**Visvesvaraya Technological University**

**Belgaum, Karnataka-590 018**

*A Project Report on*

# "ARDUINO BASED HOME AUTOMATION SYSTEM USING BLUETOOTH THROUGH AN ANDROID MOBILE"

*Project Report submitted in partial fulfillment of the requirement for the award of the degree of*

**Bachelor of Engineering**
**In**
**Electrical & Electronics Engineering**

*Submitted by*

| | |
|---|---|
| **RAHUL S** | **1CR15EE063** |
| **SRIKANTH S K** | **1CR17EE414** |
| **MAHESH M** | **1CR17EE403** |
| **SHANKAR GOUDA N** | **1CR17EE411** |

*Under the Guidance of*

**RESHMA P ELDHO**
**Assistant professor**
**Department of Electrical & Electronics Engineering**
**CMR Institute of Technology**

**CMR Institute of Technology, Bengaluru-560 037**

**Department of Electrical & Electronics Engineering**

**2019-2020**

**CMR INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**

**AECS Layout, Bengaluru-560 037**

# Certificate

Certified that the project work entitled **"ARDUINO BASED HOME AUTOMATION SYSTEM USING BLUETOOTH THROUGH AN ANDROID MOBILE "** carried out by Mr. RAHUL S, 1CR15EE063 ; Mr. SRIKANTH S K, 1CR17EE414 ; Mr. MAHESH M, 1CR17EE414 ; Mr. SHANKAR GOUDA N, 1CR17EE411 are bonafied students of CMR Institute of Technology, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Electrical & Electronics Engineering of the Visvesvaraya Technological University, Belgaum, during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect for the

*Signature of the Guide*

-----------------------------

Mr. RESHMA P ELDHO
Professor
EEE Department
CMRIT, Bengaluru

*Signature of the HOD*

--------------------------------

Dr. K. Chitra
Professor & HOD
EEE Department
CMRIT, Bengaluru

*Signature of the Principal*

-----------------------------

Dr. Sanjay Jain
Principal,
CMRIT, Bengaluru

*External Viva*

| Name of the Examiners | Signature & Date |
|---|---|
| 1. | |
| 2. | |

**CMR INSTITUTE OF TECHNOLOY**

**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**

**AECS Layout, Bengaluru-560 037**

# DECLARATION

**We,** [Mr. RAHUL S**(1CR16EE063),** Mr./ SRIKANTH S K **(1CR17EE414),** Mr. MAHESH M **(1CR17EE403),** Mr. SHANKAR GOUDA N **(1CR17EE411)],** hereby declare that the report entitled **"ARDUINO BASED HOME AUTOMATION SYSTEM USING BLUETOOTH THROUGH AN ANDROID MOBILE "** has been carried out by us under the guidance of **RESHMA P ELDHO,** Designation, Department of Electrical & Electronics Engineering, CMR Institute of Technology, Bengaluru, in partial fulfillment of the requirement for the degree of **BACHELOR OF ENGINEERING in ELECTRICAL & ELECTRONICS ENGINEERING**, of Visveswaraya Technological University, Belagaum during the academic year 2019-20. The work done in this report is original and it has not been submitted for any other degree in any university.

| | |
|---|---|
| Place: Bengaluru | **RAHUL S** (1CR15EE063) |
| Date: | **SRIKANTH S K** (1CR17EE414) |
| | **MAHESH M** (1CR17EE403) |
| | **SHANKAR GOUDA** (1CR17EE411) |

# Abstract

The main objective of this project is to develop a home automation system with Android application controlled remote. Remote controlled home automation system provides a simpler solution with Android application technology. Remote operation is achieved by any smart-phone/Tablet etc., with Android OS, upon a GUI (Graphical User Interface) based touch screen operation. Home automation is becoming more and more popular day by day due to its numerous advantages. This can be achieved by local networking or by remote control.

Our Aim is to design a kit that can be used for controlling AC Loads from Android phone by using Arduino microcontroller. Home automation refers to the use of computer and information technology to control home appliances and other features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/micro-controller based networks with varying degrees of intelligence and automation. Home automation is adopted for reasons of ease, security and energy efficiency.

In modern construction in industrialized nations, most homes have been wired for electrical power, telephones, TV outlets (cable or antenna), and a doorbell. Many household tasks were automated by the development of specialized appliances. For instance, automatic washing machines were developed to reduce the manual labor of cleaning clothes, and water heaters reduced the labor necessary for bathing.

# Acknowledgement

*The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people, who are responsible for the completion of the project and who made it possible, because success is outcome of hard work and perseverance, but steadfast of all is encouraging guidance.So with gratitude we acknowledge all those whose guidance and encouragement served us to motivate towards the success of the project work.*

*We take great pleasure in expressing our sincere thanks to* ***Dr. Sanjay Jain****,* ***Principal, CMR Institute of Technology, Bengaluru*** *for providing an excellent academic environment in the college and for his continuous motivation towards a dynamic career. We would like to profoundly thank* ***Dr. B Narasimha Murthy****, Vice-principal of CMR Institute of Technology and the whole* ***Management*** *for providing such a healthy environment for the successful completion of the project work.*

*We would like to convey our sincere gratitude to* ***Dr. K Chitra****,* ***Head of Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for her invaluable guidance and encouragement and for providing good facilities to carry out this project work.*

*We would like to express our deep sense of gratitude to* ***Na, Assistant Professor, Electrical and Electronics Engineering, CMR Institute of Technology, Bengaluru*** *for his/her exemplary guidance, valuable suggestions, expert advice and encouragement to pursue this project work.*

*We are thankful to all the faculties and laboratory staffs of* ***Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for helping us in all possible manners during the entire period.*

*Finally, we acknowledge the people who mean a lot to us, our parents, for their inspiration, unconditional love, support, and faith for carrying out this work to the finishing line. We want to give special thanks to all our friends who went through hard times together, cheered us on, helped us a lot, and celebrated each accomplishment.*

*Lastly, to the* ***Almighty****, for showering His Blessings and to many more,*

*whom we didn’t mention here.*

# LIST OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

| | |
|---|---|
| **HAS** | Home Automation System |
| **IDE** | Integrated Development Environment |
| **CPU** | Central Processing Unit |
| **RTOS** | Real Time Operating System |
| **USB** | Universal Serial Bus |
| **TCP/IP** | Transmission Control Protocol/Internet Provider |
| **UHF** | Ultra High Frequency |
| **RAM** | Random Access Memory |
| **SIG** | Special Interest Group |
| **SOC** | System on Chip |
| **AOSP** | Android Open Source Project |
| **SPDT** | Single Pole Double Throw |
| **DPDT** | Double Pole Double Throw |
| **IPS** | Indoor Positioning System |
| **PCB** | Printed Circuit Board |
| **HVAC** | Heating Ventilating Air Conditioning |
| **EEPROM** | Electrically Erasable Programmable Read Only Memory |

A Project Report on

# ARDUINO BASED HOME AUTOMATION SYSTEM USING BLUETOOTH THROUGH AN ANDROID MOBILE

## LIST OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

| | |
|---|---|
| **HAS** | Home Automation System |
| **IDE** | Integrated Development Environment |
| **CPU** | Central Processing Unit |
| **RTOS** | Real Time Operating System |
| **USB** | Universal Serial Bus |
| **TCP/IP** | Transmission Control Protocol/Internet Provider |
| **UHF** | Ultra High Frequency |
| **RAM** | Random Access Memory |
| **SIG** | Special Interest Group |
| **SOC** | System on Chip |
| **AOSP** | Android Open Source Project |
| **SPDT** | Single Pole Double Throw |
| **DPDT** | Double Pole Double Throw |
| **IPS** | Indoor Positioning System |
| **PCB** | Printed Circuit Board |
| **HVAC** | Heating Ventilating Air Conditioning |
| **EEPROM** | Electrically Erasable Programmable Read Only Memory |

# ABSTRACT

The main objective of this project is to develop a home automation system with Android application controlled remote. Remote controlled home automation system provides a simpler solution with Android application technology. Remote operation is achieved by any smart-phone/Tablet etc., with Android OS, upon a GUI (Graphical User Interface) based touch screen operation. Home automation is becoming more and more popular day by day due to its numerous advantages. This can be achieved by local networking or by remote control.

Our Aim is to design a kit that can be used for controlling AC Loads from Android phone by using Arduino microcontroller. Home automation refers to the use of computer and information technology to control home appliances and other features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/micro-controller based networks with varying degrees of intelligence and automation. Home automation is adopted for reasons of ease, security and energy efficiency.

In modern construction in industrialized nations, most homes have been wired for electrical power, telephones, TV outlets (cable or antenna), and a doorbell. Many household tasks were automated by the development of specialized appliances. For instance, automatic washing machines were developed to reduce the manual labor of cleaning clothes, and water heaters reduced the labor necessary for bathing.

# CHAPTER-1

# INTRODUCTION TO EMBEDDED SYSTEMS

## INTRODUCTION

The Goal of this Project is to develop an Embedded System, which is used to control home appliances by android mobile using Bluetooth.

## DESCRIPTION

Presently, conventional walls are located in different parts of home makes it difficult to the user to go near them to operate and thus requires a lot of maintenance and effort. Even for elderly and physically challenged people to do so, it is difficult. This project demonstrates a simple home automation system that allows the user to control it with a wireless device such as a Wi-Fi or Bluetooth enabled mobile phone. A desktop PC is used to run the server software.

The System allows the user to control each of the lights and fans individually. It can automatically turn off the main lights and turn on a night lamp at a specified time. By measuring the signal strength, it can detect when the user enters room and automatically turn on the light and fans, and then automatically turn them off when the user leaves the room.

## REQUIREMENTS

### Hardware Requirements

1. Power Supply Unit
2. ATMEGA 328PU Microcontroller
3. Bluetooth Module
4. Electrical Loads

**Software Requirements**

1. Arduino IDE
2. Android OS Marshmallow (6.0)

## EMBEDDED SYSTEMS

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-bit, 64-bit Controllers used with OS like Personal Digital Assistant and Mobile phones etc. Lower end embedded systems - Generally 8-bit, 16-bit Controllers used with a minimal operating systems and hardware layout designed for the specific purpose.

**System Design Calls**

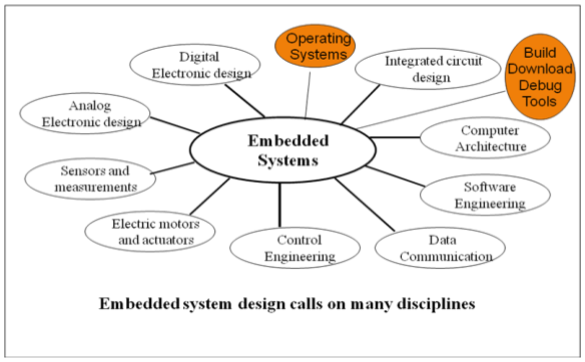


**Fig. 1.1: Embedded System Design Calls**

**Embedded System Design Cycle**

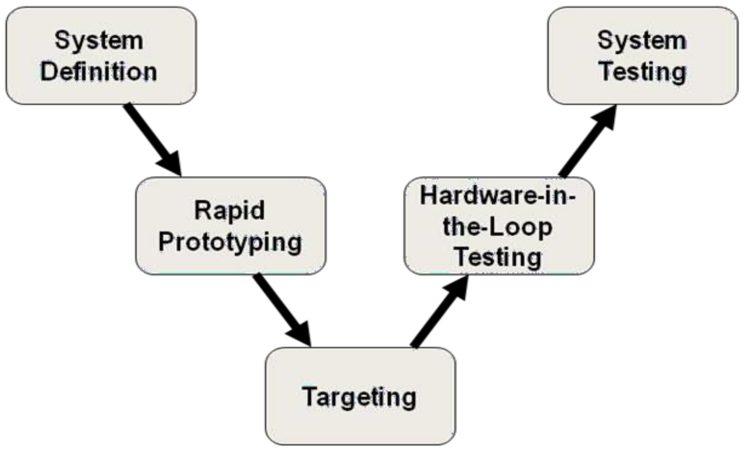


**Fig. 1.2: V Diagram**

## CHARACTERSTICS OF AN 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.

## EMBEDDED SYSTEMS SOFTWARE ARCHITECTURE

There are several different types of software architecture in common use.

### Simple Control Loop

In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

### Interrupt-Controlled System

Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

**Cooperative Multitasking**

A nonpreemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for *no operation*), etc.

The advantages and disadvantages are similar to that of the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue.

**Preemptive Multitasking or Multi-threading**

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Because of these complexities, it is common for organizations to use a real-time operating system (RTOS), allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a *generic* real time system, due to limitations regarding memory size, performance, or battery life. The choice that an RTOS is required brings in its own issues, however, as the selection must be done prior to starting to the application development process. This timing forces developers to choose the embedded operating system for their device based upon current requirements and so restricts future options to a large extent The restriction of future options becomes more of an issue as product life decreases. Additionally the level of complexity is continuously growing as devices are required to manage variables such as serial, USB, TCP/IP, Bluetooth, Wireless LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on. These trends are leading to the uptake of embedded middleware in addition to a real-time operating system.

### Microkernels and Exokernels

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to and extensible by application programmers.

### Monolithic kernels

In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment similar to a desktop operating system like Linux or Microsoft Windows, and is therefore very productive for development; on the downside, it requires considerably more hardware resources, is often more expensive, and, because of the complexity of these kernels, can be less predictable and reliable.

Common examples of embedded monolithic kernels are embedded Linux and Windows CE.

Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as wireless routers and GPS navigation systems. Here are some of the reasons:

- Ports to common embedded chip sets are available.
- They permit re-use of publicly available code for device drivers, web servers, firewalls, and other code.
- Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume.
- Many engineers believe that running application code in user mode is more reliable and easier to debug, thus making the development process easier and the code more portable.
- Features requiring faster response than can be guaranteed can often be placed in hardware.

**Exokernel**

ExoKernel is an operating system developed at the Massachusetts Institute of Technology that seeks to provide application-level management of hardware resources. The exokernel architecture is designed to separate resource protection from management to facilitate application-specific customization.

# CHAPTER-2
# BLOCK DIAGRAM

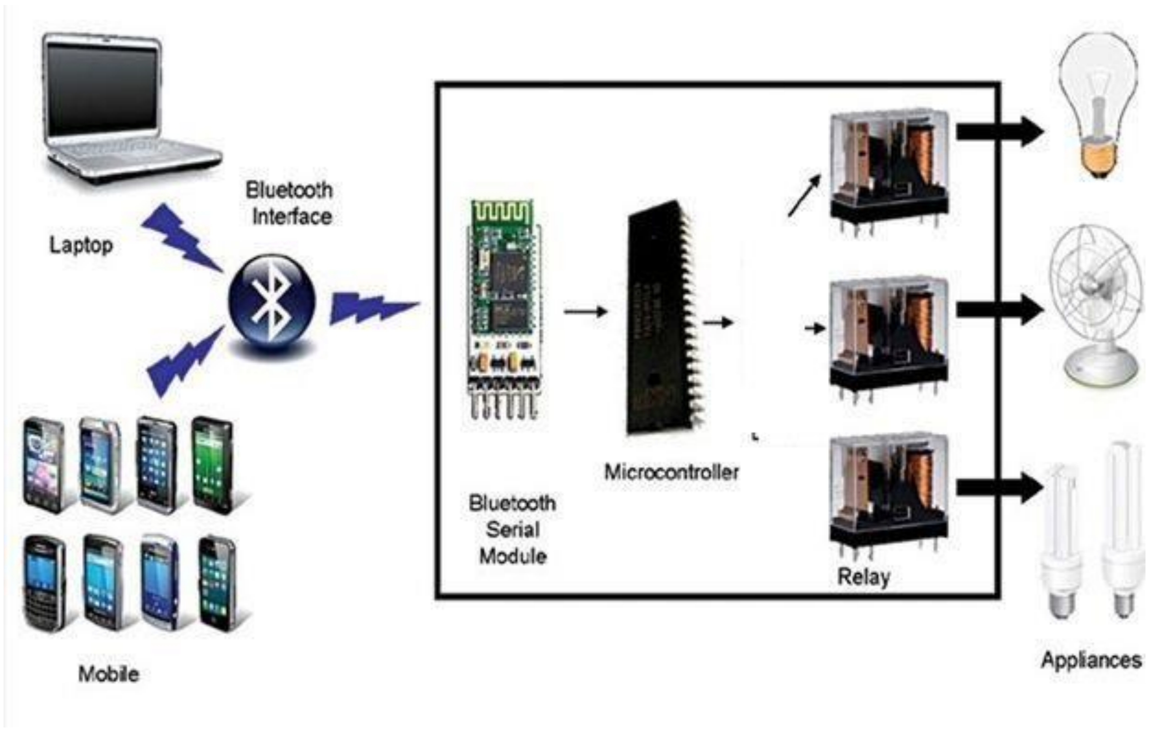


**Fig. 2.1: Project Block Diagram**

## 2.1 BLUETOOTH

**Bluetooth** is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz from fixed and mobile devices, and building personal area networks (PANs). Invented by telecom vendor Ericsson in 1994 it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 25,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The IEEE standardized Bluetooth as IEEE 802.15.1, but no longer maintains the standard. The Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks. A manufacturer must make a device meet Bluetooth SIG standards to market it as a Bluetooth device. A network of patents apply to the technology, which are licensed to individual qualifying devices.

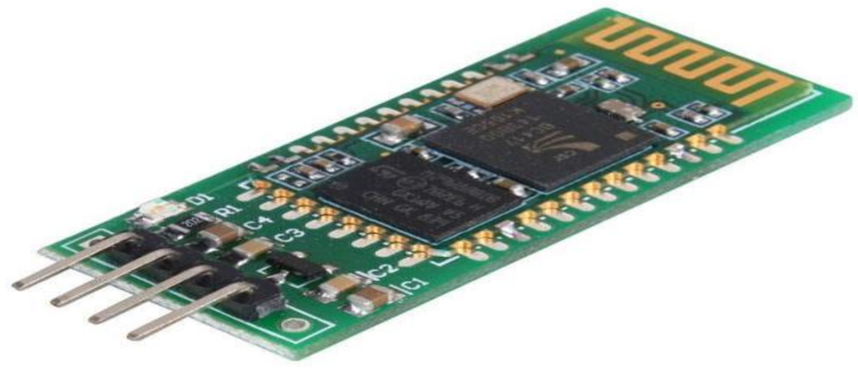


**Fig. 2.2: Bluetooth Module**

## MICROCONTROLLER

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

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

Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit mill watts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just Nano watts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

**Fig. 2.3: Micro Controller**

## ELECTRICAL LOADS

An **Electrical Load** is an electrical component or portion of a circuit that consumes electric power. This is opposed to a power source, such as a battery or generator, which produces power. In electric power circuits examples of loads are appliances and lights.

The term is used more broadly in electronics for a device connected to a signal source, whether or not it consumes power. If an electric circuit has an output port, a pair of terminals that produces an electrical signal, the circuit connected to this terminal (or its input impedance) is the *load*. For example, if a CD player is connected to an amplifier, the CD player is the source and the amplifier is the load.

Load affects the performance of circuits with respect to output voltages or currents, such as in sensors, voltage sources, and amplifiers. Mains power outlets provide an easy example: they supply power at constant voltage, with electrical appliances connected to the power circuit collectively making up the load. When a high-power appliance switches on, it dramatically reduces the load impedance.

If the load impedance is not very much higher than the power supply impedance, the voltages will drop. In a domestic environment, switching on a heating appliance may cause incandescent lights to dim noticeably.

**Fig. 2.4: Electrical Loads**

## ANDROID OS

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

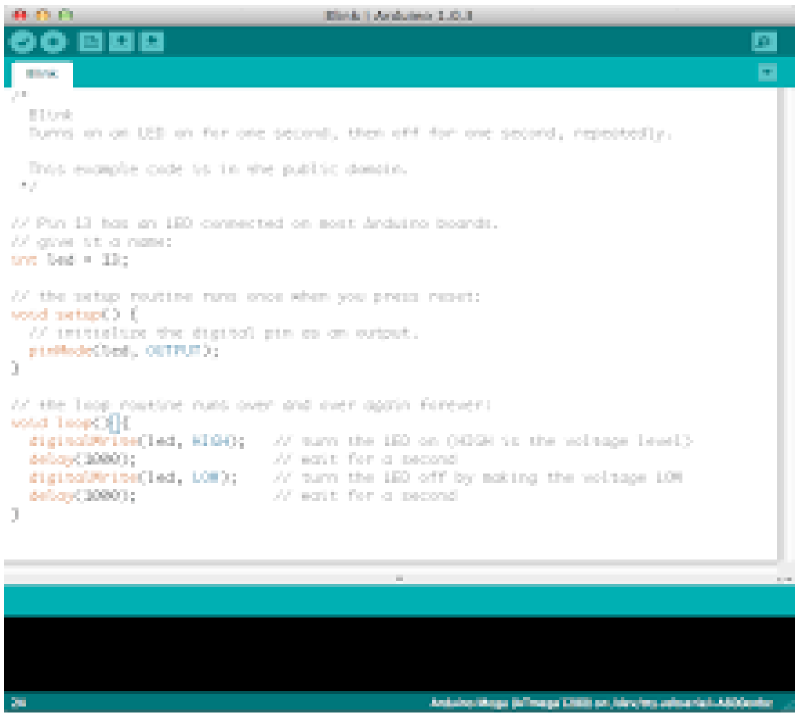
Android has an active community of developers and enthusiasts who use the Android Open Source Project (AOSP) source code to develop and distribute their own modified versions of the operating system These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, with a comparable level of quality; provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the HP Touchpad.

**Fig. 2.5: Android OS**

## ARDUINO IDE

Arduino is common term for a software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog I/O pins that can interface to various expansion boards (termed *shields*) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named *Processing*, which also supports the languages C and C++.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Fig. 2.6: Arduino IDE

# CHAPTER-3
# HARDWARE REQUIREMENTS

## TRANSFORMER

Transformers convert AC electricity from one voltage to another with a little loss of power. Step-up transformers increase voltage, step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high voltage to a safer low voltage.

**Fig. 3.1: A Typical Transformer**

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; instead they are linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down and current is stepped up.

The ratio of the number of turns on each coil, called the turn’s ratio, determines the ratio of the voltages. A step-down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil to give a low output voltage.

$$\text{Turns Ratio} = (V_p / V_s) = (N_p / N_s) \text{ -----------------------Eq.3.1}$$

Where,

Vp = primary (input) voltage.

Vs = secondary (output) voltage.

Np = number of turns on primary coil.

Ns = number of turns on secondary coil.

Ip = primary (input) current.

Is = secondary (output) current.

**Ideal Power Equation**

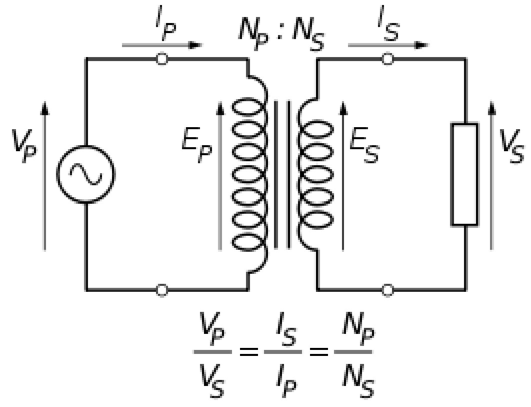


**Fig. 3.2: Transformer Circuit Diagram**

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

$$P_{\text{incoming}} = I_{\text{p}}V_{\text{p}} = P_{\text{outgoing}} = I_{\text{s}}V_{\text{s}}, \qquad \text{------------Eq.3.2}$$

Giving the ideal transformer equation

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

--------------------------Eq.3.3

If the voltage is increased, then the current is decreased by the same factor. The impedance in one circuit is transformed by the *square* of the turns ratio. For example, if an impedance $Z_s$ is attached across the terminals of the secondary coil, it appears to the primary circuit to have an impedance of $(N_p/N_s)^2Z_s$. This relationship is reciprocal, so that the impedance $Z_p$ of the primary circuit appears to the secondary to be $(N_s/N_p)^2Z_p$.

## 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.

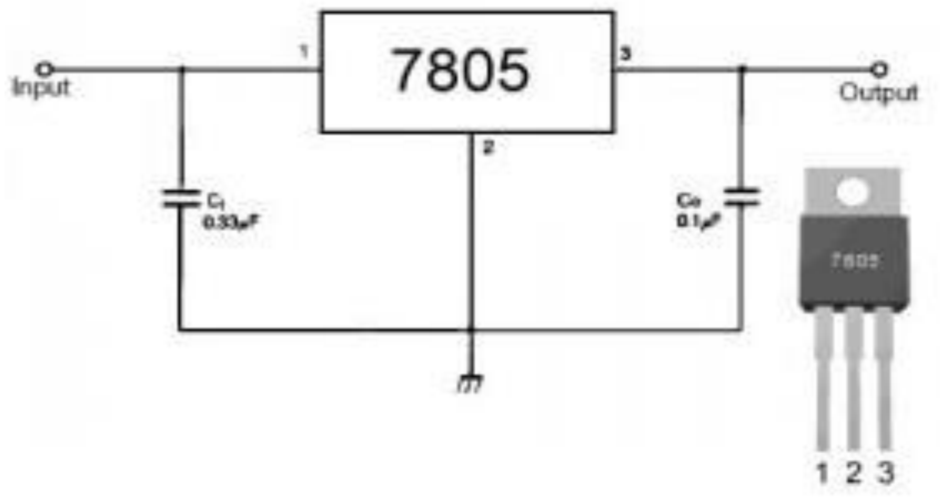


**Fig. 3.3: Voltage Regulator**

**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.

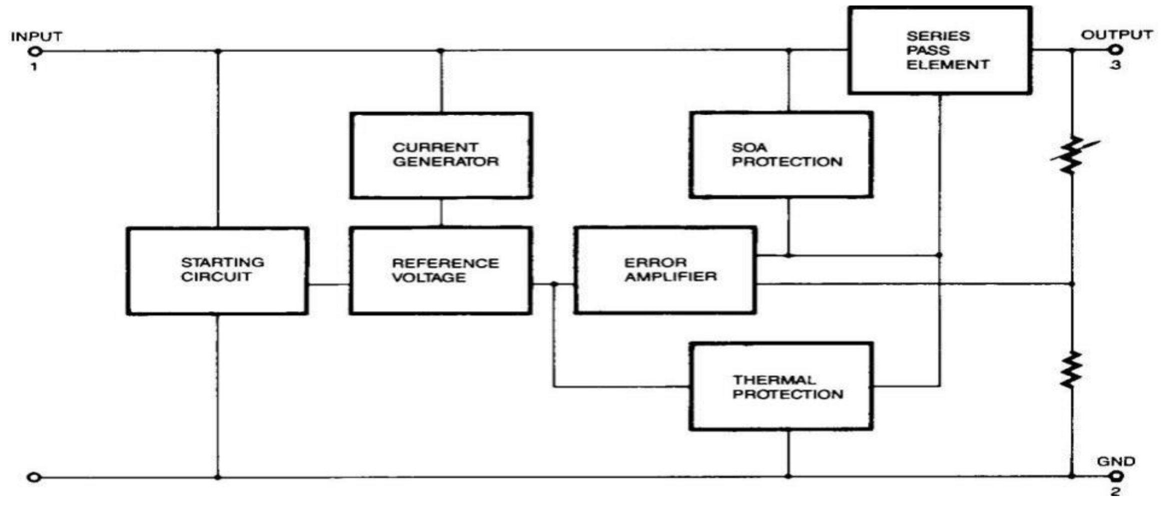
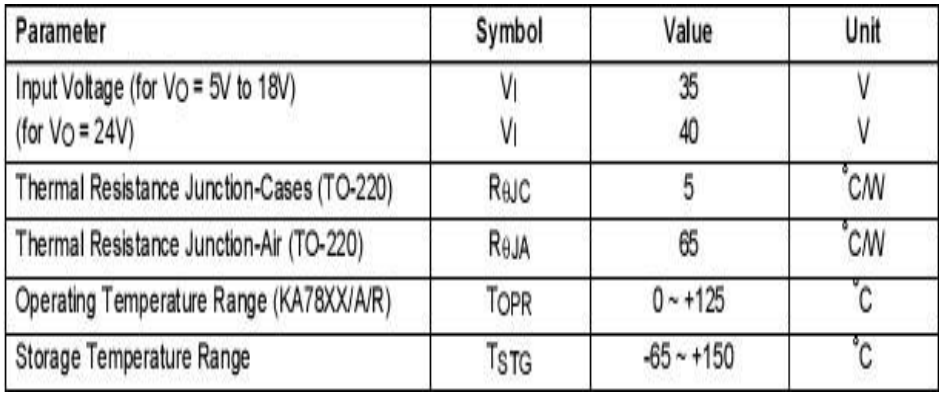


**Fig. 3.4: Block Diagram of Voltage Regulator**

Regulators, these devices can be used with external components to obtain adjustable voltages and currents.

**Absolute Maximum Ratings**



| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Input Voltage (for $V_O$ = 5V to 18V)<br>(for $V_O$ = 24V) | $V_I$<br>$V_I$ | 35<br>40 | V<br>V |
| Thermal Resistance Junction-Cases (TO-220) | $R_{\theta JC}$ | 5 | °C/W |
| Thermal Resistance Junction-Air (TO-220) | $R_{\theta JA}$ | 65 | °C/W |
| Operating Temperature Range (KA78XX/A/R) | $T_{OPR}$ | 0 ~ +125 | °C |
| Storage Temperature Range | $T_{STG}$ | -65 ~ +150 | °C |

**Table 3.2.1: Ratings of the Voltage Regulator**

## LED

LED’s 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.

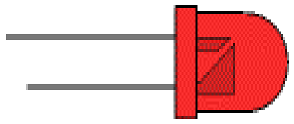


**Fig. 3.5: Typical LED**

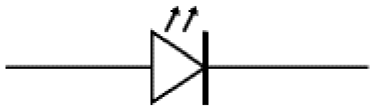


**Fig. 3.6: Circuit symbol**

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.

**Fig. 3.7: Different types of LED’s**

## 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**

$$V = IR \qquad \text{---Eq.3.4}$$

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).

**Fig. 3.8: Resistors**

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 passive electronic component which implements electrical resistance as a circuit element. When a voltage V is applied across the terminals of a resistor, a current I will flow through the resistor in direct proportion to that voltage. The reciprocal of the constant of proportionality is known as the 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:

$$I = \frac{V}{R}$$

------Eq.3.5

Resistors are common elements of 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 (wire made of a high-resistivity alloy, such as nickel-chrome). Resistors are also implemented within integrated circuits, particularly analog devices, and can also be integrated into hybrid and printed circuits.

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. When specifying that resistance in an electronic design, the required precision of the resistance may require attention to the manufacturing tolerance of the chosen resistor, according to its specific application. The temperature coefficient of the resistance may also be of concern in some precision applications. Practical resistors are also specified as having a maximum power 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. In a high voltage circuit, attention must sometimes be paid to the rated maximum working voltage of the resistor.

The series 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 or 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 (symbol: Ω) is the SI unit of electrical resistance, named after Georg Simon Ohm. An ohm is equivalent to a volt per 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 = $10^{3}$ Ω), and megohm (1 MΩ = $10^{6}$ Ω) are also in common usage.

The reciprocal of resistance R is called conductance G = 1/R and is measured in Siemens (SI unit), sometimes referred to as a mho. Thus a Siemens is the reciprocal of an ohm: $S = \Omega^{-1}$. 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.

## 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.

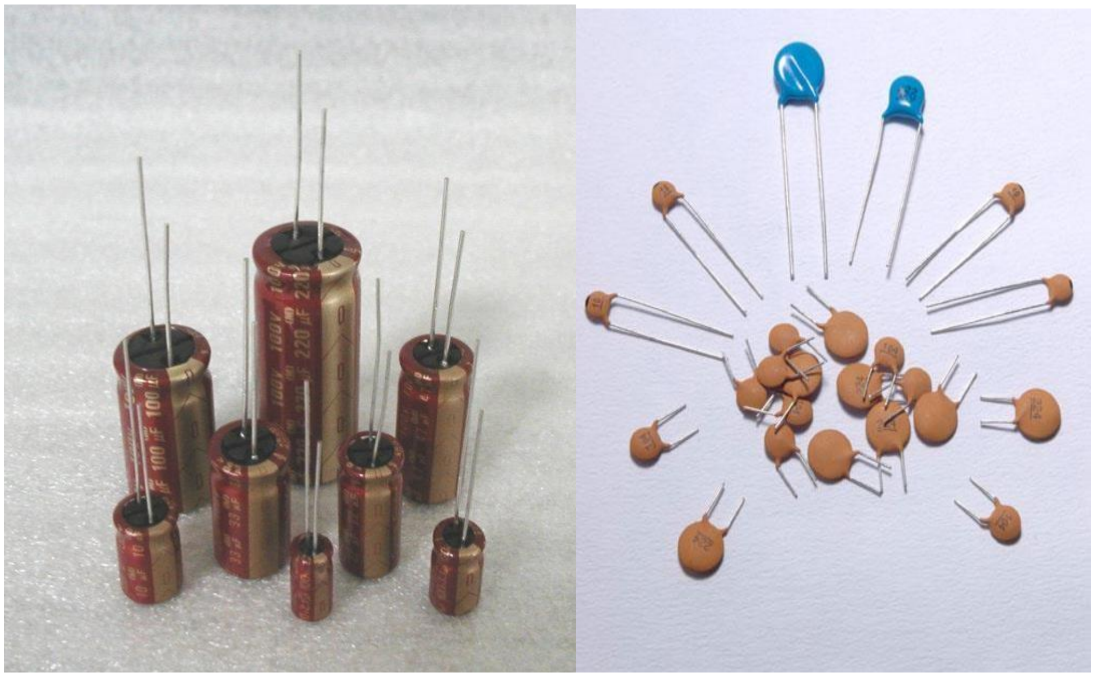


**Fig. 3.9: Capacitors**

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

A capacitor is a passive electronic component consisting of a pair of conductors separated by a dielectric (insulator). When there is a potential difference (voltage) across the conductors, a static electric field develops in the dielectric that stores energy and produces a mechanical force between the conductors. An ideal capacitor is characterized by a single constant value, capacitance, measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them.

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 and also has an electric field strength limit, resulting in a breakdown voltage, while the conductors and leads introduce an undesired inductance and resistance.

**Theory of Operation**

Main article: Capacitance

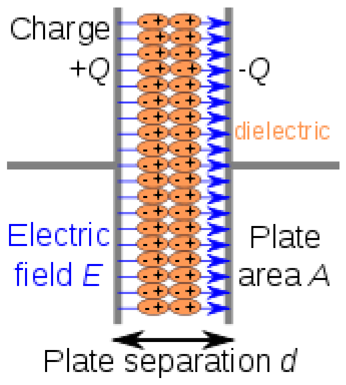


**Fig. 3.10: Capacitor Operation**

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

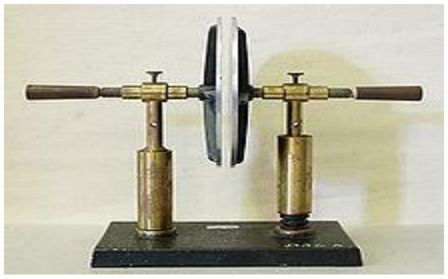


**Fig. 3.11: A Simple Demonstration of a Parallel-Plate Capacitor**

A capacitor consists of two conductors separated by a non-conductive region. The non-conductive region is called the dielectric or sometimes the dielectric medium. In simpler terms, the dielectric is just an electrical insulator. Examples of dielectric mediums are glass, air, paper, Vacuum, and even a semiconductor depletion region chemically identical to the conductors.

A capacitor is assumed to be self-contained and isolated, with no net 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 units, a capacitance of one farad means that one coulomb of charge on each conductor causes a voltage of one 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.

$$C = \frac{Q}{V}$$

------- -Eq.3.6

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

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

--------Eq.3.7

**Energy storage**

Work 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 position. The work done in establishing the electric field, and hence the amount of energy stored, is given by

$$W = \int_{q=0}^{Q} V \mathrm{d}q = \int_{q=0}^{Q} \frac{q}{C} \mathrm{d}q = \frac{1}{2}\frac{Q^2}{C} = \frac{1}{2}CV^2 = \frac{1}{2}VQ.$$

----Eq.3.8

**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 of the current as well as proportional to the voltage as discussed above. As with any antiderivative, a constant of integration is added to represent the initial voltage v ($t_0$). This is the integral form of the capacitor equation.

$$v(t) = \frac{q(t)}{C} = \frac{1}{C}\int_{t_0}^{t} i(\tau)\mathrm{d}\tau + v(t_0)$$ -----Eq.3.9

Taking the derivative of this, and multiplying by C, yields the derivative form.

$$i(t) = \frac{\mathrm{d}q(t)}{\mathrm{d}t} = C\frac{\mathrm{d}v(t)}{\mathrm{d}t}$$ ------------ -Eq.3.10

The dual of the capacitor is the inductor, which stores energy in the 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.

## PUSH BUTTONS

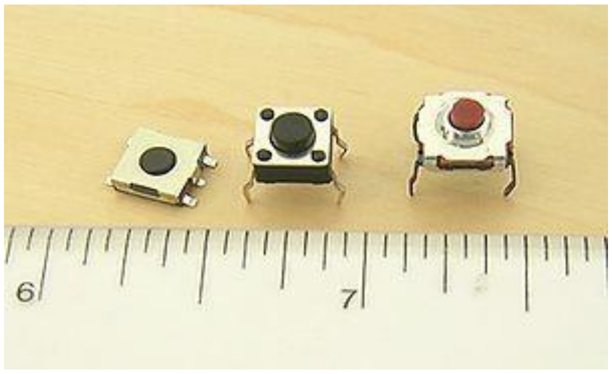


**Fig. 3.12: Push Buttons**

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**

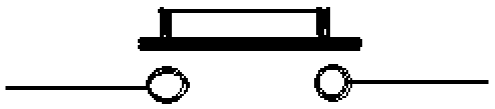


**Fig. 3.13: Push on Button**

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

## 3.7 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

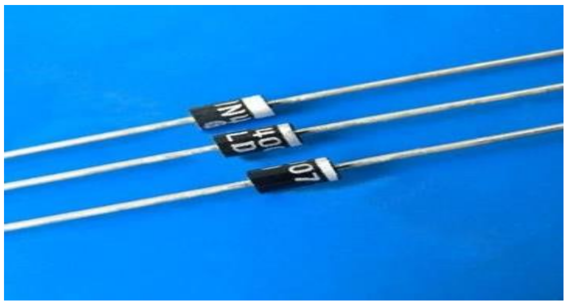


**Fig. 3.14: 1N4007 diodes**

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.

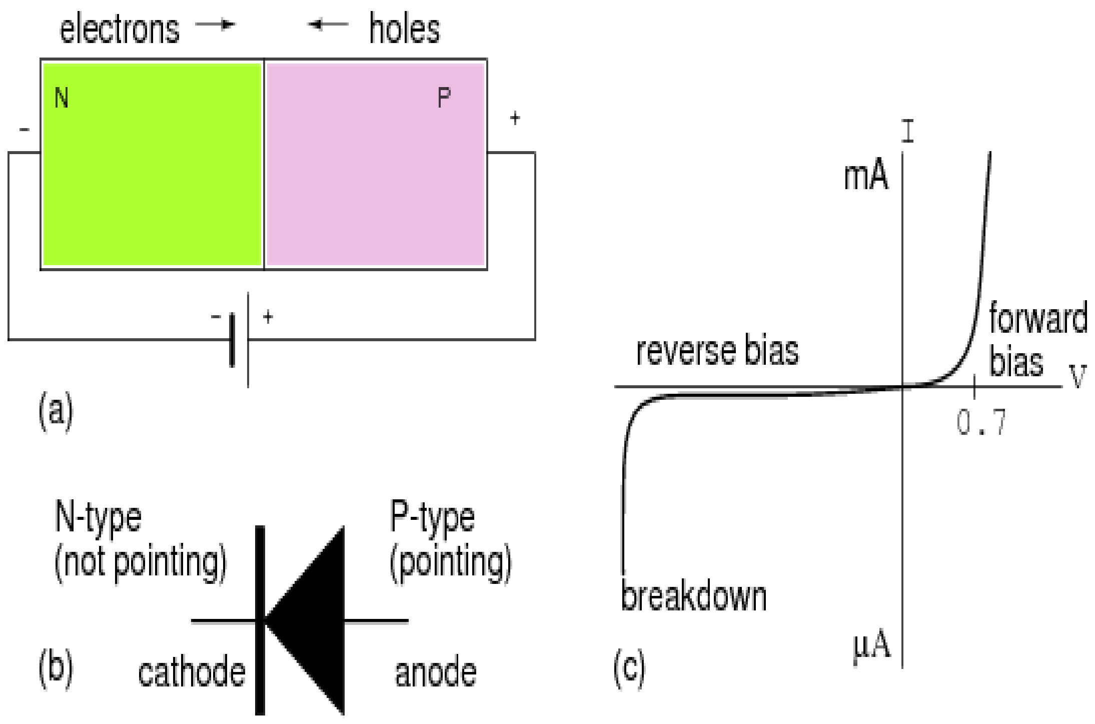


**Fig. 3.15: P-N Junction Diode**

**P-N 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 P-N junction. But before we can understand how the P-N 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. 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.

**Maximum Ratings**

| Rating | Symbol | 1N4001 | 1N4002 | 1N4003 | 1N4004 | 1N4005 | 1N4006 | 1N4007 | Unit |
|---|---|---|---|---|---|---|---|---|---|
| *Peak Repetitive Reverse Voltage<br>Working Peak Reverse Voltage<br>DC Blocking Voltage | $V_{RRM}$<br>$V_{RWM}$<br>$V_R$ | 50 | 100 | 200 | 400 | 600 | 800 | 1000 | Volts |
| *Non±Repetitive Peak Reverse Voltage<br>(halfwave, single phase, 60 Hz) | $V_{RSM}$ | 60 | 120 | 240 | 480 | 720 | 1000 | 1200 | Volts |
| *RMS Reverse Voltage | $V_{R(RMS)}$ | 35 | 70 | 140 | 280 | 420 | 560 | 700 | Volts |
| *Average Rectified Forward Current<br>(single phase, resistive load,<br>60 Hz, see Figure 8, $T_A$ = 75°C) | $I_O$ | 1.0 | | | | | | | Amp |
| *Non±Repetitive Peak Surge Current<br>(surge applied at rated load<br>conditions, see Figure 2) | $I_{FSM}$ | 30 (for 1 cycle) | | | | | | | Amp |
| Operating and Storage Junction<br>Temperature Range | $T_J$<br>$T_{stg}$ | ± 65 to +175 | | | | | | | °C |

**Table.3.7.1: Maximum Ratings of 1N4007**

## RELAYS

A relay is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

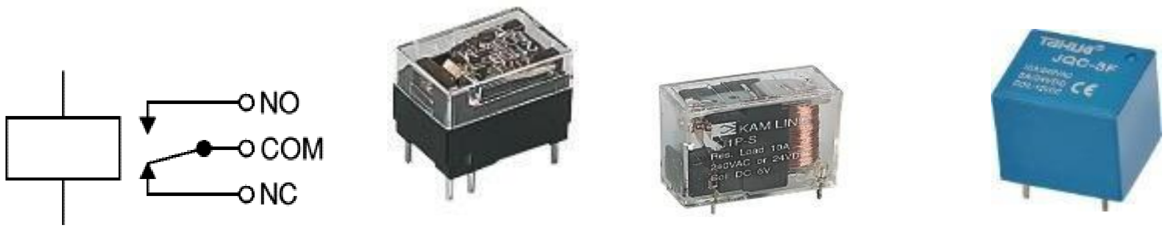


**Fig. 3.16: Typical Relays**

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and most have double throw (changeover) switch contacts as shown in the diagram.

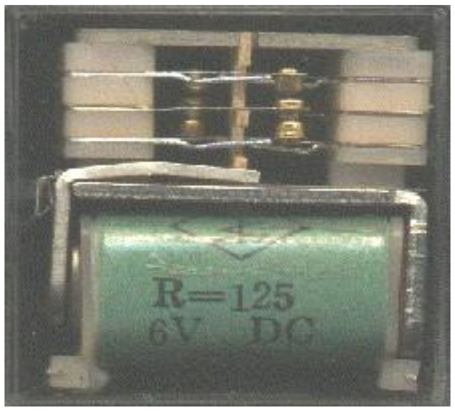


**Fig. 3.17: Relay showing coil and switch contacts**

Relays allow one circuit to switch a second circuit which can be completely separate from the first. For example a low voltage battery circuit can use a relay to switch a 230V AC mains circuit. There is no electrical connection inside the relay between the two circuits; the link is magnetic and mechanical.

The coil of a relay passes a relatively large current, typically 30mA for a 12V relay, but it can be as much as 100mA for relays designed to operate from lower voltages. Most ICs (chips) cannot provide this current and a transistor is usually used to amplify the small IC current to the larger value required for the relay coil. The maximum output current for the popular 555 timer IC is 200mA so these devices can supply relay coils directly without amplification.

Relays are usually SPDT or DPDT but they can have many more sets of switch contacts, for example relays with 4 sets of changeover contacts are readily available. For further information about switch contacts and the terms used to describe them please see the page on switches. Most relays are designed for PCB mounting but you can solder wires directly to the pins providing you take care to avoid melting the plastic case of the relay.

The supplier's catalogue should show you the relay's connections. The coil will be obvious and it may be connected either way round. Relay coils produce brief high voltage 'spikes' when they are switched off and this can destroy transistors and ICs in the circuit. To prevent damage you must connect a protection diode across the relay coil.

The figure shows a relay with its coil and switch contacts. You can see a lever on the left being attracted by magnetism when the coil is switched on. This lever moves the switch contacts.

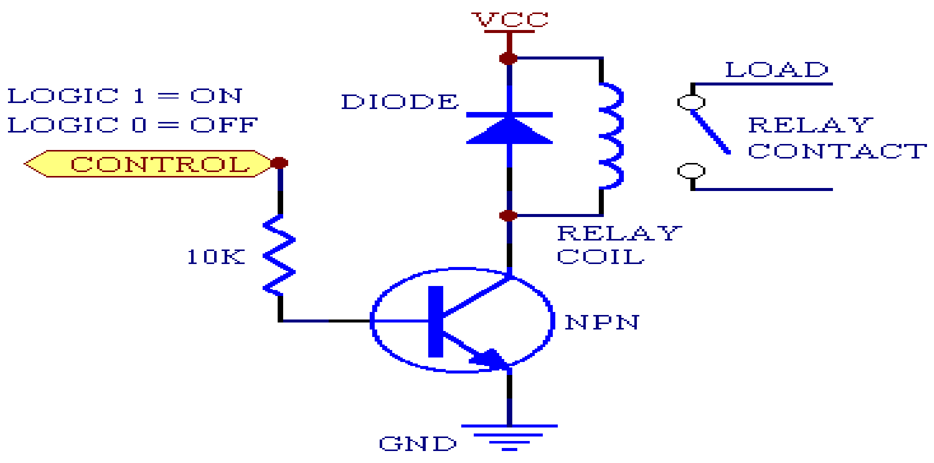


**Fig. 3.18: Schematic Relay**

There is one set of contacts (SPDT) in the foreground and another behind them, making the relay DPDT.

The relay's switch connections are usually labelled COM, NC and NO

- COM = Common, always connect to this; it is the moving part of the switch.
- NC = Normally Closed, COM is connected to this when the relay coil is off.
- NO = Normally Open, COM is connected to this when the relay coil is on.

**Applications Of Relays**

Relays are used to and for

- Control a high-voltage circuit with a low-voltage signal, as in some types of modems or audio amplifiers.
- Control a high-current circuit with a low-current signal, as in the starter solenoid of an automobile.
- Detect and isolate faults on transmission and distribution lines by opening and closing circuit breakers.
- Time delay functions. Relays can be modified to delay opening or delay closing a set of contacts. A very short (a fraction of a second) delay would use a copper disk between the armature and moving blade assembly. Current flowing in the disk maintains magnetic field for a short time, lengthening release time. For a slightly longer (up to a minute) delay, a dashpot is used. A dashpot is a piston filled with fluid that is allowed to escape slowly. The time period can be varied by increasing or decreasing the flow rate. For longer time periods, a mechanical clockwork timer is installed.

## HC-05/06 BLUETOOTH MODULE

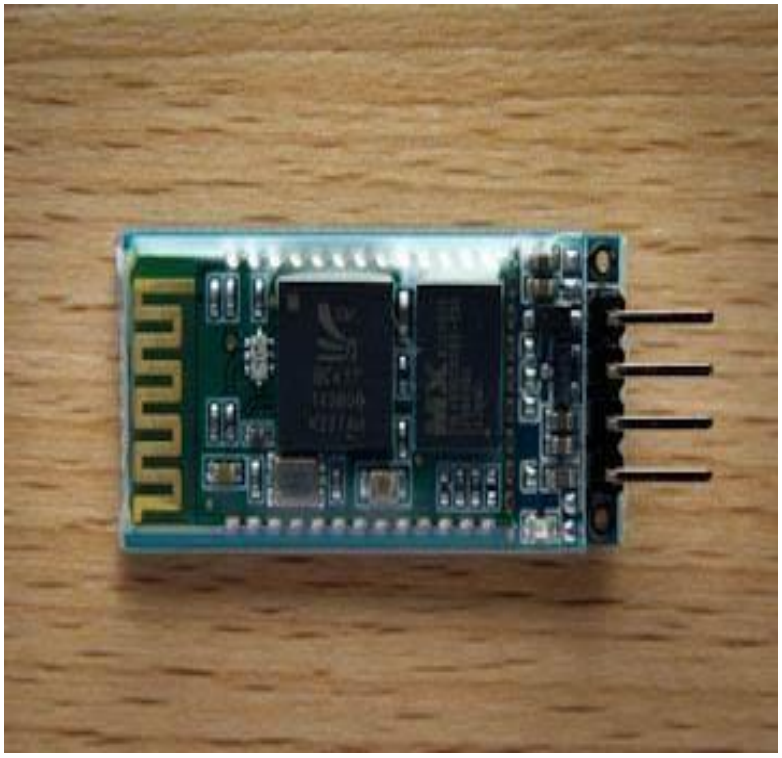
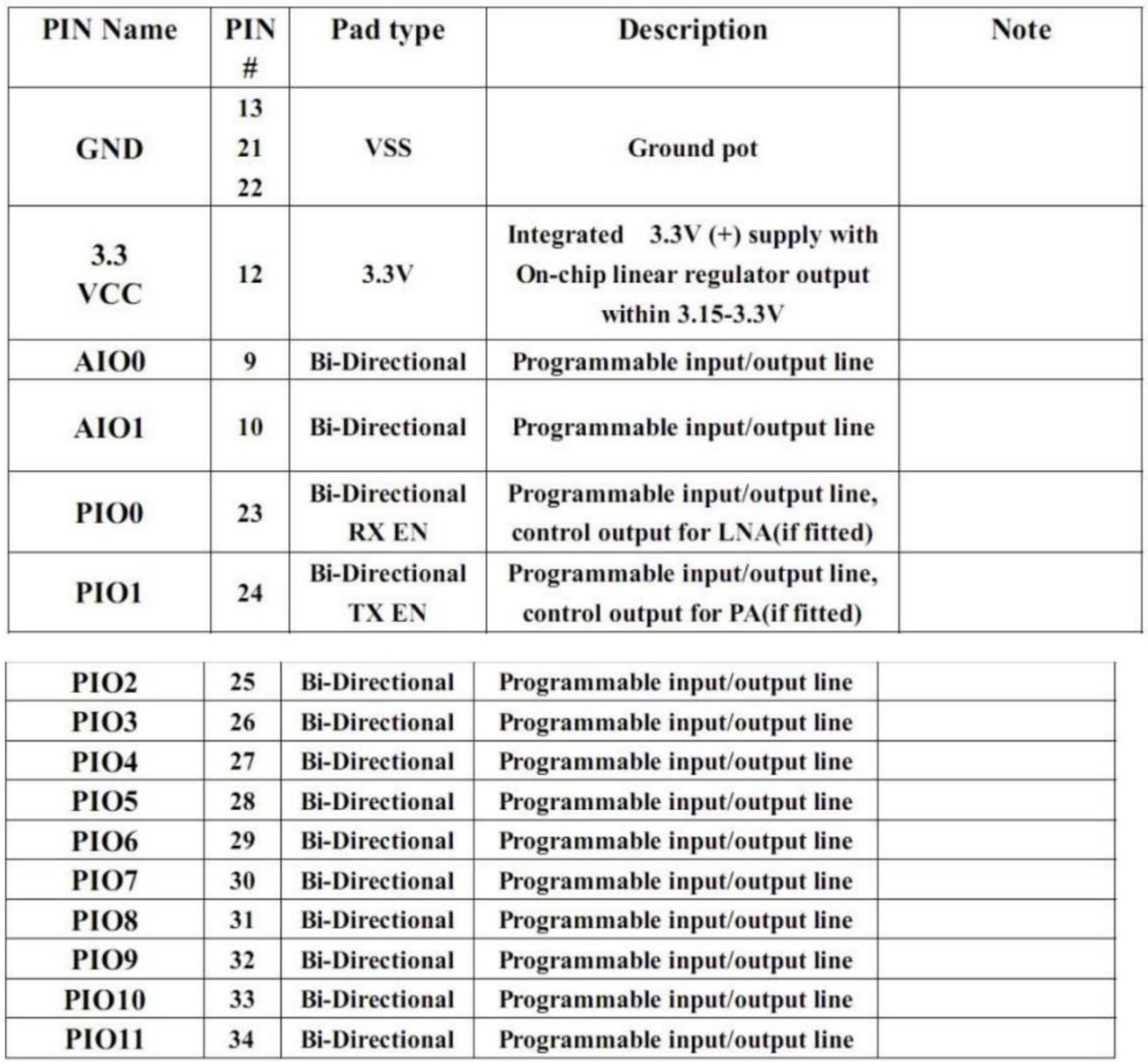


**Fig. 3.19: Bluetooth Module**

The Bluetooth module HC-05 is a MASTER/SLAVE module. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. By default the factory setting is SLAVE. This post walks through the AT commands of Bluetooth module .Through AT Commands you can change the ROLE of the module. Name, Password of the Module can be changed .The list of AT commands can be downloaded.

Remember the models HC-04 or HC-06 are SLAVE only modules .HC-05 is the module of interest in this post. To identify the model we can see the pin count. If the module has 5 or 6 pins it is HC-05.If the module has only 4 pins then it is HC-04 or HC-06. The firmware for HC04 is LINVOR & for the HC05 it is HC05 itself. If we scan for Bluetooth devices from our Android phone we can see this name. The extra pins in HC-05 are the KEY & STATE pins. The KEY pin is used to enter the Command mode.

## 3.9.1 HC -05 Pin Configuration



| PIN Name | PIN # | Pad type | Description | Note |
|---|---|---|---|---|
| GND | 13<br>21<br>22 | VSS | Ground pot | |
| 3.3 VCC | 12 | 3.3V | Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V | |
| AIO0 | 9 | Bi-Directional | Programmable input/output line | |
| AIO1 | 10 | Bi-Directional | Programmable input/output line | |
| PIO0 | 23 | Bi-Directional RX EN | Programmable input/output line, control output for LNA(if fitted) | |
| PIO1 | 24 | Bi-Directional TX EN | Programmable input/output line, control output for PA(if fitted) | |
| PIO2 | 25 | Bi-Directional | Programmable input/output line | |
| PIO3 | 26 | Bi-Directional | Programmable input/output line | |
| PIO4 | 27 | Bi-Directional | Programmable input/output line | |
| PIO5 | 28 | Bi-Directional | Programmable input/output line | |
| PIO6 | 29 | Bi-Directional | Programmable input/output line | |
| PIO7 | 30 | Bi-Directional | Programmable input/output line | |
| PIO8 | 31 | Bi-Directional | Programmable input/output line | |
| PIO9 | 32 | Bi-Directional | Programmable input/output line | |
| PIO10 | 33 | Bi-Directional | Programmable input/output line | |
| PIO11 | 34 | Bi-Directional | Programmable input/output line | |

**Table.3.9.1: HC-05 Pin Configuration**

## ARDUINO UNO

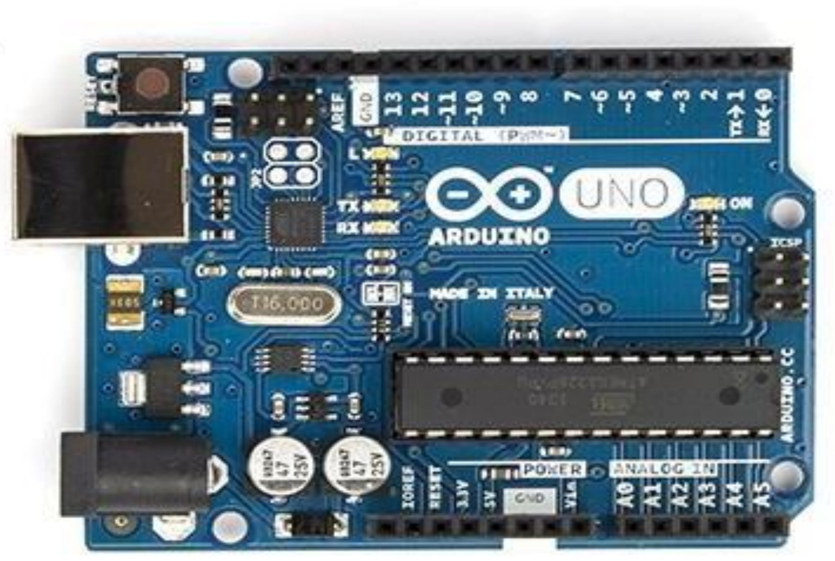


**Fig. 3.20: Arduino Uno**

The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

**Technical Specifications**

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 gm |

## Circuit Diagram

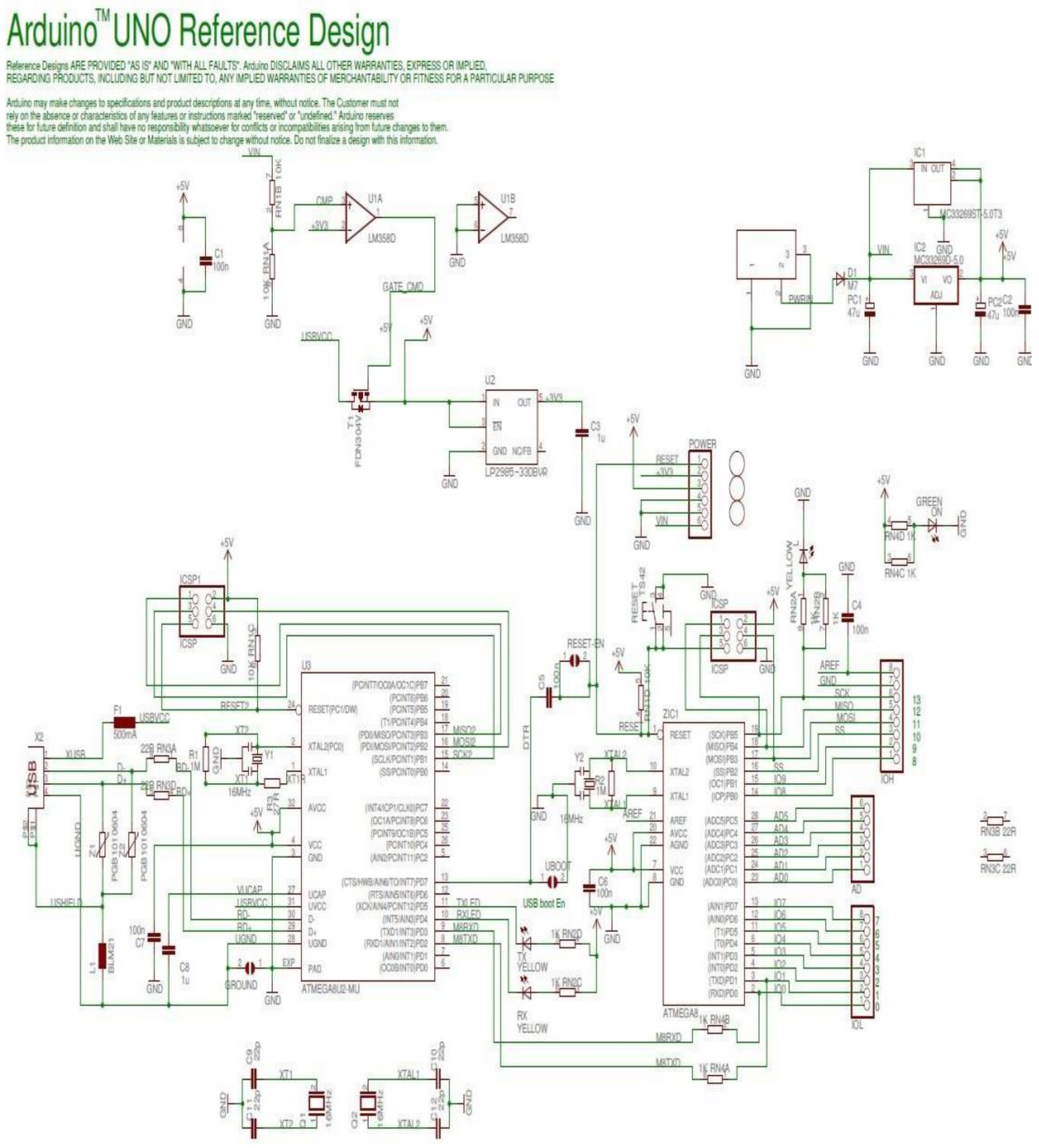

Arduino™ UNO Reference Design
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
ATMEGA8U2-MU
ATMEGA8
LM358D
LP2985-330DBVR
MC33269D-5.0
MC33269ST-5.0T3
USB
POWER
ICSP
ICSP1
IOH
IOL
AD
RESET
GND
+5V
VIN

**Fig. 3.21: Arduino Design**

# CHAPTER-4

# MICROCONTROLLER AT328PU

## FEATURES

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
- 131 Powerful Instructions – Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 20 MIPS Throughput at 20MHz
- On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
- 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
- 256/512/512/1KBytes EEPROM
- 512/1K/1K/2KBytes Internal SRAM
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- Data retention: 20 years at 85°C/100 years at 25°C(1)
- Optional Boot Code Section with Independent Lock Bits
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
- Programming Lock for Software Security
- Atmel® QTouch® library support
- Capacitive touch buttons, sliders and wheels
- QTouch and QMatrix® acquisition
- Up to 64 sense channels
- Peripheral Features
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and

- Capture Mode
- Real Time Counter with Separate Oscillator
- Six PWM Channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package
- Temperature Measurement
- 6-channel 10-bit ADC in PDIP Package
- Temperature Measurement
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change

## DESCRIPTION

The ATmegaXX8 family of AVR devices includes the ATmega48, ATmega88, ATmega168, and ATmega328. They are similar in that they have identical I/O. They have three timers - two 8-bit and one 16-bit.They differ in the amount of memory available, with flash memory size of from 4kB to 32kB. The devices are widely used in Arduino and Arduino clone boards. The difference between the A and PA (ATmega328 vs ATmega328P) is that the PA parts are Pico power. They can run at lower Vcc's than the A parts, thereby using much less power.

The ATmega48A/PA/88A/PA/168A/PA/328/P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48A/PA/88A/PA/168A/PA/328/P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed. The device is manufactured using Atmel’s high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional on-volatile memory programmer, or by an On-chip Boot program running on the AVR core.

## ATmega328P-PU PIN DIAGRAM

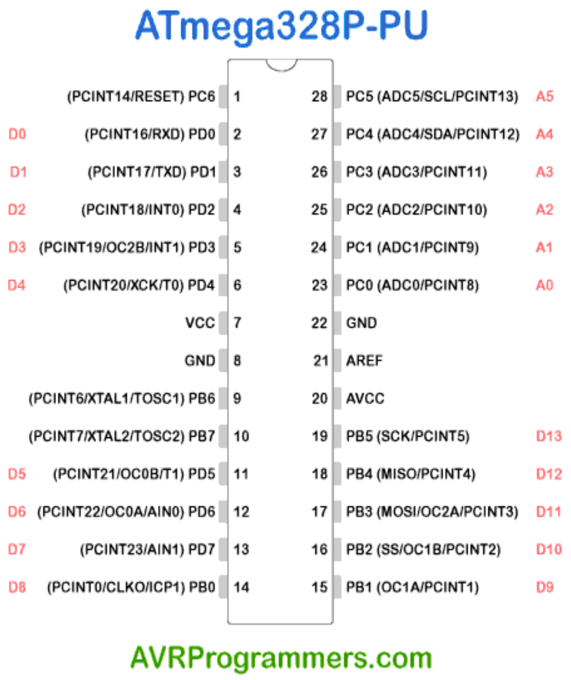


**Fig. 4.1: ATmega328P-PU Pin Diagram**

## AVR MEMORIES

### Overview

This section describes the different memories in the ATmega48A /PA/88A/PA/168A/PA/328/P. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega48A /PA/88A/PA/168A/PA/328/P features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

### In-System Reprogrammable Flash Program Memory

The ATmega48A/PA/88A/PA/168A/PA/328/P contains 4/8/16/32Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as2/4/8/16K x 16. For software security, the Flash Program memory space is divided into two sections, Bootloader Section and Application Program Section in ATmega88PA and ATmega168PA.

The Flash memory has an endurance of at least 10,000 write/erase cycles. TheATmega48A/PA/88A/PA/168A/PA/328/P Program Counter (PC) is 11/12/13/14 bits wide, thus addressing the2/4/8/16K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Self-Programming the Flash, AT mega 48A/48PA" on page 255and "Boot Loader Support – Read-While-Write Self-Programming" on page 263. "Memory Programming" on page 280 contains a detailed description on Flash Programming in SPI- or Parallel Programming mode.

### Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega48A/PA/88A/PA/168A/PA/328/P. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" .The interrupt vectors in ATMEGA 48A/48PA, ATmega88A/88PA, ATmega168A/168PA and ATmega328/328P are generally the same, with the following differences:

- Each Interrupt Vector occupies two instruction words in ATmega168A/168PA and ATmega328/328P, and
- One instruction word in ATmega 48A/48PA and ATmega88A/88PA.

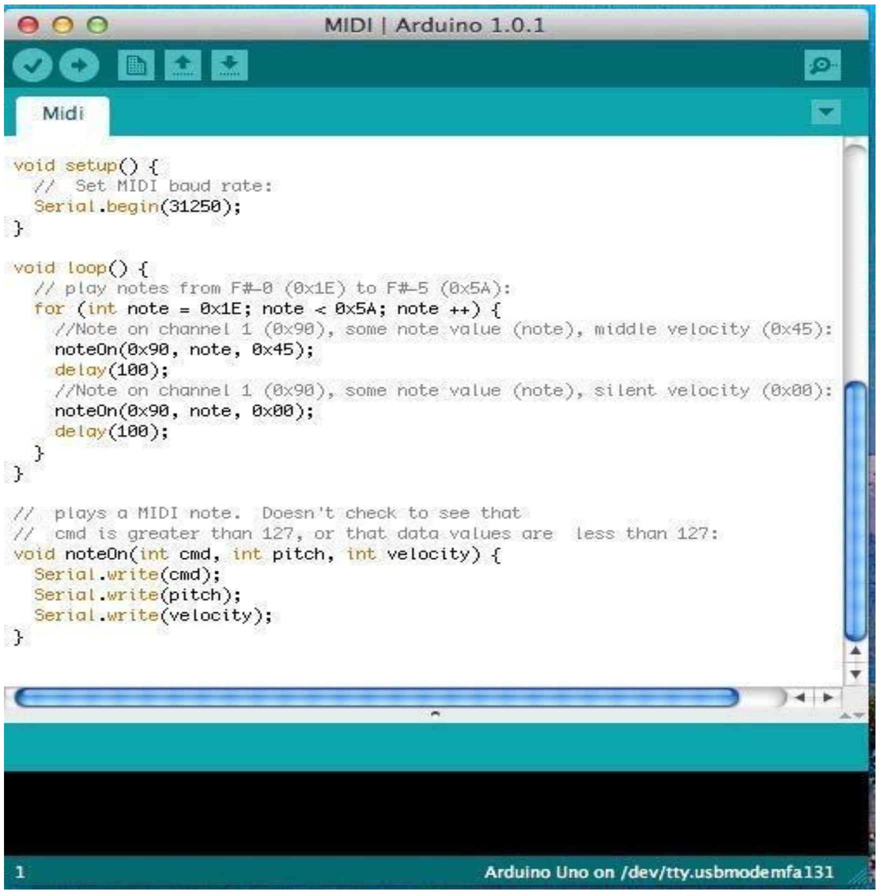
- ATmega 48A/48PA does not have a separate Boot Loader Section.
- ATmega168A/168PA and ATmega328/328P, the Reset Vector is affected by the BOOTRST fuse, and the
- Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

# CHAPTER-5

# SOFTWARE REQUIREMENTS

## ARDUINO SOFTWARE

The board can be programmed from the Arduino software, which is available for different platforms such as Windows, Mac OSX and Linux. It is open source software, which is designed using a Java environment and is also based on processing and avr-gcc.



```
void setup() {
  //  Set MIDI baud rate:
  Serial.begin(31250);
}

void loop() {
  // play notes from F#-0 (0x1E) to F#-5 (0x5A):
  for (int note = 0x1E; note < 0x5A; note ++) {
    //Note on channel 1 (0x90), some note value (note), middle velocity (0x45):
    noteOn(0x90, note, 0x45);
    delay(100);
    //Note on channel 1 (0x90), some note value (note), silent velocity (0x00):
    noteOn(0x90, note, 0x00);
    delay(100);
  }
}

//  plays a MIDI note.  Doesn't check to see that
//  cmd is greater than 127, or that data values are  less than 127:
void noteOn(int cmd, int pitch, int velocity) {
  Serial.write(cmd);
  Serial.write(pitch);
  Serial.write(velocity);
}
```

**Fig. 5.1: Arduino Software**

The software allows users to write their code in C and upload to the board. The boot loader allows the uploading without the need of external hardware programmer. Thus the software is very easy to use and efficient. (Arduino Software 2012 ) Arduino programs may be written in any programming language with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub *main ()* into an executable cyclic executive program.

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

After compiling and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

## ARDUINO DEVELOPMENT

Arduino is an open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the

Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2.

Although the hardware and software designs are freely available under copy left lice-nses, the developers have requested that the name "Arduino" be exclusive to the official product and not be used for derived works without permission. The official policy docu-ment on use of the Arduino name emphasizes that the project is open to incorporating by others into the official product.Several Arduino-compatible products commercially relea-sed have avoided the *Arduino* name by using *-duino* name variants.

## 5.2 CODE DEVELOPED

```
#define LED_PIN1  4
#define LED_PIN2  5
#define LED_PIN3  6
#define LED_PIN4  7
#define LED_PIN5  8
#define LED_PIN6  9
#define LED_PIN7  10
#define LED_PIN8  11

int firstSensor = 0;       // first analog sensor
int secondSensor = 0;      // second analog sensor
int thirdSensor = 0;       // digital sensor
int inByte = 0;            // incoming serial byte
boolean status_unlock;
boolean status_bluetooth;
long interval = 1000;          // interval at which to blink (milliseconds)
long previousMillis = 0;        // will store last time LED was update
int minite,sec;
void setup()
{
// start serial port at 9600 bps:
Serial.begin(9600);
//pinMode(2, INPUT); // digital sensor is on digital pin 2
//establishContact();    // send a byte to establish contact until receiver responds

pinMode(LED_PIN1, OUTPUT);
pinMode(LED_PIN2, OUTPUT);
pinMode(LED_PIN3, OUTPUT);
pinMode(LED_PIN4, OUTPUT);
pinMode(LED_PIN5, OUTPUT);
pinMode(LED_PIN6, OUTPUT);
pinMode(LED_PIN7, OUTPUT);
pinMode(LED_PIN8, OUTPUT);
digitalWrite(LED_PIN1, LOW); // switch off LED
digitalWrite(LED_PIN2, LOW); // switch off LED
digitalWrite(LED_PIN3, LOW); // switch off LED
digitalWrite(LED_PIN4, LOW); // switch off LED
digitalWrite(LED_PIN5, LOW); // switch off LED
digitalWrite(LED_PIN6, LOW); // switch off LED
digitalWrite(LED_PIN7, LOW); // switch off LED
digitalWrite(LED_PIN8, LOW); // switch off LED
status_bluetooth = true;
status_unlock = false;
sec = 0;
}
```

```
void loop()
{
if (Serial.available() > 0) {
inByte = Serial.read();                    // get incoming byte:
if(inByte == 'A'){
digitalWrite(LED_PIN1, HIGH);        // switch on LED
Serial.print('A');                   // send a char
delay(800);
digitalWrite(LED_PIN1, LOW);         // switch off LED

status_unlock = false;
inByte = 0;
    }
if(inByte == 'a'){
digitalWrite(LED_PIN2, HIGH);        // switch on LED
Serial.print('a');                   // send a char
delay(800);
digitalWrite(LED_PIN2, LOW);         // switch off LED

status_unlock = true;
sec = 0;
inByte = 0;
}
if(inByte == 'B'){
digitalWrite(LED_PIN3, HIGH);        // switch on LED
Serial.print('B');                   // send a char
inByte = 0;
}
if(inByte == 'b'){
digitalWrite(LED_PIN3, LOW);         // switch off LED
```

```
Serial.print('b');                    // send a char
inByte = 0;
}
if(inByte == 'C'){
digitalWrite(LED_PIN4, HIGH);        // switch on LED
Serial.print('C');                    // send a char
inByte = 0;
}

if(inByte == 'c'){
digitalWrite(LED_PIN4, LOW);         // switch off LED
Serial.print('c');                    // send a char
inByte = 0;
}
if(inByte == 'D'){
digitalWrite(LED_PIN5, HIGH);        // switch on LED
Serial.print('D');                    // send a char
inByte = 0;
}
if(inByte == 'd'){
digitalWrite(LED_PIN5, LOW);         // switch off LED
Serial.print('d');                    // send a char
inByte = 0;
}
if(inByte == 'E'){
digitalWrite(LED_PIN6, HIGH);        // switch on LED
Serial.print('E');                    // send a char
inByte = 0;
}
if(inByte == 'e'){
```

```
digitalWrite(LED_PIN6, LOW);        // switch off LED
Serial.print('e');                  // send a char
inByte = 0;
}
if(inByte == 'F'){
digitalWrite(LED_PIN7, HIGH);       // switch on LED
Serial.print('F');                  // send a char
inByte = 0;
}

if(inByte == 'f'){
digitalWrite(LED_PIN7, LOW);        // switch off LED
Serial.print('f');                  // send a char
inByte = 0;
}
if(inByte == 'G'){
digitalWrite(LED_PIN8, HIGH);       // switch on LED
Serial.print('G');                  // send a char
inByte = 0;
}
if(inByte == 'g'){
digitalWrite(LED_PIN8, LOW);        // switch off LED
Serial.print('g');                  // send a char
inByte = 0;
}
if(inByte == 'S'){
Serial.print('S');                  // send a char
status_bluetooth = true;
sec = 0;
}
```

```
} // if(Serial
/*
unsigned long currentMillis = millis();

if(currentMillis - previousMillis > interval) {
previousMillis = currentMillis;         // save the last time you blinked the LED
sec++;
if(status_unlock==true){
if(sec== 11){
digitalWrite(LED_PIN1, HIGH);        // switch on LED

delay(800);
digitalWrite(LED_PIN1, LOW);        // switch off LED
status_unlock = false;
sec = 0;
}
}
else sec = 0;
}
*/
} //Loop
void establishContact() {
while (Serial.available() <= 0) {
Serial.print('.');                  // send a capital A
delay(500);
}
}
```

# CHAPTER-6

# ADVANTAGES & APPLICATIONS

## ADVANTAGES

- Adds Safety Through Appliance aand Lighting Control
- Secures Home Through Automated Door Locks
- Increases Awareness Through Security Camera
- Increases Convenience Through Temperature Adjustment
- Saves Time
- Saves Money And Increases Convenience
- Contributes To Economy
- Increases Peace Of Mind
- Allows To Control When Out Of Town
- Keeps Tabs On Children

## APPLICATIONS

- Lighting Appliance Control Subsystem
- Curtain Control Subsystem
- Industrial Automation
- Proximity And Anti-Theft Accessories
- Indoor Positioning System
- HVAC (Heating, Ventilating And Air Conditioning)

# CHAPTER-7

# RESULTS

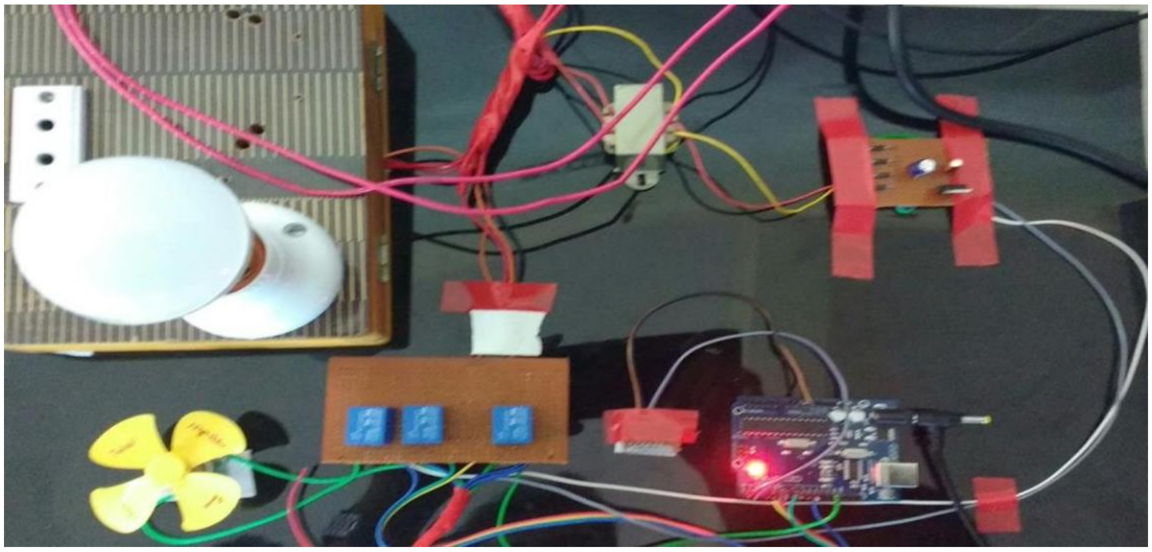


**Fig. 7.1: HAS Input**

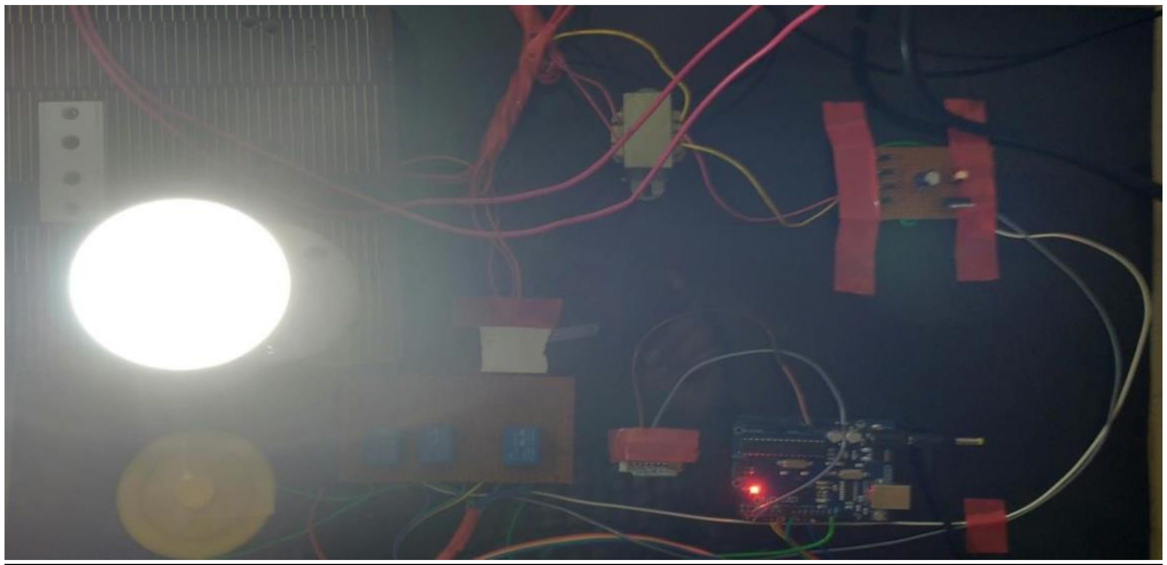


**Fig. 7.2: HAS Output**

## EXPLANATION OF RESULT

I. This is the overview of our Project having a Transformer, Bridge Rectifier, Regulator, Arduino board Bluetooth Module and Electrical loads as input and output sections respectively.

II. In the Input Section we have Android Mobile with Bluetooth.

III. Receives Signal through wireless transmission of Bluetooth.

IV. By Programming Microcontroller and Relay Operation ON/OFF take place.

V. In the Output Section Electrical Loads are connected.

# CHAPTER-8

# CONCLUSION & FUTURE SCOPE

## CONCLUSION

I. This System is designed to improve Standard of Living in Home.

II. Design and implementation of a Home Automation System using Android for mobile phone has been discussed.

III. The purpose of the system is to use mobile phone's inbuilt Bluetooth facility for automation.

IV. Different hardware and software unit of the system are described.

V. The complete application software has been designed using Android, Bluetooth API and Embedded C Language.

VI. The AS application program is tested on various Android mobile phones which are quite satisfactory and response received from the community in general is encouraging.

VII. The AS furnishes a good paradigm for any Automation System based on Android Mobile Phone and Bluetooth.

## FUTURE SCOPE

An automated home can be a very simple grouping of controls, or it can be heavily automated where any appliance that is plugged into electrical power is remotely controlled. Costs mainly include equipment, components, furniture, and custom installation. Ongoing costs include electricity to run the control systems, maintenance costs for the control and networking systems, including troubleshooting, and eventual cost of upgrading as standards change. Increased complexity may also increase maintenance costs for networked devices. Learning to use a complex system effectively may take significant time and training. Control system security may be difficult and costly to maintain, especially if the control system extends beyond the home, for instance by wireless or by connection to the internet or other networks.

Future of Automation: Future will be of Automation of all products. Each and every product will be smart devices that we use daily and that will be controlled through a smart chip called microcontrollers. Each and Every home appliances will be controlled either by PC or hand held devices like PDA or mobile handsets. Some examples of it are when you want you can switch on/off Fan of your home by mobile handset or PC.Smart Grid: Home automation technologies are viewed as integral additions to the Smart grid. The ability to control lighting, appliances, HVAC as well as Smart applications (load shedding, demand response, real-time power usage and price reporting) will become vital as Smart Grid initiatives are rolled out.

# REFERENCES

[1] 2005 arduino (2012). Roadmap. Retrieved September 6, 2012, from http://arduino.cc/en/Main/ArduinoRoadmap

[2] App Inventor. (2012). MIT App Inventor. Retrieved September 19, 2012, from http://appinventor.mit.edu/

[3] Arduino. (2012) Arduino. Retrieved September 6, 2012, from http://arduino.cc/en/

[4] Arduino Uno R3 (2012). Arduino. Retrieved September 6, 2012, from http://arduino.cc/en/Main/ArduinoBoardUno

[5] Ardumoto (2012). Ardumoto-Motor Driver Shield. Retrieved September 11, 2012, from https://www.sparkfun.com/products/9815

[6] Arduino Software (2012). Software Download. Retrieved September 11, 2012, from http://arduino.cc/en/Main/Software

[7] Basic Stamp (2012). Parallax. Retrieved September 6, 2012, from http://www.parallax.com/tabid/295/Default.aspx

[8] BlueSMiRF (2012). Bluetooth Modem- BlueSMiRF . Retrieved September12 2012, from https://www.sparkfun.com/products/10269

[9] Bluetooth (2012). Bluetooth Technology. Retrieved September 12, 2012, from http://www.bluetooth.com/Pages/Basics.aspx

[10] CDIO (2012) CDIO Initiative. Retrieved November 9, 2012, from http://www.cdio.org/

[11] David Kushner (October 2011). Making of Arduino. Retrieved September 6, 2012, from http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino/0

# APPENDIX A

## Electrical Characteristics Of LM7805, $V_I = 10$ V, $I_O = 500$ mA

| PARAMETER | TEST CONDITIONS | $T_J$ † | µA7805C MIN | µA7805C TYP | µA7805C MAX | UNIT |
|---|---|---|---|---|---|---|
| Output voltage | $I_O$ = 5 mA to 1 A, $V_I$ = 7 V to 20 V, $P_D \le 15$ W | 25° C | 4.8 | 5 | 5.2 | V |
| | | 0° C to 125° C | 4.75 | | 5.25 | |
| Input voltage regulation | $V_I$ = 7 V to 25 V | 25° C | | 3 | 100 | mV |
| | $V_I$ = 8 V to 12 V | | | 1 | 50 | |
| Ripple rejection | $V_I$ = 8 V to 18 V, f = 120 Hz | 0° C to 125° C | 62 | 78 | | dB |
| Output voltage regulation | $I_O$ = 5 mA to 1.5 A | 25° C | | 15 | 100 | mV |
| | $I_O$ = 250 mA to 750 Ma | | | 5 | 50 | |
| Output resistance | f = 1 kHz | 0° C to 125° C | | 0.017 | | Ω |
| Temperature coefficient of output voltage | $I_O$ = 5 mA | 0° C to 125° C | | –1.1 | | mV/° C |
| Output noise voltage | f = 10 Hz to 100 kHz | 25° C | | 40 | | µ V |
| Dropout voltage | $I_O$ = 1 A | 25° C | | 2 | | V |
| Bias current | | 25° C | | 4.2 | 8 | mA |
| Bias current change | $V_I$ = 7 V to 25 V | 0° C to 125° C | | | 1.3 | mA |
| | $I_O$ = 5 mA to 1 A | | | | 0.5 | |
| Short-circuit output current | | 25° C | | 750 | | mA |
| Peak output current | | 25° C | | 2.2 | | A |

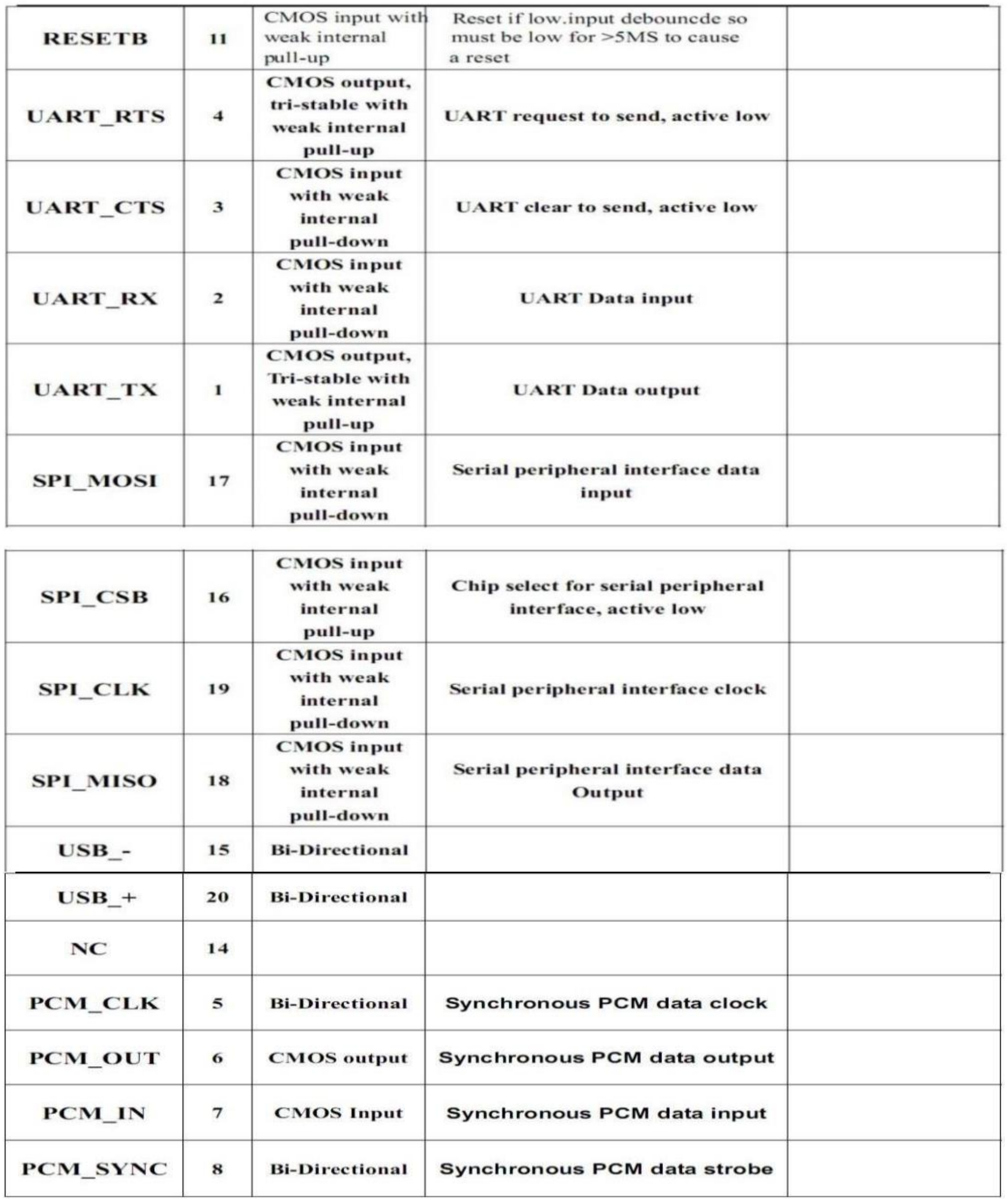
# APPENDIX B

**Electrical Characteristics of 1N4007**

| Rating | Symbol | Typ | Max | Unit |
|---|---|---|---|---|
| Maximum Instantaneous Forward Voltage Drop<br>($i_F$ = 1.0 Amp, $T_J$ = 25°C) Figure 1 | $v_F$ | 0.93 | 1.1 | Volts |
| Maximum Full±Cycle Average Forward Voltage Drop<br>($I_O$ = 1.0 Amp, $T_L$ = 75°C, 1 inch leads) | $^V F(AV)$ | Ð | 0.8 | Volts |
| Maximum Reverse Current (rated dc voltage)<br>($T_J$ = 25°C)<br>($T_J$ = 100°C) | $I_R$ | <br>0.05<br>1.0 | <br>10<br>50 | mA |
| Maximum Full±Cycle Average Reverse Current<br>($I_O$ = 1.0 Amp, $T_L$ = 75°C, 1 inch leads) | $^I R(AV)$ | Ð | 30 | mA |

# APPENDIX C

## Pin Characteristics of HC-05



| | | | | |
|---|---|---|---|---|
| RESETB | 11 | CMOS input with weak internal pull-up | Reset if low.input debouncde so must be low for >5MS to cause a reset | |
| UART_RTS | 4 | CMOS output, tri-stable with weak internal pull-up | UART request to send, active low | |
| UART_CTS | 3 | CMOS input with weak internal pull-down | UART clear to send, active low | |
| UART_RX | 2 | CMOS input with weak internal pull-down | UART Data input | |
| UART_TX | 1 | CMOS output, Tri-stable with weak internal pull-up | UART Data output | |
| SPI_MOSI | 17 | CMOS input with weak internal pull-down | Serial peripheral interface data input | |
| SPI_CSB | 16 | CMOS input with weak internal pull-up | Chip select for serial peripheral interface, active low | |
| SPI_CLK | 19 | CMOS input with weak internal pull-down | Serial peripheral interface clock | |
| SPI_MISO | 18 | CMOS input with weak internal pull-down | Serial peripheral interface data Output | |
| USB_- | 15 | Bi-Directional | | |
| USB_+ | 20 | Bi-Directional | | |
| NC | 14 | | | |
| PCM_CLK | 5 | Bi-Directional | Synchronous PCM data clock | |
| PCM_OUT | 6 | CMOS output | Synchronous PCM data output | |
| PCM_IN | 7 | CMOS Input | Synchronous PCM data input | |
| PCM_SYNC | 8 | Bi-Directional | Synchronous PCM data strobe | |