 KBytes |
| SRAM: | 2 KBytes |
| Max I/O Pins: | 23 |
| Frequency Max: | 20 MHz |
| VCC: | 1.8-5.5 |
| 10-bit A/D Channels: | 6 |
| Analog Comparator: | Yes |
| 16-bit Timers: | 1 |
| 8-bit Timer: | 2 |
| Brown Out Detector: | Yes |
| Ext Interrupts: | 2 |
| Hardware Multiplier: | Yes |
| Interrupts: | 26 |
| ISP: | Yes |
| On Chip Oscillator: | Yes |
| PWM Channels: | 6 |
| RTC: | Yes |
| Self Program Memory: | Yes |
| SPI: | 1 |
| TWI: | Yes |
| UART: | 1 |
| Watchdog: | Yes |
| Pacakage: | Lead Free PDIP 28 |
| **Pin Diagram** |  |



**Pin configuration**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin No.** | **Pin name** | **Description** | **Secondary Function** |
| 1 | PC6 (RESET) | Pin6 of  PORTC | Pin by default is used as RESET pin. PC6 can only be used as I/O pin when RSTDISBL Fuse is programmed. |
| 2 | PD0 (RXD) | Pin0  of  PORTD | RXD (Data Input Pin for USART)  USART Serial Communication Interface  [Can be used for programming] |
| 3 | PD1 (TXD) | Pin1 of  PORTD | TXD (Data Output Pin for USART)  USART Serial Communication Interface  [Can be used for programming]    INT2( External Interrupt 2 Input) |
| 4 | PD2 (INT0) | Pin2  of  PORTD | External Interrupt source 0 |
| 5 | PD3 (INT1/OC2B) | Pin3  of  PORTD | External Interrupt source1    OC2B(PWM - Timer/Counter2 Output Compare Match B Output) |
| 6 | PD4 (XCK/T0) | Pin4  of  PORTD | T0( Timer0 External Counter Input)  XCK ( USART External Clock I/O) |
| 7 | VCC |  | Connected to positive voltage |
| 8 | GND |  | Connected to ground |
| 9 | PB6 (XTAL1/TOSC1) | Pin6  of  PORTB | XTAL1 (Chip Clock Oscillator pin 1 or External clock input)  TOSC1 (Timer Oscillator pin 1) |
| 10 | PB7 (XTAL2/TOSC2) | Pin7  of  PORTB | XTAL2 (Chip Clock Oscillator pin 2)  TOSC2 (Timer Oscillator pin 2) |
| 11 | PD5  (T1/OC0B) | Pin5 of  PORTD | T1(Timer1 External Counter Input)    OC0B(PWM - Timer/Counter0 Output Compare Match B Output) |
| 12 | PD6 (AIN0/OC0A) | Pin6  of  PORTD | AIN0(Analog Comparator Positive I/P)    OC0A(PWM - Timer/Counter0 Output Compare Match A Output) |
| 13 | PD7 (AIN1) | Pin7  of  PORTD | AIN1(Analog Comparator Negative I/P) |
| 14 | PB0 (ICP1/CLKO) | Pin0  of  PORTB | ICP1(Timer/Counter1 Input Capture Pin)    CLKO (Divided System Clock. The divided system clock can be output on the PB0 pin) |
| 15 | PB1 (OC1A) | Pin1  of  PORTB | OC1A (Timer/Counter1 Output Compare Match A Output) |
| 16 | PB2 (SS/OC1B) | Pin2  of  PORTB | SS (SPI Slave Select Input).  This pin is low when controller acts as slave.  [Serial Peripheral Interface (SPI) for programming]    OC1B (Timer/Counter1 Output Compare Match B Output) |
| 17 | PB3 (MOSI/OC2A) | Pin3  of  PORTB | MOSI (Master Output Slave Input). When controller acts as slave, the data is received by this pin. [Serial Peripheral Interface (SPI) for programming]  OC2 (Timer/Counter2 Output Compare Match Output) |
| 18 | PB4 (MISO) | Pin4  of  PORTB | MISO (Master Input Slave Output). When controller acts as slave, the data is sent to master by this controller through this pin.    [Serial Peripheral Interface (SPI) for programming] |
| 19 | PB5 (SCK) | Pin5  of  PORTB | SCK (SPI Bus Serial Clock). This is the clock shared between this controller and other system for accurate data transfer.  [Serial Peripheral Interface (SPI) for programming] |
| 20 | AVCC |  | Power for Internal ADC Converter |
| 21 | AREF |  | Analog Reference Pin for ADC |
| 22 | GND |  | GROUND |
| 23 | PC0 (ADC0) | Pin0  of  PORTC | ADC0 (ADC Input Channel 0) |
| 24 | PC1 (ADC1) | Pin1  of  PORTC | ADC1 (ADC Input Channel 1) |
| 25 | PC2 (ADC2) | Pin2  of  PORTC | ADC2 (ADC Input Channel 2) |
| 26 | PC3 (ADC3) | Pin3  of  PORTC | ADC3 (ADC Input Channel 3) |
| 27 | PC4 (ADC4/SDA) | Pin4  of  PORTC | ADC4 (ADC Input Channel 4)  SDA (Two-wire Serial Bus Data Input/output Line) |
| 28 | PC5 (ADC5/SCL) | Pin5  of  PORTC | ADC5 (ADC Input Channel 5)  SCL (Two-wire Serial Bus Clock Line) |

**Block Diagram of Atmega328**



**Applications**

There are hundreds of applications for ATMEGA328P:

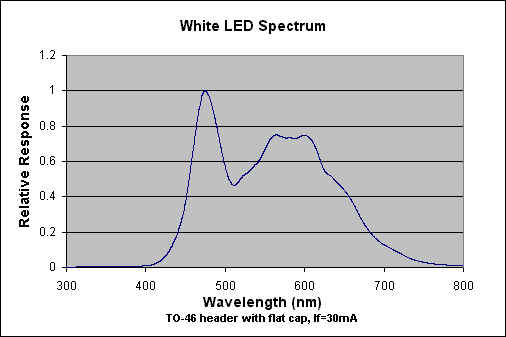
* Used in ARDUINO UNO, ARDUINO NANO and ARDUINO MICRO boards.
* Industrial control systems.
* SMPS and Power Regulation systems.
* Digital data processing.
* Analog signal measuring and manipulations.
* Embedded systems like coffee machine, vending machine.
* Motor control systems.
* Display units.
* Peripheral Interface system.

**LED**

Light Emitting Diodes (LED) have recently become available that are white and bright, so bright that they seriously compete with incandescent lamps in lighting applications. They are still pretty expensive as compared to a GOW lamp but draw much less current and project a fairly well focused beam.

The diode in the photo came with a neat little reflector that tends to sharpen the beam a little but doesn't seem to add much to the overall intensity.

When run within their ratings, they are more reliable than lamps as well. Red LEDs are now being used in automotive and truck tail lights and in red traffic signal lights. You will be able to detect them because they look like an array of point sources and they go on and off instantly as compared to conventional incandescent lamps.



LEDs are monochromatic (one color) devices. The color is determined by the band gap of the semiconductor used to make them. Red, green, yellow and blue LEDs are fairly common. White light contains all colors and cannot be directly created by a single LED. The most common form of "white" LED really isn't white. It is a Gallium Nitride blue LED coated with a phosphor that, when excited by the blue LED light, emits a broad range spectrum that in addition to the blue emission, makes a fairly white light.

There is [a claim](http://www.bivar.com/bullethtml/bull-nov2.htm) that these white LED's have a limited life. After 1000 hours or so of operation, they tend to yellow and dim to some extent. Running the LEDs at more than their rated current will certainly accelerate this process.

There are two primary ways of producing high intensity white-light using LED’S. One is to use individual LED’S that emit three [primary colours](http://en.wikipedia.org/wiki/Primary_color)—red, green, and blue—and then mix all the colours to form white light. The other is to use a phosphor material to convert monochromatic light from a blue or UV LED to broad-spectrum white light, much in the same way a fluorescent light bulb works. Due to [metamerism](http://en.wikipedia.org/wiki/Metamerism_(color)), it is possible to have quite different spectra that appear white.

LEDs are semiconductor devices. Like transistors, and other diodes, LEDs are made out of silicon. What makes an LED give off light are the small amounts of chemical impurities that are added to the silicon, such as gallium, arsenide, indium, and nitride.

When current passes through the LED, it emits photons as a byproduct. Normal light bulbs produce light by heating a metal filament until it is white hot. LEDs produce photons directly and not via heat, they are far more efficient than incandescent bulbs.

LED circuit symbol

Fig 3.1(a): circuit symbol

Not long ago LEDs were only bright enough to be used as indicators on dashboards or electronic equipment. But recent advances have made LEDs bright enough to rival traditional lighting technologies. Modern LEDs can replace incandescent bulbs in almost any application.

**Types of LED’S**

LEDs are produced in an array of shapes and sizes. The 5 mm cylindrical package is the most common, estimated at 80% of world production. The color of the plastic lens is often the same as the actual color of light emitted, but not always. For instance, purple plastic is often used for infrared LEDs, and most blue devices have clear housings. There are also LEDs in extremely tiny packages, such as those found on blinkers and on cell phone keypads. The main types of LEDs are miniature, high power devices and custom designs such as alphanumeric or multi-color.

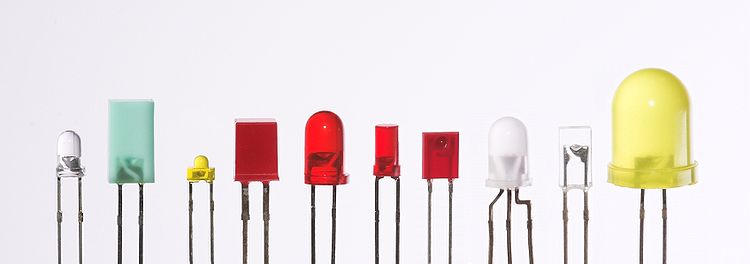
**[](http://en.wikipedia.org/wiki/File:Verschiedene_LEDs.jpg)**

Fig 3.1(b) Different types of LED’S

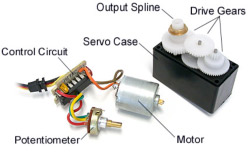
**Servo motors** have been around for a long time and are utilized in many applications. They are small in size but pack a big punch and are very energy-efficient. These features allow them to be used to operate remote-controlled or radio-controlled [**toy cars**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Education%20%26%20Hobby%20Kits&category=70&refine=1&position=1&history=yajv7t0n%7CfreeText~cars%5Esearch_type~jamecoall%5EprodPage~15%5Epage~SEARCH%252BNAV), [**robots**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Robotics&category=75&refine=1&position=1&history=jpodqbl6%7CfreeText~robot%5Esearch_type~jamecoall%5EprodPage~15%5Epage~SEARCH%252BNAV) and airplanes. Servo motors are also used in industrial applications, robotics, in-line manufacturing, pharmaceutics and food services. But how do the little guys work?   
  
The servo circuitry is built right inside the motor unit and has a positionable shaft, which usually is fitted with a [**gear**](http://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&storeId=10001&productId=1810072&catalogId=10001) (as shown below). The motor is controlled with an electric signal which determines the amount of movement of the shaft. 

### What's inside the servo?

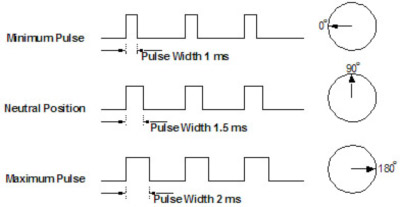
*Hitec HS-322HD Standard Heavy Duty Servo*

To fully understand how the servo works, you need to take a look under the hood. Inside there is a pretty simple set-up: a small [**DC motor**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Electromechanical&category=35&refine=1&position=1&history=28hlc38h%7CfreeText~dc%2Bmotor%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV),[**potentiometer**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Passive%20Components&category=20&refine=1&position=1&history=d0ww5ora%7CfreeText~potentiometer%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV), and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.   
  
When the shaft of the motor is at the desired position, [**power**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Power%20Supplies%20%26%20Wall%20Adapters&category=45) supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through the [**signal wire**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=cat_25&subCategoryName=Wire%20%26%20Cable%20%2F%20Bulk%20Wire&category=2550&refine=1&position=1&history=wus4q9hh%7CsubCategoryName~Wire%2B%2526%2BCable%5Ecategory~25%5EcategoryName~category_root%5EprodPage~50%5Epage~SEARCH%252BNAV). The motor's speed is proportional to the difference between its actual position and desired position. So if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called **proportional control.** This means the motor will only run as hard as necessary to accomplish the task at hand, a very efficient little guy. 

### How is the servo controlled?

*The guts of a servo motor (L) and an assembled servo (R)*

Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90 degrees in either direction for a total of 180 degree movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the [**motor**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&freeText=motor&search_type=jamecoall) determines position of the shaft, and based on the duration of the pulse sent via the control wire; the [**rotor**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=cat_3540&subCategoryName=Electromechanical%20%2F%20Switches%20%2F%20Rotary&category=354055&refine=1&position=1&history=kv7hqebe%7CfreeText~rotor%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV%405hha4bcd%7Ccategory~35%5EcategoryName~category_root%5Eposition~1%5Erefine~1%5EsubCategoryName~Electromechanical%5EprodPage~50%5Epage~SEARCH%252BNAV) will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90-degree position. Shorter than 1.5ms moves it to 0 degrees, and any longer than 1.5ms will turn the servo to 180 degrees. 

*Variable Pulse width control servo position*

When these servos are commanded to move, they will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is called the **torque rating** of the servo. Servos will not hold their position forever though; the position pulse must be repeated to instruct the servo to stay in position. 

### Types of Servo Motors

There are two types of servo motors - AC and DC. AC servo can handle higher current surges and tend to be used in industrial machinery.[**DC servos**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=cat_75&subCategoryName=Robotics%20%2F%20Motors&category=7545&refine=1&position=1&history=ko9l7dni%7CfreeText~servo%2Bmotor%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV%40gu9oa7zv%7Ccategory~75%5EcategoryName~category_root%5Eposition~1%5Erefine~1%5EsubCategoryName~Robotics%5EprodPage~50%5Epage~SEARCH%252BNAV) are not designed for high current surges and are usually better suited for smaller applications. Generally speaking, DC motors are less expensive than their AC counterparts. These are also servo motors that have been built specifically for continuous rotation, making it an easy way to get your robot moving. They feature two ball bearings on the output shaft for reduced friction and easy access to the rest-point adjustment [**potentiometer**](http://www.jameco.com/webapp/wcs/stores/servlet/JamecoSearch?langId=-1&storeId=10001&catalogId=10001&categoryName=category_root&subCategoryName=Passive%20Components&category=20&refine=1&position=1&history=d0ww5ora%7CfreeText~potentiometer%5Esearch_type~jamecoall%5EprodPage~50%5Epage~SEARCH%252BNAV). 

### Servo Motor Applications

Servos are used in radio-controlled airplanes to position control surfaces like elevators, rudders, walking a robot, or operating [**grippers**](http://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&productId=358811&position=1&category=7540&catalogId=10001&subCategoryName=Robotics+%2F+Mechanical&storeId=10001&refine=1&app.products.maxperpage=15&parentCategoryId=75%24%247540&categ). Servo motors are small, have built-in control circuitry and have good power for their size.   
  
In food services and pharmaceuticals, the tools are designed to be used in harsher environments, where the potential for corrosion is high due to being washed at high pressures and temperatures repeatedly to maintain strict hygiene standards. Servos are also used in **in-line manufacturing**, where high repetition yet precise work is necessary.   
  
Of course, you don't have to know how a servo works to use one, but as with most electronics, the more you understand, the more doors open for [**expanded projects**](http://www.jameco.com/Jameco/Pressroom/DIY.html) and projects' capabilities. Whether you're a [**hobbyist building robots**](http://www.jameco.com/Jameco/workshop/diy/rodney.html), an engineer designing industrial systems, or just constantly curious, where will servo motors take you?

**4.9 PUSH BUTTONS**

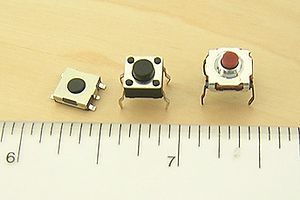
[](http://en.wikipedia.org/wiki/File:Tactile_switches.jpg)

Fig 4.9(a): Push Buttons

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state. Different people use different terms for the "pushing" of the button, such as press, depress, mash, and punch.

**Uses:**

In industrial and commercial applications push buttons can be linked together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process have no electrical circuits for control.

Pushbuttons are often color-coded to associate them with their function so that the operator will not push the wrong button in error. Commonly used colors are red for stopping the machine or process and green for starting the machine or process.

Red pushbuttons can also have large heads (mushroom shaped) for easy operation and to facilitate the stopping of a machine. These pushbuttons are called emergency stop buttons and are mandated by the electrical code in many jurisdictions for increased safety. This large mushroom shape can also be found in buttons for use with operators who need to wear gloves for their work and could not actuate a regular flush-mounted push button. As an aid for operators and users in industrial or commercial applications, a pilot light is commonly added to draw the attention of the user and to provide feedback if the button is pushed. Typically this light is included into the center of the pushbutton and a lens replaces the pushbutton hard center disk.

The source of the energy to illuminate the light is not directly tied to the contacts on the back of the pushbutton but to the action the pushbutton controls. In this way a start button when pushed will cause the process or machine operation to be started and a secondary contact designed into the operation or process will close to turn on the pilot light and signify the action of pushing the button caused the resultant process or action to start.

In popular culture, the phrase "the button" refers to a (usually fictional) button that a military or government leader could press to launch nuclear weapons.

## Push to ON button:

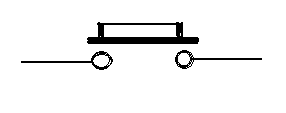
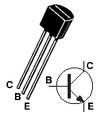


Fig. 4.9(b): push on button

Initially the two contacts of the button are open. When the button is pressed they become connected. This makes the switching operation using the push button.

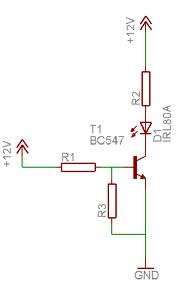
**4.10 BC547**

The BC547 transistor is an NPN Epitaxial Silicon Transistor. The BC547 transistor is a general-purpose transistor in small plastic packages. It is used in general-purpose switching and amplification BC847/BC547 series 45 V, 100 mA NPN general-purpose transistors.



BC 547 TRANSISTOR PINOUTS

The BC547 transistor is an NPN bipolar transistor, in which the letters "N" and "P" refer to the majority charge carriers inside the different regions of the transistor. Most bipolar transistors used today are NPN, because electron mobility is higher than hole mobility in semiconductors, allowing greater currents and faster operation. NPN transistors consist of a layer of P-doped semiconductor (the "base") between two N-doped layers. A small current entering the base in common-emitter mode is amplified in the collector output. In other terms, an NPN transistor is "on" when its base is pulled high relative to the emitter. The arrow in the NPN transistor symbol is on the emitter leg and points in the direction of the conventional current flow when the device is in forward active mode. One mnemonic device for identifying the symbol for the NPN transistor is "not pointing in." An NPN transistor can be considered as two diodes with a shared anode region. In typical operation, the emitter base junction is forward biased and the base collector junction is reverse biased. In an NPN transistor, for example, when a positive voltage is applied to the base emitter junction, the equilibrium between thermally generated carriers and the repelling electric field of the depletion region becomes unbalanced, allowing thermally excited electrons to inject into the base region. These electrons wander (or "diffuse") through the base from the region of high concentration near the emitter towards the region of low concentration near the collector. The electrons in the base are called minority carriers because the base is doped p-type which would make holes the majority carrier in the base.



Whenever base is high, then current starts flowing through base and emitter and after that only current will pass from collector to emitter. So that the LED which is connected to collector will glow to indicate that transistor is ON.

**4.11 1N4007**

Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must he kept in mind while using any type of diode.

1. Maximum forward current capacity
2. Maximum reverse voltage capacity
3. Maximum forward voltage capacity

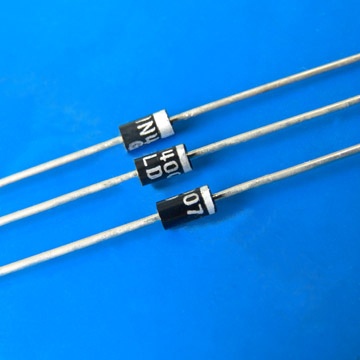


Fig: 1N4007 diodes

The number and voltage capacity of some of the important diodes available in the market are as follows:

* Diodes of number IN4001, IN4002, IN4003, IN4004, IN4005, IN4006 and IN4007 have maximum reverse bias voltage capacity of 50V and maximum forward current capacity of 1 Amp.
* Diode of same capacities can be used in place of one another. Besides this diode of more capacity can be used in place of diode of low capacity but diode of low capacity cannot be used in place of diode of high capacity. For example, in place of IN4002; IN4001 or IN4007 can be used but IN4001 or IN4002 cannot be used in place of IN4007.The diode BY125made by company BEL is equivalent of diode from IN4001 to IN4003. BY 126 is equivalent to diodes IN4004 to 4006 and BY 127 is equivalent to diode IN4007.

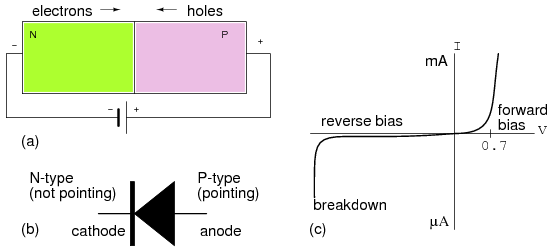


Fig:PN Junction diode

PN JUNCTION OPERATION

Now that you are familiar with P- and N-type materials, how these materials are joined together to form a diode, and the function of the diode, let us continue our discussion with the operation of the PN junction. But before we can understand how the PN junction works, we must first consider current flow in the materials that make up the junction and what happens initially within the junction when these two materials are joined together.

Current Flow in the N-Type Material

Conduction in the N-type semiconductor, or crystal, is similar to conduction in a copper wire. That is, with voltage applied across the material, electrons will move through the crystal just as current would flow in a copper wire. This is shown in figure 1-15. The positive potential of the battery will attract the free electrons in the crystal. These electrons will leave the crystal and flow into the positive terminal of the battery. As an electron leaves the crystal, an electron from the negative terminal of the battery will enter the crystal, thus completing the current path. Therefore, the majority current carriers in the N-type material (electrons) are repelled by the negative side of the battery and move through the crystal toward the positive side of the battery.

Current Flow in the P-Type Material

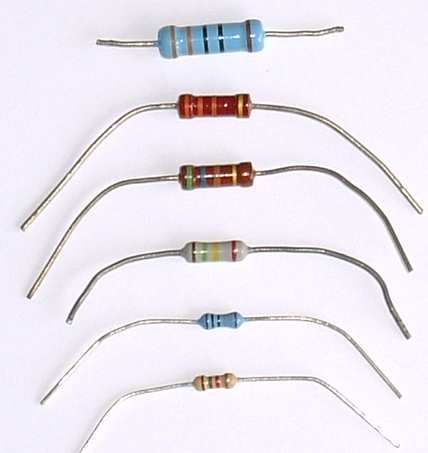
Current flow through the P-type material is illustrated. Conduction in the P material is by positive holes, instead of negative electrons. A hole moves from the positive terminal of the P material to the negative terminal. Electrons from the external circuit enter the negative terminal of the material and fill holes in the vicinity of this terminal. At the positive terminal, electrons are removed from the covalent bonds, thus creating new holes. This process continues as the steady stream of holes (hole current) moves toward the negative terminal

**4.12 RESISTORS**

A resistor is a two-terminal electronic component designed to oppose an electric current by producing a voltage drop between its terminals in proportion to the current, that is, in accordance with Ohm's law:

*V* = *IR*

Resistors are used as part of electrical networks and electronic circuits. They are extremely commonplace in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).



The primary characteristics of resistors are their resistance and the power they can dissipate. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance depends upon the materials constituting the resistor as well as its physical dimensions; it's determined by design.

Resistors can be integrated into hybrid and printed circuits, as well as integrated circuits. Size, and position of leads (or terminals) are relevant to equipment designers; resistors must be physically large enough not to overheat when dissipating their power.

A resistor is a two-[terminal](http://en.wikipedia.org/wiki/Terminal_%28electronics%29) [passive](http://en.wikipedia.org/wiki/Passivity_%28engineering%29) [electronic component](http://en.wikipedia.org/wiki/Electronic_component) which implements [electrical resistance](http://en.wikipedia.org/wiki/Electrical_resistance) as a circuit element. When a voltage V is applied across the terminals of a resistor, a current I will flow through the resistor in [direct proportion](http://en.wikipedia.org/wiki/Direct_proportion) to that voltage. The reciprocal of the constant of proportionality is known as the [resistance](http://en.wikipedia.org/wiki/Resistance) R, since, with a given voltage V, a larger value of R further "resists" the flow of current I as given by [Ohm's law](http://en.wikipedia.org/wiki/Ohm%27s_law):

*I = {V \over R}*

Resistors are common elements of [electrical networks](http://en.wikipedia.org/wiki/Electrical_networks) and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as [resistance wire](http://en.wikipedia.org/wiki/Resistance_wire) (wire made of a high-resistivity alloy, such as nickel-chrome). Resistors are also implemented within [integrated circuits](http://en.wikipedia.org/wiki/Integrated_circuits), particularly analog devices, and can also be integrated into [hybrid](http://en.wikipedia.org/wiki/Hybrid_circuit) and [printed circuits](http://en.wikipedia.org/wiki/Printed_circuit_board).

The electrical functionality of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than 9 [orders of magnitude](http://en.wikipedia.org/wiki/Orders_of_magnitude). When specifying that resistance in an electronic design, the required precision of the resistance may require attention to the [manufacturing tolerance](http://en.wikipedia.org/wiki/Engineering_tolerance#Electrical_component_tolerance) of the chosen resistor, according to its specific application. The [temperature coefficient](http://en.wikipedia.org/wiki/Temperature_coefficient) of the resistance may also be of concern in some precision applications. Practical resistors are also specified as having a maximum [power](http://en.wikipedia.org/wiki/Power_%28physics%29) rating which must exceed the anticipated power dissipation of that resistor in a particular circuit: this is mainly of concern in power electronics applications. Resistors with higher power ratings are physically larger and may require [heat sinking](http://en.wikipedia.org/wiki/Heat_sink). In a high voltage circuit, attention must sometimes be paid to the rated maximum working voltage of the resistor.

The series [inductance](http://en.wikipedia.org/wiki/Inductance) of a practical resistor causes its behaviour to depart from ohms law; this specification can be important in some high-frequency applications for smaller values of resistance. In a [low-noise amplifier](http://en.wikipedia.org/wiki/Low-noise_amplifier) or [pre-amp](http://en.wikipedia.org/wiki/Pre-amp) the noise characteristics of a resistor may be an issue. The unwanted inductance, excess noise, and temperature coefficient are mainly dependent on the technology used in manufacturing the resistor. They are not normally specified individually for a particular family of resistors manufactured using a particular technology. A family of discrete resistors is also characterized according to its form factor, that is, the size of the device and position of its leads (or terminals) which is relevant in the practical manufacturing of circuits using them.

## Units

The [ohm](http://en.wikipedia.org/wiki/Ohm_%28unit%29) (symbol: [Ω](http://en.wikipedia.org/wiki/%CE%A9)) is the [SI](http://en.wikipedia.org/wiki/SI) unit of [electrical resistance](http://en.wikipedia.org/wiki/Electrical_resistance), named after [Georg Simon Ohm](http://en.wikipedia.org/wiki/Georg_Simon_Ohm). An ohm is equivalent to a [volt](http://en.wikipedia.org/wiki/Volt) per [ampere](http://en.wikipedia.org/wiki/Ampere). Since resistors are specified and manufactured over a very large range of values, the derived units of milliohm (1 mΩ = 10−3 Ω), kilohm (1 kΩ = 103 Ω), and megohm (1 MΩ = 106 Ω) are also in common usage.

The reciprocal of resistance R is called [conductance](http://en.wikipedia.org/wiki/Conductance) G = 1/R and is measured in [Siemens](http://en.wikipedia.org/wiki/Siemens_%28unit%29) ([SI](http://en.wikipedia.org/wiki/SI) unit), sometimes referred to as a [mho](http://en.wikipedia.org/wiki/Mho). Thus a Siemens is the reciprocal of an ohm: *S* = Ω − 1. Although the concept of conductance is often used in circuit analysis, practical resistors are always specified in terms of their resistance (ohms) rather than conductance.

## Theory of operation

### Ohm's law

The behaviour of an ideal resistor is dictated by the relationship specified in [Ohm's law](http://en.wikipedia.org/wiki/Ohm%27s_law):

V=I \cdot R

Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) passing through it, where the constant of proportionality is the resistance (R).

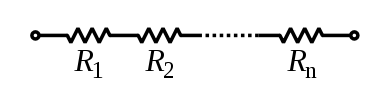
Equivalently, Ohm's law can be stated:

I = \frac{V}{R}

This formulation of Ohm's law states that, when a voltage (V) is present across a resistance (R), a current (I) will flow through the resistance. This is directly used in practical computations. For example, if a 300 [ohm](http://en.wikipedia.org/wiki/Ohm) resistor is attached across the terminals of a 12 [volt](http://en.wikipedia.org/wiki/Volt) battery, then a current of 12 / 300 = 0.04 [amperes](http://en.wikipedia.org/wiki/Amperes) (or 40 mill amperes) will flow through that resistor.

### Series and parallel resistors

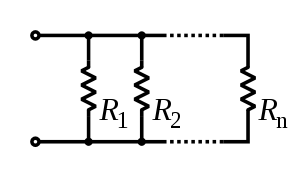
In a [series](http://en.wikipedia.org/wiki/Series_and_parallel_circuits) configuration, the current through all of the resistors is the same, but the voltage across each resistor will be in proportion to its resistance. The potential difference (voltage) seen across the network is the sum of those voltages, thus the total resistance can be found as the sum of those resistances:

[](http://en.wikipedia.org/wiki/File:Resistors_in_series.svg)

R_\mathrm{eq} = R_1  + R_2 + \cdots + R_n

As a special case, the resistance of N resistors connected in series, each of the same resistance R, is given by NR.

Resistors in a [parallel](http://en.wikipedia.org/wiki/Series_and_parallel_circuits) configuration are each subject to the same potential difference (voltage), however the currents through them add. The [conductance](http://en.wikipedia.org/wiki/Conductance) of the resistors then add to determine the conductance of the network. Thus the equivalent resistance (*Req*) of the network can be computed:

[](http://en.wikipedia.org/wiki/File:Resistors_in_parallel.svg)

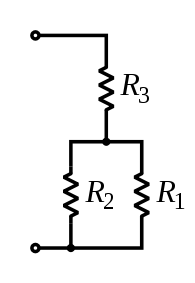
\frac{1}{R_\mathrm{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots +  \frac{1}{R_n}

The parallel equivalent resistance can be represented in equations by two vertical lines "||" (as in geometry) as a simplified notation. For the case of two resistors in parallel, this can be calculated using:

R_\mathrm{eq} = R_1 \| R_2 = {R_1 R_2 \over R_1 + R_2}

As a special case, the resistance of N resistors connected in parallel, each of the same resistance R, is given by R/N.

A resistor network that is a combination of parallel and series connections can be broken up into smaller parts that are either one or the other. For instance,

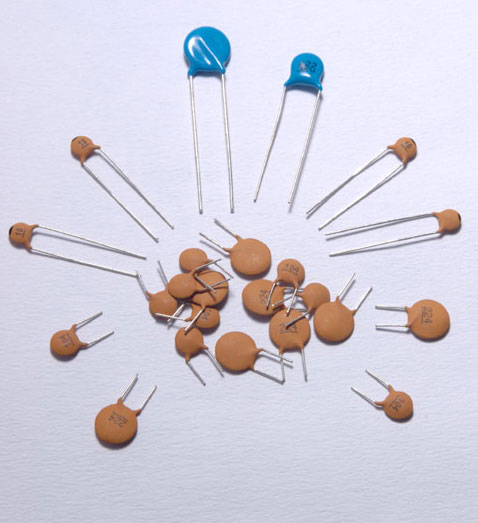
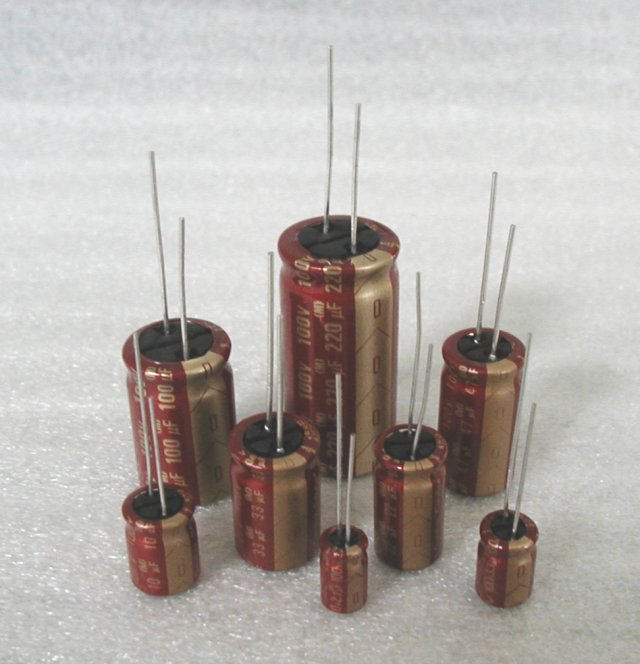
[](http://en.wikipedia.org/wiki/File:Resistors_in_series_and_parallel.svg)

R_\mathrm{eq} = \left( R_1 \| R_2 \right) + R_3 = {R_1 R_2 \over R_1 + R_2} + R_3

However, some complex networks of resistors cannot be resolved in this manner, requiring more sophisticated circuit analysis. For instance, consider a [cube](http://en.wikipedia.org/wiki/Cube), each edge of which has been replaced by a resistor. What then is the resistance that would be measured between two opposite vertices? In the case of 12 equivalent resistors, it can be shown that the corner-to-corner resistance is 5⁄6 of the individual resistance. More generally, the [Y-Δ transform](http://en.wikipedia.org/wiki/Y-%CE%94_transform), or [matrix methods](http://en.wikipedia.org/wiki/Equivalent_impedance_transforms#2-terminal.2C_n-element.2C_3-element-kind_networks) can be used to solve such a problem. One practical application of these relationships is that a non-standard value of resistance can generally be synthesized by connecting a number of standard values in series and/or parallel. This can also be used to obtain a resistance with a higher power rating than that of the individual resistors used. In the special case of N identical resistors all connected in series or all connected in parallel, the power rating of the individual resistors is thereby multiplied by N.

**4.13 CAPACITORS**

A capacitor or condenser is a passive electronic component consisting of a pair of conductors separated by a dielectric. When a voltage potential difference exists between the conductors, an electric field is present in the dielectric. This field stores energy and produces a mechanical force between the plates. The effect is greatest between wide, flat, parallel, narrowly separated conductors.

****

An ideal capacitor is characterized by a single constant value, capacitance, which is measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In practice, the dielectric between the plates passes a small amount of leakage current. The conductors and leads introduce an equivalent series resistance and the dielectric has an electric field strength limit resulting in a breakdown voltage.

The properties of capacitors in a circuit may determine the resonant frequency and quality factor of a resonant circuit, power dissipation and operating frequency in a digital logic circuit, energy capacity in a high-power system, and many other important aspects.

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

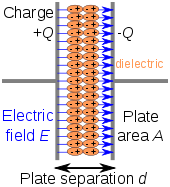
Capacitors are widely used in electronic circuits for blocking [direct current](http://en.wikipedia.org/wiki/Direct_current) while allowing [alternating current](http://en.wikipedia.org/wiki/Alternating_current) to pass, in filter networks, for smoothing the output of [power supplies](http://en.wikipedia.org/wiki/Power_supply), in the [resonant circuits](http://en.wikipedia.org/wiki/LC_circuit) that tune radios to particular [frequencies](http://en.wikipedia.org/wiki/Frequency) and for many other purposes.

A capacitor is a [passive](http://en.wikipedia.org/wiki/Passivity_%28engineering%29) [electronic component](http://en.wikipedia.org/wiki/Electronic_component) consisting of a pair of [conductors](http://en.wikipedia.org/wiki/Electrical_conductor) separated by a [dielectric](http://en.wikipedia.org/wiki/Dielectric) (insulator). When there is a [potential difference](http://en.wikipedia.org/wiki/Potential_difference) (voltage) across the conductors, a static [electric field](http://en.wikipedia.org/wiki/Electric_field) develops in the dielectric that stores [energy](http://en.wikipedia.org/wiki/Energy) and produces a mechanical force between the conductors. An ideal capacitor is characterized by a single constant value, [capacitance](http://en.wikipedia.org/wiki/Capacitance), measured in [farads](http://en.wikipedia.org/wiki/Farad). This is the ratio of the [electric charge](http://en.wikipedia.org/wiki/Electric_charge) on each conductor to the potential difference between them.

The capacitance is greatest when there is a narrow separation between large areas of conductor, hence capacitor conductors are often called "plates", referring to an early means of construction. In practice the dielectric between the plates passes a small amount of [leakage current](http://en.wikipedia.org/wiki/Leakage_%28electronics%29) and also has an electric field strength limit, resulting in a [breakdown voltage](http://en.wikipedia.org/wiki/Breakdown_voltage), while the conductors and [leads](http://en.wikipedia.org/wiki/Lead_%28electronics%29) introduce an undesired [inductance](http://en.wikipedia.org/wiki/Equivalent_series_inductance) and [resistance](http://en.wikipedia.org/wiki/Equivalent_series_resistance).

## Theory of operation

[Capacitance](http://en.wikipedia.org/wiki/Capacitance)

[](http://en.wikipedia.org/wiki/File:Capacitor_schematic_with_dielectric.svg)

Charge separation in a parallel-plate capacitor causes an internal electric field. A dielectric (orange) reduces the field and increases the capacitance.

A capacitor consists of two [conductors](http://en.wikipedia.org/wiki/Electrical_conductor) separated by a non-conductive region. The non-conductive region is called the dielectric or sometimes the [dielectric medium](http://en.wikipedia.org/wiki/Dielectric_medium). In simpler terms, the dielectric is just an [electrical insulator](http://en.wikipedia.org/wiki/Insulator_%28electrical%29). Examples of dielectric mediums are glass, air, paper, [vacuum](http://en.wikipedia.org/wiki/Vacuum), and even a [semiconductor](http://en.wikipedia.org/wiki/Semiconductor) [depletion region](http://en.wikipedia.org/wiki/Depletion_region) chemically identical to the conductors. A capacitor is assumed to be self-contained and isolated, with no net [electric charge](http://en.wikipedia.org/wiki/Electric_charge) and no influence from any external electric field. The conductors thus hold equal and opposite charges on their facing surfaces, and the dielectric develops an electric field. In [SI](http://en.wikipedia.org/wiki/SI) units, a capacitance of one [farad](http://en.wikipedia.org/wiki/Farad) means that one [coulomb](http://en.wikipedia.org/wiki/Coulomb) of charge on each conductor causes a voltage of one [volt](http://en.wikipedia.org/wiki/Volt) across the device.

The capacitor is a reasonably general model for electric fields within electric circuits. An ideal capacitor is wholly characterized by a constant capacitance C, defined as the ratio of charge ±Q on each conductor to the voltage V between them:

C= \frac{Q}{V}

Sometimes charge build-up affects the capacitor mechanically, causing its capacitance to vary. In this case, capacitance is defined in terms of incremental changes:

C= \frac{\mathrm{d}q}{\mathrm{d}v}

### Energy storage

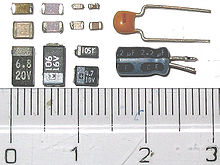
[Work](http://en.wikipedia.org/wiki/Work_%28thermodynamics%29) must be done by an external influence to "move" charge between the conductors in a capacitor. When the external influence is removed the charge separation persists in the electric field and energy is stored to be released when the charge is allowed to return to its [equilibrium](http://en.wikipedia.org/wiki/Equilibrium) position. The work done in establishing the electric field, and hence the amount of energy stored, is given by:

W= \int_{q=0}^Q V \text{d}q = \int_{q=0}^Q \frac{q}{C} \text{d}q = {1 \over 2} {Q^2 \over C} = {1 \over 2}  C V^2 = {1 \over 2} VQ.

### Current-voltage relation

The current i(t) through any component in an electric circuit is defined as the rate of flow of a charge q(t) passing through it, but actual charges, electrons, cannot pass through the dielectric layer of a capacitor, rather an electron accumulates on the negative plate for each one that leaves the positive plate, resulting in an electron depletion and consequent positive charge on one electrode that is equal and opposite to the accumulated negative charge on the other. Thus the charge on the electrodes is equal to the [integral](http://en.wikipedia.org/wiki/Integral) of the current as well as proportional to the voltage as discussed above. As with any [anti-derivative](http://en.wikipedia.org/wiki/Antiderivative), a [constant of integration](http://en.wikipedia.org/wiki/Constant_of_integration) is added to represent the initial voltage v (t0). This is the integral form of the capacitor equation,

[](http://en.wikipedia.org/wiki/File:Condensators.JPG)v(t)= \frac{q(t)}{C} = \frac{1}{C}\int_{t_0}^t i(\tau) \mathrm{d}\tau+v(t_0).

[](http://en.wikipedia.org/wiki/File:Photo-SMDcapacitors.jpg)Taking the derivative of this, and multiplying by C, yields the derivative form,

i(t)= \frac{\mathrm{d}q(t)}{\mathrm{d}t}=C\frac{\mathrm{d}v(t)}{\mathrm{d}t}.

The [dual](http://en.wikipedia.org/wiki/Duality_%28electrical_circuits%29) of the capacitor is the [inductor](http://en.wikipedia.org/wiki/Inductor), which stores energy in the [magnetic field](http://en.wikipedia.org/wiki/Magnetic_field) rather than the electric field. Its current-voltage relation is obtained by exchanging current and voltage in the capacitor equations and replacing C with the inductance L.

**ULTRASONIC SENSOR MODULE**

****

Here is a more easy use serial ultrasonic module. It will auto output the distance information via serial port after power on, you don't need to do any trigger and calculated, just need to read the serial pin and get the distance information.

Ultrasonic sensor provides a very low-cost and easy method of distance measurement. This sensor is perfect for any number of applications that require you to perform measurements between moving or stationary objects. Naturally, robotics applications are very popular but you'll also find this product to be useful in security systems or as an infrared replacement if so desired. You will definitely appreciate the activity status LED and the economic use of just one I/O pin.

The ultrasonic sensor measures distance using sonar; an ultrasonic (well above human hearing) pulse is transmitted from the unit and distance-to-target is determined by measuring the time required for the echo return. Output from the ultrasonic sensor is a variable-width pulse that corresponds to the distance to the target.

**Features:**

* Provides precise, non-contact distance measurements within a 2 cm to 3 m range
* Simple pulse in/pulse out communication
* Burst indicator LED shows measurement in progress
* 20 mA power consumption
* Narrow acceptance angle
* 3-pin header makes it easy to connect using a servo extension cable, no soldering required

**Key Specifications:**

* power supply :5V DC
* quiescent current : <15mA
* effectual angle: <15°
* ranging distance : 2cm – 350 cm
* resolution : 0.3 cm
* Output cycle : 50ms
* Baud Rate : 9600

**Output frame format** （**4Bytes**）

* 1st Byte : 0xFF (Start bit, Fixed value)
* 2nd Byte: H\_Data (High 8 bit of the distance)
* 3rd Byte: L\_Date (Low 8 bit of the distance)
* 4th Byte: Check Sum (0XFF+H\_DATA+L\_DATA=SUM, Check SUM is the low 8 bit of the SUM)

**Uses:**

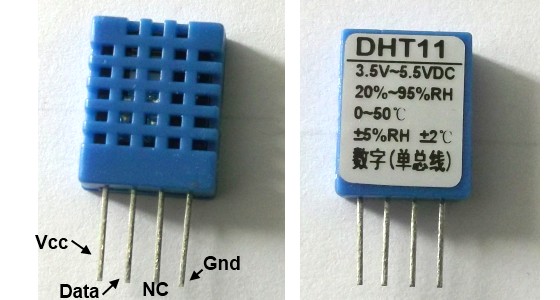
Ultrasonic sensors are used to detect the presence of targets and to measure the distance to targets in many automated factories and process plants. Sensors with an on or off digital output are available for detecting the presence of objects, and sensors with an analog output which varies proportionally to the sensor to target separation distance are commercially available.

Because ultrasonic sensors use sound rather than light for detection, they work in applications where photoelectric sensors may not. Ultrasonic are a great solution for clear object detection and for liquid level measurement, applications that photo electrics struggle with because of target translucence. Target colour and/or reflectivity don't affect ultrasonic sensors which can operate reliably in high-glare environments.

Other types of transducers are used in commercially available ultrasonic cleaning devices. An ultrasonic transducer is affixed to a stainless steel pan which is filled with a solvent (frequently water or isopropanol) and a square wave is applied to it, imparting vibrational energy on the liquid.

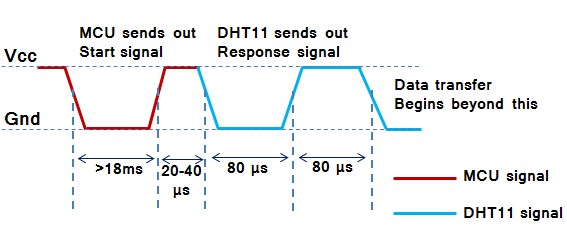
TEMPERATURE AND HUMIDITY SENSOR

The DHT11 sensor comes in a single row 4-pin package and operates from 3.5 to 5.5V power supply. It can measure temperature from 0-50 °C with an accuracy of ±2°C and relative humidity ranging from 20-95% with an accuracy of  ±5%. The sensor provides fully calibrated digital outputs for the two measurements. It has got its own proprietary 1-wire protocol, and therefore, the communication between the sensor and a microcontroller is not possible through a direct interface with any of its peripherals. The protocol must be implemented in the firmware of the MCU with precise timing required by the sensor.

[](http://embedded-lab.com/blog/wp-content/uploads/2012/01/DHT11Sensor1.jpg)

DHT11 sensor comes in a single row 4-pin package

The following timing diagrams describe the data transfer protocol between a MCU and the DHT11 sensor. The MCU initiates data transmission by issuing a “***Start***” signal. The MCU pin must be configured as output for this purpose. The MCU first pulls the data line low for at least 18 ms and then pulls it high for next 20-40 ?s before it releases it. Next, the sensor responds to the MCU “***Start***”  signal by pulling the line low for 80 ?s followed by a logic high signal that also lasts for 80 ?s. Remember that the MCU pin must be configured to input after finishing the “***Start***” signal. Once detecting the response signal from the sensor, the MCU should be ready to receive data from the sensor. The sensor then sends 40 bits (5 bytes) of data continuously in the data line. Note that while transmitting bytes, the sensor sends the most significant bit first.

[](http://embedded-lab.com/blog/wp-content/uploads/2012/01/TimingDiagram1.jpg)

"Start" and "Response" signals

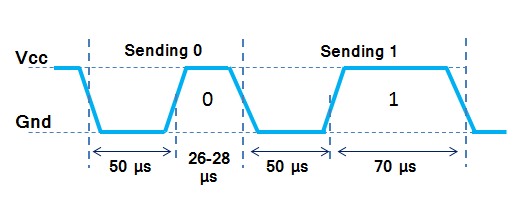
The 40-bit data from the sensor has the following structure.

***Data (40-bit) = Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp. + Checksum Byte***

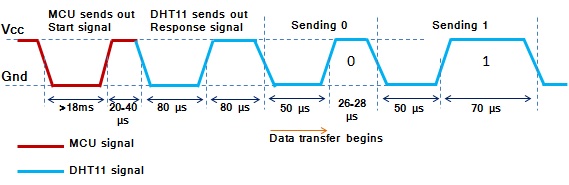
For DHT11 sensor, the decimal bytes of temperature and humidity measurements are always zero. Therefore, the first and third bytes of received data actually give the numeric values of the measured relative humidity (%) and temperature (°C). The last byte is the checksum byte which is used to make sure that the data transfer has happened without any error. If all the five bytes are transferred successfully then the checksum byte must be equal to the last 8 bits of the sum of the first four bytes, i.e.,

***Checksum****=****Last 8 bits of****(Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp.)*

Now lets talk about the most important thing, which is signalling for transmitting “0” and “1”. In order to send a bit of data, the sensor first pulls the line low for 50 ?s. Then it raises the line to high for 26-28 ?s if it has to send “0”, or for 70 ?s if the bit to be transmitted is “1”. So it is the width of the positive pulse that carries information about 1 and 0.

[](http://embedded-lab.com/blog/wp-content/uploads/2012/01/TimingDiagram2.jpg)

Timing difference for transmitting "1s" and "0s"

[](http://embedded-lab.com/blog/wp-content/uploads/2012/01/CombinedTiming.jpg)

Start, Response and Data signals in sequence

At the end of the last transmitted bit, the sensor pulls the data line low for 50 ?s and then releases it. The DHT11 sensor requires an external pull-up resistor to be connected between its Vcc and the data line so that under idle condition, the data line is always pulled high. After finishing the data transmission and releasing the data line, the DHT11 sensor goes to the low-power consumption mode until a new “***Start***” signal arrives from the MCU.

**LIQUID CRYSTAL DISPLAY (LCD)**

**Description:**

This is the example for the Parallel Port. This example doesn't use the Bi-directional feature found on newer ports, thus it should work with most, if not all Parallel Ports. It however doesn't show the use of the Status Port as an input for a 16 Character x 2 Line LCD Module to the Parallel Port. These LCD Modules are very common these days, and are quite simple to work with, as all the logic required running them is on board.

**Pros:**

* Very compact and light
* Low power consumption
* No geometric distortion
* Little or no flicker depending on backlight technology
* Not affected by screen burn-in
* No high voltage or other hazards present during repair/service
* Can be made in almost any size or shape
* No theoretical resolution limit

**LCD Background:**

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

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

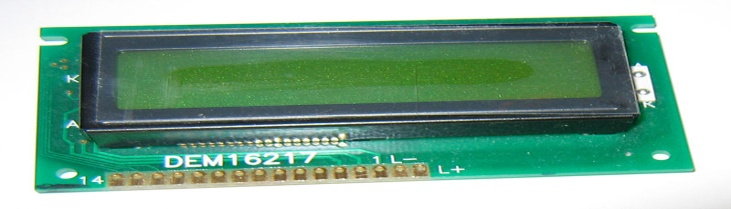
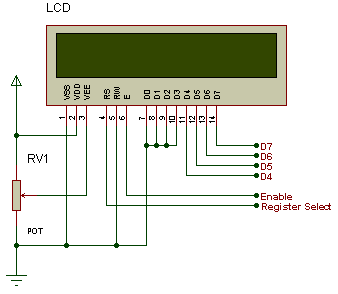
[](http://upload.wikimedia.org/wikipedia/commons/2/24/LCD_display_16x2_alphanumeric.jpg)

FIG 4.10: LCD

**44780 LCD BACKGROUND**

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



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

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

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

The RW line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low .Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

**How LCDs Work?**

The principle behind the LCD’s is that when an electrical current is applied to the liquid crystal molecule, the molecule tends to untwist. This causes the angle of light which is passing through the molecule of the polarized glass and also cause a change in the angle of the top polarizing filter. As a result a little light is allowed to pass the polarized glass through a particular area of the LCD. Thus that particular area will become dark compared to other. The LCD works on the principle of blocking light. While constructing the LCD’s, a reflected mirror is arranged at the back. An electrode plane is made of indium-tin oxide which is kept on top and a polarized glass with a polarizing film is also added on the bottom of the device. The complete region of the LCD has to be enclosed by a common electrode and above it should be the liquid crystal matter.

Next comes to the second piece of glass with an electrode in the form of the rectangle on the bottom and, on top, another polarizing film. It must be considered that both the pieces are kept at right angles. When there is no current, the light passes through the front of the LCD it will be reflected by the mirror and bounced back. As the electrode is connected to a battery the current from it will cause the liquid crystals between the common-plane electrode and the electrode shaped like a rectangle to untwist. Thus the light is blocked from passing through. That particular rectangular area appears blank.

**Advantages of an LCD’s:**

* LCD’s consumes less amount of power compared to CRT and LED
* LCD’s are consist of some microwatts for display in comparison to some mill watts for LED’s
* LCDs are of low cost
* Provides excellent contrast
* LCD’s are thinner and lighter when compared to cathode ray tube and LED

**5. SOFTWARE REQUIREMENTS**

**Software Requirement:**

**Arduino**

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on[Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

### Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

**Arduino Uno**  
  
The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. In the next step, there is a more complete rundown of its features.  
  
**Arduino NG, Diecimila, and the Duemilanove (Legacy Versions)**  
  
Legacy versions of the Arduino Uno product line consist of the NG, Diecimila, and the Duemilanove. The important thing to note about legacy boards is that they lack particular feature of the Arduino Uno. Some key differences:

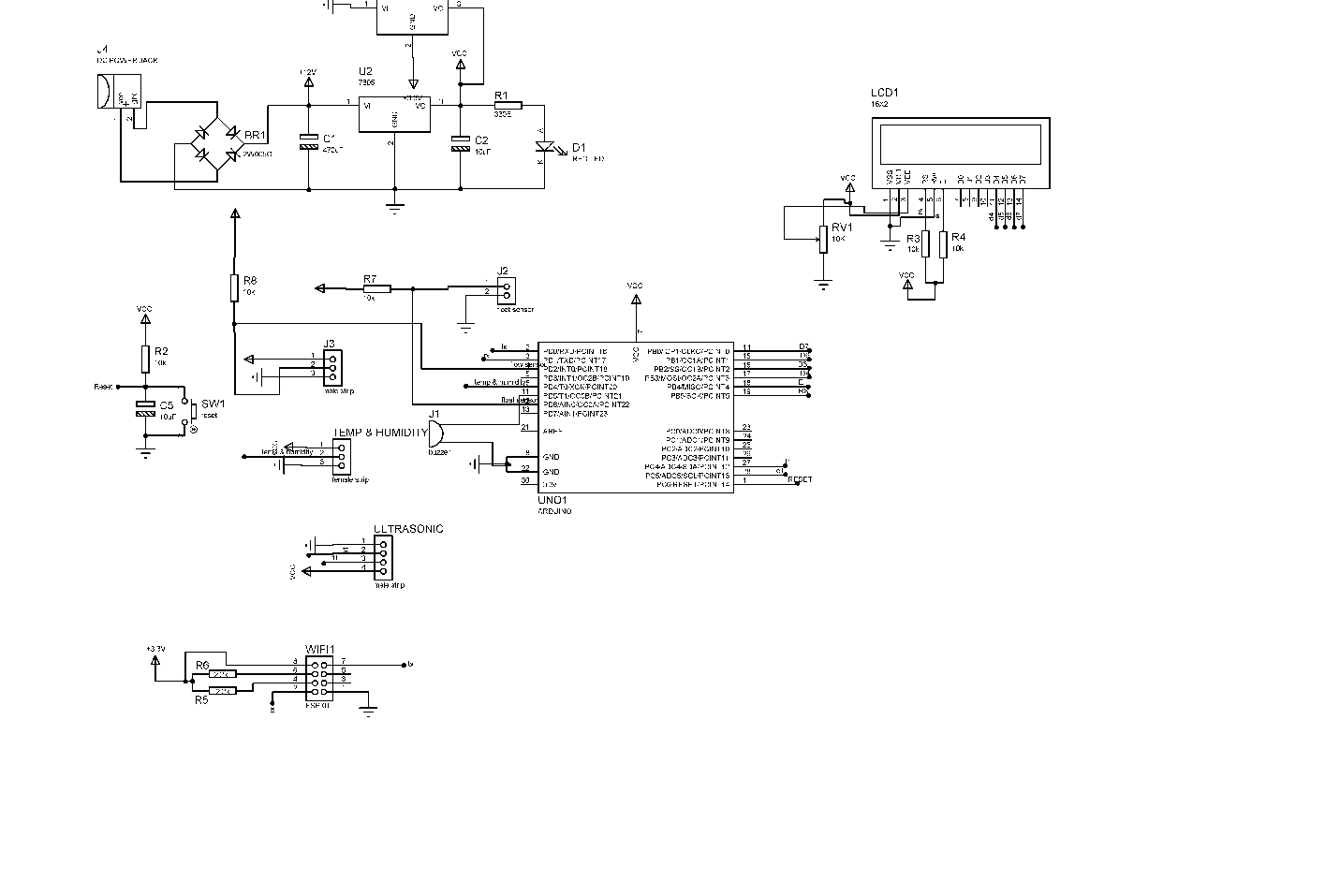
* The Diecimila and NG use an ATMEGA168 chips (as opposed to the more powerful ATMEGA328),
* Both the Diecimila and NG have a jumper next to the USB port and require manual selection of either USB or battery power.
* The Arduino NG requires that you hold the rest button on the board for a few seconds prior to uploading a program.

**Arduino Mega 2560**   
  
The Mega is the second most commonly encountered version of the Arduino family. The Arduino Mega is like the Arduino Uno's beefier older brother. It boasts 256 KB of memory (8 times more than the Uno). It also had 54 input and output pins, 16 of which are analog pins, and 14 of which can do PWM. However, all of the added functionality comes at the cost of a slightly larger circuit board. It may make your project more powerful, but it will also make your project larger. Check out the official [Arduino Mega 2560 page](http://arduino.cc/en/Main/ArduinoBoardMega2560) for more details.  
  
**Arduino Mega ADK**  
  
This specialized version of the Arduino is basically an Arduino Mega that has been specifically designed for interfacing with Android smartphones.  
  
**Arduino LilyPad**  
  
The LilyPad was designed for wearable and e-textile applications. It is intended to be sewn to fabric and connected to other sewable components using conductive thread. This board requires the use of a special [FTDI-USB TTL serial programming cable](http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm). For more information, the [Arduino LilyPad page](http://arduino.cc/en/Guide/ArduinoLilyPad) is a decent starting point.

Some people think of the entire Arduino board as a microcontroller, but this is inaccurate. The Arduino board actually is a specially designed circuit board for programming and prototyping with Atmel microcontrollers.  
  
The nice thing about the Arduino board is that it is relatively cheap, plugs straight into a computer's USB port, and it is dead-simple to setup and use (compared to other development boards).  
  
Some of the key features of the Arduino Uno include:

* An open source design. The advantage of it being open source is that it has a [large community](http://arduino.cc/forum/) of people using and troubleshooting it. This makes it easy to find someone to help you debug your projects.
* An easy USB interface . The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.
* Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.
* An easy-to-find, and dirt cheap, microcontroller "brain." The ATmega328 chip retails for about $2.88 on Digikey. It has countless number of nice hardware features like timers, PWM pins, external and internal interrupts, and multiple sleep modes. Check out the official [datasheet](http://www.atmel.com/Images/8271s.pdf) for more details.
* A 16mhz clock. This makes it not the speediest microcontroller around, but fast enough for most applications.
* 32 KB of flash memory for storing your code.
* 13 digital pins and 6 analog pins. These pins allow you to connect external hardware to your Arduino. These pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these pins and you are good to go.
* An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to [re-bootload your chip](http://www.instructables.com/id/Bootload-an-Arduino-with-a-ZIF-Socket/) if it corrupts and can no longer talk to your computer.
* An on-board LED attached to digital pin 13 for fast an easy debugging of code.
* And last, but not least, a button to reset the program on the chip.

**7. LAYOUT DIAGRAM**



**9. CODING**

#include <iotgecko.h>

#include <LiquidCrystal.h>

#include "DHT.h"

#define esp\_baudrate 9600

///////variables////////////////

String id = "tejas.chandegara3@gmail.com"; // iotgecko login id

String pass = "9583"; // iotgecko login password

String ssid = "iot"; // SSID/name of your wifi router or wifi hotspot

String pass\_key = "project1234"; // Wifi password

bool notConected = true;

bool login = false;

const int no\_of\_senddata = 5;

const int no\_of\_getdata = 0;

String iot\_data[no\_of\_senddata];

int motor\_status[no\_of\_getdata];

int pump\_status = 0;

unsigned long current\_time ;

iotgecko gecko = iotgecko(esp\_baudrate);

const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#define trigPin A4

#define echoPin A5

#define DHTPIN 4

#define Water\_Level\_Sensor 6

#define buzzer 5

int buttonState = 0; // variable for reading the pushbutton status

long duration;

int distance;

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

volatile int flow\_frequency; // Measures flow sensor pulses

unsigned int l\_hour; // Calculated litres/hour

unsigned char flowsensor = 2; // Sensor Input

unsigned long currentTime;

unsigned long cloopTime;

int humidity, temperature;

bool water\_level;

void flow () // Interrupt function

{

flow\_frequency++;

}

void setup() {

lcd.begin (20, 4);

lcd.clear();

//lcd.setCursor(0,0);

lcd.print(" FLOOD DETECTION");

// lcd.clear();

lcd.setCursor(0, 1);

lcd.print(" SYSTEM");

// lcd.clear();

delay(2000);

lcd.clear();

lcd.print(F("Connecting Wifi"));

gecko.listen();

while (notConected)

{

if (gecko.GeckoConnect(ssid, pass\_key)) //connect to wifi with given SSID and password

{

while (!login)

{

if (gecko.GeckoVerify(id, pass)) //login to iotgecko.com with given ID and password

{

lcd.clear();

lcd.print(F("Connected"));

login = true;

notConected = false;

}

}

}

delay(1000);

lcd.clear();

// pinMode (Led, OUTPUT);

pinMode (trigPin, OUTPUT);

pinMode (echoPin, INPUT);

dht.begin();

pinMode(Water\_Level\_Sensor, INPUT);

// pinMode(LED\_BUILTIN, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(flowsensor, INPUT);

digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up

attachInterrupt(0, flow, RISING); // Setup Interrupt

sei(); // Enable interrupts

currentTime = millis();

cloopTime = currentTime;

Serial.begin (9600);

}

}

void loop()

{

loop1();

//delay(500);

loop2();

//delay(500);

loop3();

//delay(500);

loop4();

delay(2000);

//iot\_data[0] =(String)2;

iot\_data[0] =(String)temperature;

iot\_data[1] = (String)humidity;

iot\_data[2] = (String)distance;

iot\_data[3] = (String)l\_hour;

iot\_data[4] = (String)water\_level;

int iot\_status = gecko.SendandGetGParams(iot\_data,no\_of\_senddata, motor\_status , no\_of\_getdata);//send and get data on iotgecko.com

if(iot\_status == VALID)

{

}

else

{

lcd.clear();

lcd.print(F("reconnecting"));

while(!gecko.GeckoReconnect())//reconnect to the iotgecko.com

{

//Serial.println(F("conncetion failed.....reconnecting"));

delay(2000);

}

lcd.clear();

lcd.print("Connected");

//Serial.println(F("connected succesfully"));

}

delay(3000);

}

/\*else

{

lcd.clear();

lcd.setCursor(5,0);

lcd.print("FLOOD DETECTION");

lcd.clear();

lcd.setCursor(8,1);

lcd.print("SYSTEM");

}

}\*/

void loop1()

{

digitalWrite (trigPin, LOW);

delayMicroseconds (2);

digitalWrite (trigPin, HIGH);

delayMicroseconds (10);

digitalWrite (trigPin, LOW);

duration = pulseIn (echoPin, HIGH);

distance = duration / 58 ;

Serial.print ("Distance :");

Serial.print (distance);

Serial.println (" cm ");

delay (120);

// lcd.clear();

lcd.setCursor(0, 0); // Sets the location at which subsequent text written to the LCD will be displayed

lcd.print("Distance :"); // Prints string "Distance" on the LCD

lcd.print(distance); // Prints the distance value from the sensor

lcd.print(" cm ");

//lcd.clear();

delay (120);

}

void loop2()

{

delay(2000);

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

int h = dht.readHumidity();

// Read temperature as Celsius (the default)

int t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

//float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Compute heat index in Fahrenheit (the default)

// float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

//float hic = dht.computeHeatIndex(t, h, false);

Serial.println("Humidity: ");

Serial.print(h);

Serial.println(" %\t");

Serial.print("Temperature: ");

Serial.println(t);

Serial.println(" \*C ");

// lcd.clear();

humidity = h;

temperature = t;

lcd.setCursor(0, 1);

lcd.print(" HUM:");

lcd.print(h);

lcd.print("%");

lcd.print(" Temp:");

lcd.print(t);

lcd.print("\*C");

delay(120);

//Serial.print(f);

// Serial.print(" \*F\t");

// Serial.print("Heat index: ");

// Serial.print(hic);

// Serial.print(" \*C ");

// Serial.print(hif);

// Serial.println(" \*F");

}

void loop3()

{

buttonState = digitalRead(Water\_Level\_Sensor);

if (buttonState == LOW) {

Serial.println( "WATER LEVEL - LOW");

// digitalWrite(LED\_BUILTIN, LOW); // turn the LED on (HIGH is the voltage level)

digitalWrite(buzzer, LOW);

// lcd.clear();

lcd.setCursor(0, 3);

lcd.print("Water Level: LOW");

water\_level = 0;

delay(500);

}

else {

Serial.println( "WATER LEVEL - HIGH" );

//digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

digitalWrite(buzzer, HIGH);

// lcd.clear();

lcd.setCursor(0, 3);

lcd.print("Water Level:HIGH");

water\_level=1;

// lcd.print("Floa

delay(500);

}

}

void loop4()

{

currentTime = millis();

// Every second, calculate and print litres/hour

if (currentTime >= (cloopTime + 1000))

{

cloopTime = currentTime; // Updates cloopTime

// Pulse frequency (Hz) = 7.5Q, Q is flow rate in L/min.

l\_hour = (flow\_frequency \* 60 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

flow\_frequency = 0; // Reset Counter

Serial.print(l\_hour, DEC); // Print litres/hour

Serial.println(" L/hour");

lcd.setCursor(0, 2);

lcd.print("Flow Level :");

lcd.print(l\_hour, DEC);

lcd.print(" L/H ");

}

}#include <iotgecko.h>

#include <LiquidCrystal.h>

#include "DHT.h"

#define esp\_baudrate 9600

///////variables////////////////

String id = "tejas.chandegara3@gmail.com"; // iotgecko login id

String pass = "9583"; // iotgecko login password

String ssid = "iot"; // SSID/name of your wifi router or wifi hotspot

String pass\_key = "project1234"; // Wifi password

bool notConected = true;

bool login = false;

const int no\_of\_senddata = 5;

const int no\_of\_getdata = 0;

String iot\_data[no\_of\_senddata];

int motor\_status[no\_of\_getdata];

int pump\_status = 0;

unsigned long current\_time ;

iotgecko gecko = iotgecko(esp\_baudrate);

const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#define trigPin A4

#define echoPin A5

#define DHTPIN 4

#define Water\_Level\_Sensor 6

#define buzzer 5

int buttonState = 0; // variable for reading the pushbutton status

long duration;

int distance;

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

volatile int flow\_frequency; // Measures flow sensor pulses

unsigned int l\_hour; // Calculated litres/hour

unsigned char flowsensor = 2; // Sensor Input

unsigned long currentTime;

unsigned long cloopTime;

int humidity, temperature;

bool water\_level;

void flow () // Interrupt function

{

flow\_frequency++;

}

void setup() {

lcd.begin (20, 4);

lcd.clear();

//lcd.setCursor(0,0);

lcd.print(" FLOOD DETECTION");

// lcd.clear();

lcd.setCursor(0, 1);

lcd.print(" SYSTEM");

// lcd.clear();

delay(2000);

lcd.clear();

lcd.print(F("Connecting Wifi"));

gecko.listen();

while (notConected)

{

if (gecko.GeckoConnect(ssid, pass\_key)) //connect to wifi with given SSID and password

{

while (!login)

{

if (gecko.GeckoVerify(id, pass)) //login to iotgecko.com with given ID and password

{

lcd.clear();

lcd.print(F("Connected"));

login = true;

notConected = false;

}

}

}

delay(1000);

lcd.clear();

// pinMode (Led, OUTPUT);

pinMode (trigPin, OUTPUT);

pinMode (echoPin, INPUT);

dht.begin();

pinMode(Water\_Level\_Sensor, INPUT);

// pinMode(LED\_BUILTIN, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(flowsensor, INPUT);

digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up

attachInterrupt(0, flow, RISING); // Setup Interrupt

sei(); // Enable interrupts

currentTime = millis();

cloopTime = currentTime;

Serial.begin (9600);

}

}

void loop()

{

loop1();

//delay(500);

loop2();

//delay(500);

loop3();

//delay(500);

loop4();

delay(2000);

//iot\_data[0] =(String)2;

iot\_data[0] =(String)temperature;

iot\_data[1] = (String)humidity;

iot\_data[2] = (String)distance;

iot\_data[3] = (String)l\_hour;

iot\_data[4] = (String)water\_level;

int iot\_status = gecko.SendandGetGParams(iot\_data,no\_of\_senddata, motor\_status , no\_of\_getdata);//send and get data on iotgecko.com

if(iot\_status == VALID)

{

}

else

{

lcd.clear();

lcd.print(F("reconnecting"));

while(!gecko.GeckoReconnect())//reconnect to the iotgecko.com

{

//Serial.println(F("conncetion failed.....reconnecting"));

delay(2000);

}

lcd.clear();

lcd.print("Connected");

//Serial.println(F("connected succesfully"));

}

delay(3000);

}

/\*else

{

lcd.clear();

lcd.setCursor(5,0);

lcd.print("FLOOD DETECTION");

lcd.clear();

lcd.setCursor(8,1);

lcd.print("SYSTEM");

}

}\*/

void loop1()

{

digitalWrite (trigPin, LOW);

delayMicroseconds (2);

digitalWrite (trigPin, HIGH);

delayMicroseconds (10);

digitalWrite (trigPin, LOW);

duration = pulseIn (echoPin, HIGH);

distance = duration / 58 ;

Serial.print ("Distance :");

Serial.print (distance);

Serial.println (" cm ");

delay (120);

// lcd.clear();

lcd.setCursor(0, 0); // Sets the location at which subsequent text written to the LCD will be displayed

lcd.print("Distance :"); // Prints string "Distance" on the LCD

lcd.print(distance); // Prints the distance value from the sensor

lcd.print(" cm ");

//lcd.clear();

delay (120);

}

void loop2()

{

delay(2000);

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

int h = dht.readHumidity();

// Read temperature as Celsius (the default)

int t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

//float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Compute heat index in Fahrenheit (the default)

// float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

//float hic = dht.computeHeatIndex(t, h, false);

Serial.println("Humidity: ");

Serial.print(h);

Serial.println(" %\t");

Serial.print("Temperature: ");

Serial.println(t);

Serial.println(" \*C ");

// lcd.clear();

humidity = h;

temperature = t;

lcd.setCursor(0, 1);

lcd.print(" HUM:");

lcd.print(h);

lcd.print("%");

lcd.print(" Temp:");

lcd.print(t);

lcd.print("\*C");

delay(120);

//Serial.print(f);

// Serial.print(" \*F\t");

// Serial.print("Heat index: ");

// Serial.print(hic);

// Serial.print(" \*C ");

// Serial.print(hif);

// Serial.println(" \*F");

}

void loop3()

{

buttonState = digitalRead(Water\_Level\_Sensor);

if (buttonState == LOW) {

Serial.println( "WATER LEVEL - LOW");

// digitalWrite(LED\_BUILTIN, LOW); // turn the LED on (HIGH is the voltage level)

digitalWrite(buzzer, LOW);

// lcd.clear();

lcd.setCursor(0, 3);

lcd.print("Water Level: LOW");

water\_level = 0;

delay(500);

}

else {

Serial.println( "WATER LEVEL - HIGH" );

//digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

digitalWrite(buzzer, HIGH);

// lcd.clear();

lcd.setCursor(0, 3);

lcd.print("Water Level:HIGH");

water\_level=1;

// lcd.print("Floa

delay(500);

}

}

void loop4()

{

currentTime = millis();

// Every second, calculate and print litres/hour

if (currentTime >= (cloopTime + 1000))

{

cloopTime = currentTime; // Updates cloopTime

// Pulse frequency (Hz) = 7.5Q, Q is flow rate in L/min.

l\_hour = (flow\_frequency \* 60 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

flow\_frequency = 0; // Reset Counter

Serial.print(l\_hour, DEC); // Print litres/hour

Serial.println(" L/hour");

lcd.setCursor(0, 2);

lcd.print("Flow Level :");

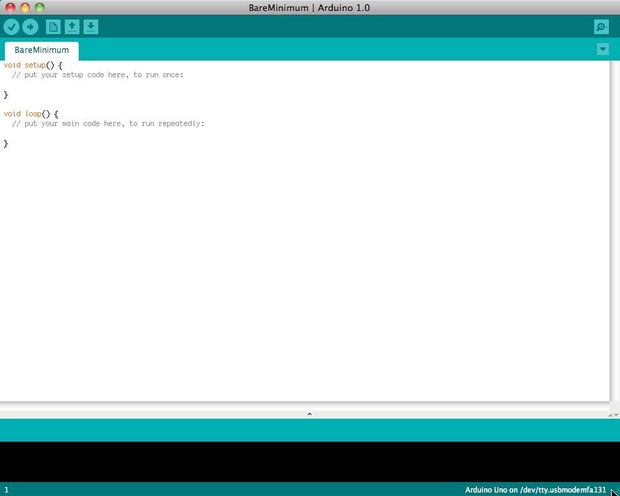
lcd.print(l\_hour, DEC);

lcd.print(" L/H ");

}

}

**Arduino IDE**



Before you can start doing anything with the Arduino, you need to download and install the [Arduino IDE](http://www.arduino.cc/en/Main/software) (integrated development environment). From this point on we will be referring to the Arduino IDE as the Arduino Programmer.  
  
The Arduino Programmer is based on the [Processing IDE](http://processing.org/) and uses a variation of the C and C++ programming languages.

## Settings

## Picture of Settings

## cap024.jpg

Before you can start doing anything in the Arduino programmer, you must set the board-type and serial port.  
  
To set the board, go to the following:

Tools --> Boards

Select the version of board that you are using. Since I have an Arduino Uno  plugged in, I obviously selected "Arduino Uno."  
  
To set the serial port, go to the following:

Tools --> Serial Port

Select the serial port that looks like:

/dev/tty.usbmodem [random numbers]

## Picture of Run a sketch

## 6B.jpg

## 6C.jpg

Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.  
  
To get the LED tied to digital pin 13 to blink on and off, let's load the blink example.  
  
The blink example can be found here:

Files --> Examples --> Basics --> Blink

## The blink example basically sets pin D13 as an output and then blinks the test LED on the Arduino board on and off every second. Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing the upload button, which looks like an arrow pointing to the right. Notice that the surface mount status LED connected to pin 13 on the Arduino will start to blink. You can change the rate of the blinking by changing the length of the delay and pressing the upload button again.

## Serial monitor

## Picture of Serial monitor

## cap025.jpg

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displays it in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.  
  
For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:

File --> Examples --> 1.Basics --> AnalogReadSerial

## Click the button to engage the serial monitor which looks like a magnifying glass.  You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.  The numbers will be between the range of 0 and 1023. The reason for this is that the analog pin is converting a voltage between 0 and 5V to a discreet number.

The Arduino has two different types of input pins, those being analog and digital.  
  
To begin with, lets look at the digital input pins.  
  
Digital input pins only have two possible states, which are on or off. These two on and off states are also referred to as:

* HIGH or LOW
* 1 or 0
* 5V or 0V.

This input is commonly used to sense the presence of voltage when a switch is opened or closed.  
  
Digital inputs can also be used as the basis for countless digital communication protocols. By creating a 5V (HIGH) pulse or 0V (LOW) pulse, you can create a binary signal, the basis of all computing. This is useful for talking to digital sensors like a PING ultrasonic sensor, or communicating with other devices.  
  
For a simple example of a digital input in use, connect a switch from digital pin 2 to 5V, a 10K resistor\*\* from digital pin 2 to ground, and run the following code:

File --> Examples --> 2.Digital --> Button

## Analog in

## Aside from the digital input pins, the Arduino also boasts a number of analog input pins. Analog input pins take an analog signal and perform a 10-bit analog-to-digital (ADC) conversion to turn it into a number between 0 and 1023 (4.9mV steps).  This type of input is good for reading resistive sensors. These are basically sensors which provide resistance to the circuit. They are also good for reading a varying voltage signal between 0 and 5V. This is useful when interfacing with various types of analog circuitry.

## Picture of Write your own code

To write your own code, you will need to learn some basic programming language syntax. In other words, you have to learn how to properly form the code for the programmer to understand it. You can think of this kind of like understanding grammar and punctuation. You can write an entire book without proper grammar and punctuation, but no one will be abler to understand it, even if it is in English.

Some important things to keep in mind when writing your own code:

* An Arduino program is called a sketch.
* All code in an Arduino sketch is processed from top to bottom.
* Arduino sketches are typically broken into five parts.

1. The sketch usually starts with a header that explains what the sketch is doing, and who wrote it.
2. Next, it usually defines global variables. Often, this is where constant names are given to the different Arduino pins.
3. After the initial variables are set, the Arduino begins the setup routine. In the setup function, we set initial conditions of variables when necessary, and run any preliminary code that we only want to run once. This is where serial communication is initiated, which is required for running the serial monitor.
4. From the setup function, we go to the loop routine. This is the main routine of the sketch. This is not only where your main code goes, but it will be executed over and over, so long as the sketch continues to run.
5. Below the loop routine, there is often other functions listed. These functions are user-defined and only activated when called in the setup and loop routine. When these functions are called, the Arduino processes all of the code in the function from top to bottom and then goes back to the next line in the sketch where it left off when the function was called. Functions are good because they allow you to run standard routines - over and over - without having to write the same lines of code over and over. You can simply call upon a function multiple times, and this will free up memory on the chip because the function routine is only written once. It also makes code easier to read. To learn how to form your own functions, check out[this page](http://arduino.cc/en/Reference/FunctionDeclaration).

* All of that said, the only two parts of the sketch which are mandatory are the Setup and Loop routines.
* Code must be written in the [Arduino Language](http://arduino.cc/en/Reference/HomePage), which is roughly based on C.
* Almost all statements written in the Arduino language must end with a ;
* Conditionals (such as [if statements](http://arduino.cc/en/Reference/If) and [for loops](http://arduino.cc/en/Reference/For)) do not need a ;
* Conditionals have their own rules and can be found under "[Control Structures](http://arduino.cc/en/Reference/HomePage)" on the [Arduino Language](http://arduino.cc/en/Reference/HomePage) page
* Variables are storage compartments for numbers. You can pass values into and out of variables. Variables must be defined (stated in the code) before they can be used and need to have a data type associated with it. To learn some of the basic data types, review the [Language Page](http://arduino.cc/en/Reference/HomePage).

Okay! So let us say we want to write code that reads a photocell connected to pin A0, and use the reading we get from the photocell to control the brightness of an LED connected to pin D9.

First, we want to open the BareMinimum sketch, which can be found at:

File --> Examples --> 1.Basic --> BareMinimum

## The BareMinimum Sketch should look like this:

<pre>void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Next, lets put a header on the code, so other people know about what we are making, why, and under what terms:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Once that is all squared away, let us define the pin names, and establish variables:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

const int analogInPin = A0;

// name digital pin 9 a constant name

const int LEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

## Now that variables and pin names are set, let us write the actual code:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

const int analogInPin = A0;

// name digital pin 9 a constant name

const int LEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

//nothing here right now

}

void loop() {

//read the analog in pin and set the reading to the photocell variable

photocell = analogRead(analogInPin);

//control the LED pin using the value read by the photocell

analogWrite(LEDPin, photocell);

//pause the code for 1/10 second

//1 second = 1000

delay(100);

}

If we want to see what numbers the analog pin is actually reading from the photocell, we will need to use the serial monitor. Let's activate the serial port and output those numbers:

<pre>/\*

LED Dimmer

by Genius Arduino Programmer

2012

Controls the brightness of an LED on pin D9

based on the reading of a photocell on pin A0

This code is in the Public Domain

\*/

// name analog pin 0 a constant name

const int analogInPin = A0;

// name digital pin 9 a constant name

const int LEDPin = 9;

//variable for reading a photocell

int photocell;

void setup() {

Serial.begin(9600);

}

void loop() {

//read the analog in pin and set the reading to the photocell variable

photocell = analogRead(analogInPin);

//print the photocell value into the serial monitor

Serial.print("Photocell = " );

Serial.println(photocell);

//control the LED pin using the value read by the photocell

analogWrite(LEDPin, photocell);

//pause the code for 1/10 second

//1 second = 1000

delay(100);

}

**10.HARDWARE TESTING**

**10.1 CONTINUITY TEST:**

In electronics, a continuity test is the checking of an electric circuit to see if current flows (that it is in fact a complete circuit). A continuity test is performed by placing a small voltage (wired in series with an LED or noise-producing component such as a piezoelectric speaker) across the chosen path. If electron flow is inhibited by broken conductors, damaged components, or excessive resistance, the circuit is "open".

Devices that can be used to perform continuity tests include multi meters which measure current and specialized continuity testers which are cheaper, more basic devices, generally with a simple light bulb that lights up when current flows.

An important application is the continuity test of a bundle of wires so as to find the two ends belonging to a particular one of these wires; there will be a negligible resistance between the "right" ends, and only between the "right" ends.

This test is the performed just after the hardware soldering and configuration has been completed. This test aims at finding any electrical open paths in the circuit after the soldering. Many a times, the electrical continuity in the circuit is lost due to improper soldering, wrong and rough handling of the PCB, improper usage of the soldering iron, component failures and presence of bugs in the circuit diagram. We use a multi meter to perform this test. We keep the multi meter in buzzer mode and connect the ground terminal of the multi meter to the ground. We connect both the terminals across the path that needs to be checked. If there is continuation then you will hear the beep sound.

**10.2 POWER ON TEST:**

This test is performed to check whether the voltage at different terminals is according to the requirement or not. We take a multi meter and put it in voltage mode. Remember that this test is performed without microcontroller. Firstly, we check the output of the transformer, whether we get the required 12 v AC voltage.

Then we apply this voltage to the power supply circuit. Note that we do this test without microcontroller because if there is any excessive voltage, this may lead to damaging the controller. We check for the input to the voltage regulator i.e., are we getting an input of 12v and an output of 5v. This 5v output is given to the microcontrollers’ 40th pin. Hence we check for the voltage level at 40th pin. Similarly, we check for the other terminals for the required voltage. In this way we can assure that the voltage at all the terminals is as per the requirement.

**BIBLIOGRAPHY**

**TEXT BOOKS REFERED**

1. ATMEGA 328 Data Sheets.

**WEBSITES**

* www.atmel.com
* www.beyondlogic.org
* www.wikipedia.org
* www.howstuffworks.com
* www.alldatasheets.com