

**Vidya Jyothi Institute of Technology**

**(Autonomous)**

**Aziz Nagar, Hyderabad -500075**

**A Major Project Report**

**On**

**IOT Based Power Management and Controlled Socket**

**Submitted for partial fulfillment of the requirements for the award of the degree**

**of**

**BACHELOR OF TECHNOLOGY**

**IN**

**ELECTRICAL AND ELECTRONICS ENGINEERING**

**BY**

**A.Shiva Chaithanya (16911A0252)**

**B.Nikhil (16911A0254)**

**K.Tarun (16911A0270)**

**P.Harish Chandra Prasad (16911A0286)**

**Under the guidance of**

**Dr.A.Srujana**

**Professor**

**Department of Electrical and electronics Engineering**

**VJIT, Hyderabad.**



**Department of Electrical and Electronics Engineering**

*A. Srujana*

**Head of the Department**  
**Department of Electrical & Electronics Engg**  
**Vidya Jyothi Institute of Technology**  
**HYDERABAD-600 075**

*A. Prasad*

**PRINCIPAL**  
**Vidya Jyothi Institute of Technology**  
**Himayatnagar (VIII), C.B. Post,**  
**Hyderabad-75.**

**VIDYA JYOTHI INSTITUTE OF TECHNOLOGY**

**(Autonomous)**

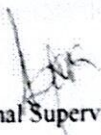
**Himayathnagar (vi), C.B. Post, R.R Dist. 500075**



**CERTIFICATE**

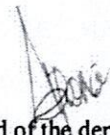
This is to certify that the project work entitled "IOT BASED POWER MANAGEMENT AND CONTROLLED SOCKET" is a bonafide work carried out by Mr. A. Shiva chaithanya (16911A0252), Mr. B Nikhil (16911A0254), Mr. K Tarun (16911A0270), Mr. P Harish Chandra Prasad(16911A0286), in partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN ELECTRICAL AND ELECTRONICS ENGINEERING** to be awarded by the **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, Hyderabad.**

The content in this report has not been submitted to any other university or institute for the award of any degree or diploma.

  
Internal Supervisor

Department of EEE

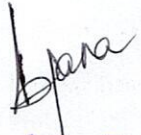
Hyderabad


  
Head of the department

Department of EE

Hyderabad

  
**External Examiner**

  
**Head of the Department**  
Department of Electrical & Electronics Engg  
Vidya Jyothi Institute of Technology  
HYDERABAD-500 075

  
Vidya Jyothi Institute of Technology  
Himayathnagar (vi), C.B. Post,  
Hyderabad-75.

### **DECLARATION**

This is to certify that the work reported in the present project entitled **IOT BASED POWER MANAGEMENT AND CONTROLLED SCOKET** is a record of work done by us in the Department of **Electrical and Electronics Engineering**, Vidya Jyothi Institute of Technology (Autonomous), Jawaharlal Nehru Technological University, Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source.

**Mr. A Shiva Chaithanya (16911A0252)**

**Mr. B Nikhil (16911A0254)**

**Mr. K Tarun (16911A0270)**

**Mr.P Harish Chandra Prasad(16911A0286)**

## ACKNOWLEDGEMENT

I would like to express my sincere gratitude and indebtedness to my project supervisor **Dr. A Srujana** for his valuable suggestions and interest throughout the course of this project.

I am also thankful to Head of the department **Dr. A Srujana** for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, I would like to take this opportunity to thank my family for their support through the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

## **ABSTRACT**

Life today is getting easier and simpler with advancement of automation technology. Manual systems are getting replaced by automatic systems. With the rapid increase in Internet users it has become part of life.

One of its kinds is IoT, latest and emerging technology. Things like consumer goods, industrial goods, etc., can be networked to share information and complete the task remotely. Basic home functions and features can be controlled using IoT from anywhere in the world.

It is meant to save human and electrical energy. In this system each load is monitored by current and potential transformer.



## TABLE OF CONTENT

	PAGE NO
Certificate	i
Declaration	ii
Acknowledgements	iii
Abstract	iv
Table of Content	v
List of Figures	ix
CHAPTER I	1-2
1. Introduction	
1.1 Existing Systems	
1.2 Proposed System	
1.3 Block Diagram	
1.4 Block Diagram Description	
CHAPTER II	3-6
2.1 Overview Of Embedded Systems	
2.2 Block Diagram Of An Embedded System	
2.3 Characteristics Of Embedded Systems	
2.4 Application Specific Systems	
2.5 Reactive Systems	
2.6 Distributed Systems	
2.7 Heterogeneous Architectures	
2.8 Harsh environment	

- 3.1 System Safety And Reliability
- 3.2 Control Of Physical Systems
- 3.3 Small And Low Weight
- 3.4 Cost Sensitivity
- 3.5 Power Management
- 3.6 Current Sensors
- 3.7 Current Sensing Principles
- 3.8 Passive Element Based Current Sensing Techniques
  - 3.8.1 Sense Resistors
    - 3.8.2 Low value in order to minimize power losses
    - 3.8.3 Low inductance because of high di/dt.
    - 3.8.4 Tight tolerance
    - 3.8.5 Low temperature coefficient for accuracy
    - 3.8.6 High peak power rating to handle short duration high current pulses
    - 3.8.7 High temperature rating for reliability

- 4. CURRENT SENSING TECHNIQUES
  - 4.1 Current Sensing With a Copper Resistor
  - 4.2 MOSFET-R
  - 4.3 Sense-FET Technique
- 4.4 Average Current
- 4.5 Filter-Sense the inductor
- 4.6 Magentic Field Based Sensing Technique
  - 4.6.1 Hall Effect Sensors
  - 4.6.2 Open Loop Hall effect technology
  - 4.6.3 Closed Loop Hall effect technology
  - 4.6.4 Electronic technology
  - 4.6.5 Rogowski Coil
  - 4.6.6 Transformer Technique

4.6.7 Fiber optic current sensors	
4.7 Voltage Sensors	
CHAPTER V	19-31
5.1 Introduction to controllers	
5.2 Basic arduino	
5.3 Digital	
5.4 Things to remember about digital:	
5.5 Analog	
5.6 Things to remember about analog	
5.7 Output signals	
5.8 Things to remember about output	
5.9 Input signals	
5.10 Software tips	
5.11 Arduino s3v3 new features	
CHAPTER-VI	32-36
6.1 Intoduction to Node MCU	
6.1.1 How to start with NodeMCU?	
6.1.2 How to write codes for NodeMCU?	
6.2 Arduino IDE	
CHAPTER-VII	32-42
7.1 Why is a relay used	
7.2 Relay Design	
7.3 How relay works?	
7.4 Relay Basics	
7.5 Pole and Throw	
7.6 Relay Applications	
7.7 Application of Overload Relay	
7.8 Relay Selection	
CHAPTER-VIII	43-45
8.1 What isIoT?	



8.2 Create a Channel

CONCLUSIONS

## LIST OF FIGURES

S.NO	FIGURE NAME	PAGE NO
Fig:1.1	block diagram of power management and controlled sockets	01
Fig:2.1	block diagram of a typical embedded system	04
Fig 2.2	Embedded Systems having Heterogeneous Architectures	06
Fig:3.1	current sensor	08
Fig:3.2	current sensing principles	09
Fig:4.1	Voltage Sensor	17
Fig:5.1	Arduino UNO	18
Fig:5.2	Pin description of arduino	20
Fig:6.1	Node MCU	30
Fig:6.2	USB printer cable	32
Fig:6.2.1	path to select blink	32
Fig:6.2.2	path to select board type	33
Fig:6.2.3	path to select COM	33
Fig:7.1	Electromagnetic relay	35
Fig:7.2	Schematic diagram of relay	35
Fig:7.3	4-pin relay	36
Fig:8.1	Block diagram of IOT	40

## CHAPTER-I

### INTRODUCTION

The most critical problem faces by today's world is irregular power. People in many countries don't get the primary needs of lights, fans, etc. Researchers expect the capabilities of existing energy production will fail to meet future demand without new energy sources. We can make use of available power efficiently. A system can be created to achieve efficient use of power which monitors the environment and controls the power device and turns ON only when needed. The electrical parameters like voltage, current and frequency from smart grid can be acquired remotely and send these real time values using IOT.

#### 1.1 EXISTING SYSTEMS

- EB electric meter based

#### 1.2 PROPOSED SYSTEM

Power consumption monitoring system that can measure the power usage of each of the loads individually. This system is designed around nodemcucumicrocontroller board. If the overall consumption goes beyond a specified level user will be notified about this and fine will be added if still consumption is not reduced. This system can be used for detecting faulty electrical devices in a household that is consuming unusual amount of power.

#### 1.3 BLOCK DIAGRAM

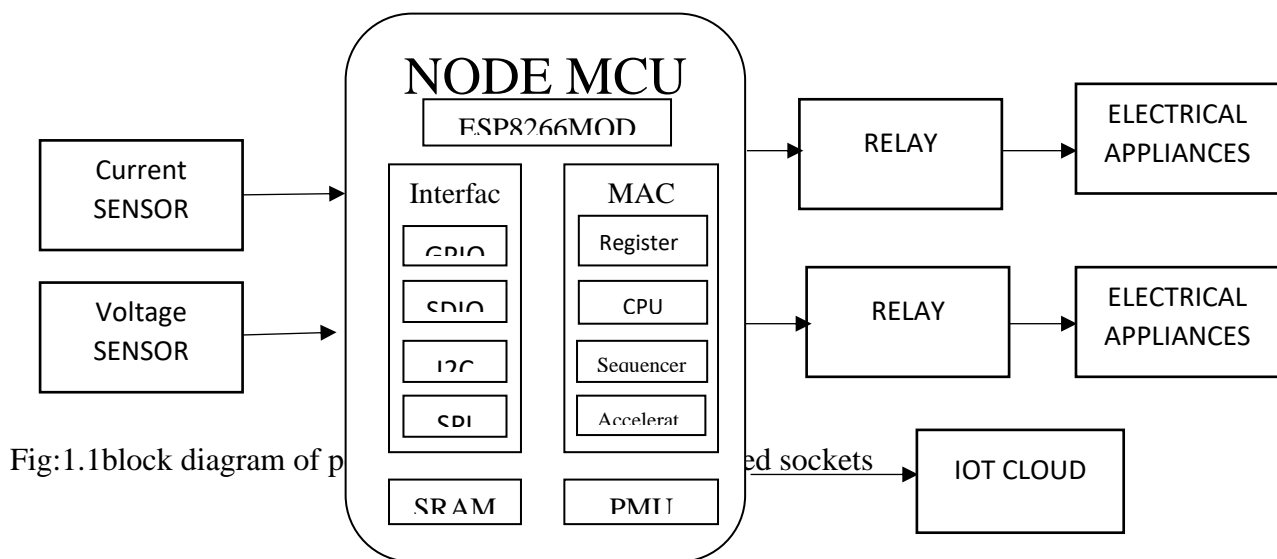


Fig:1.1block diagram of p

#### **1.4 BLOCK DIAGRAM DESCRIPTION**

- Nodemcu is the controller board used here
- Every other component is connected to nodemcu
- It reads the sensor values through an external ADC IC
- All the loads are connected to it using relays
- Relays are connected to GPIO pins in nodemcu
- Nodemcu is a has a wifiSoC called ESP8266
- It is used for connecting nodemcu to a WiFi network

## **CHAPTER-II**

### **2.1 OVERVIEW OF EMBEDDED SYSTEMS**

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

In general, "embedded system" is not an exactly defined term, as many systems have some element of programmability. For example, Handheld computers share some elements with embedded systems — such as the operating systems and microprocessors which power them — but are not truly embedded systems, because they allow different applications to be loaded and peripherals to be connected.

Embedded systems provide several functions

- Monitor the environment; embedded systems read data from input sensors. This data is then processed and the results displayed in some format to a user or users
- Control the environment; embedded systems generate and transmit commands for actuators.
- Transform the information; embedded systems transform the data collected in some meaningful way, such as data compression/decompression

Although interaction with the external world via sensors and actuators is an important aspect of embedded systems, these systems also provide functionality specific to their applications. Embedded systems typically execute applications such as control laws, finite state machines, and signal processing algorithms. These systems must also detect and react to faults in both the internal computing environment as well as the surrounding electromechanical systems.

## 2.2 BLOCK DIAGRAM OF AN EMBEDDED SYSTEM

An embedded system usually contains an embedded processor. Many appliances that have a digital interface -- microwaves, VCRs, cars -- utilize embedded systems. Some embedded systems include an operating system. Others are very specialized resulting in the entire logic being implemented as a single program. These systems are embedded into some device for some specific purpose other than to provide general purpose computing . A typical embedded system is shown in Fig 1.1

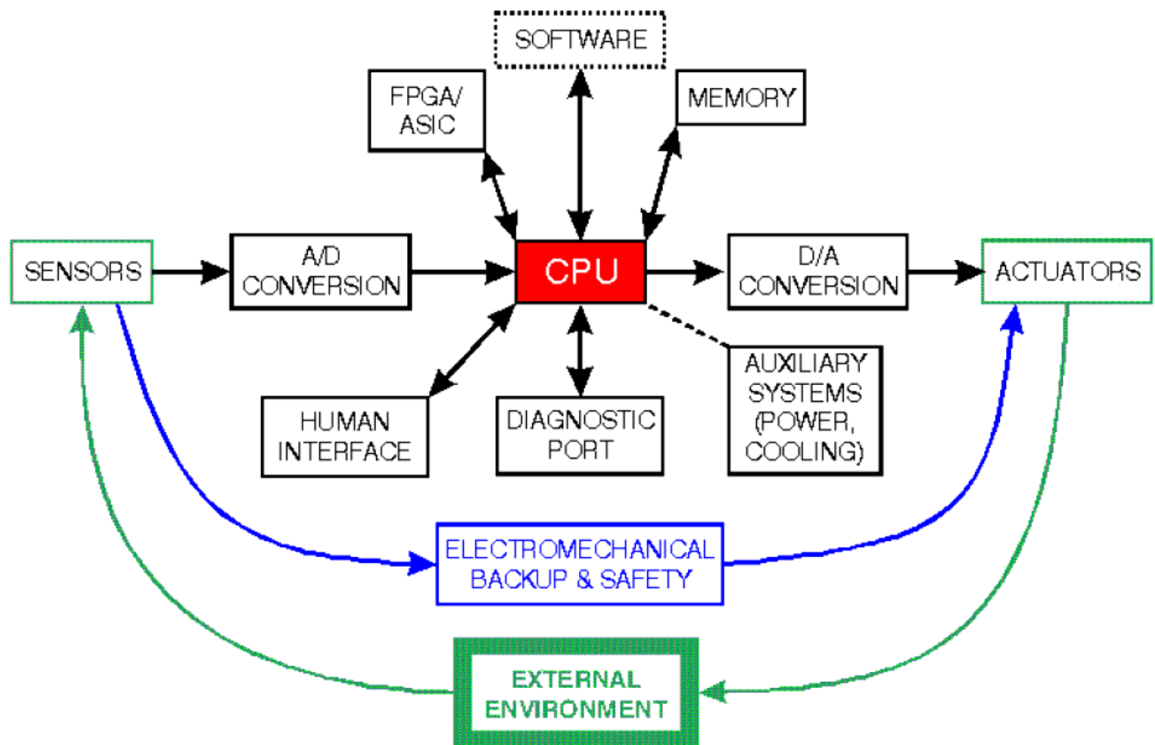


Fig:2.1 BLOCK DIAGRAM OF A TYPICAL EMBEDDED SYSTEM

## 2.3 CHARACTERISTICS OF EMBEDDED SYSTEMS

Embedded systems are characterized by a unique set of characteristics. Each of these characteristics imposed a specific set of design constraints on embedded systems designers. The challenge to designing embedded systems is to conform to the specific set of constraints for the application.

## 2.4 APPLICATION SPECIFIC SYSTEMS

Embedded systems are not general-purpose computers. Embedded system designs are optimized for a specific application. Many of the job characteristics are known before the hardware is designed. This allows the designer to focus on the specific design constraints of a well-defined application. As such, there is limited user



reprogram ability. Some embedded systems, however, require the flexibility of reprogram ability. Programmable DSPs are common for such applications.

## **2.5 REACTIVE SYSTEMS**

As mentioned earlier, a typical embedded systems model responds to the environment via sensors and control the environment using actuators. This requires embedded systems to run at the speed of the environment. This characteristic of embedded system is called “reactive”. Reactive computation means that the system (primarily the software component) executes in response to external events. External events can be either periodic or aperiodic. Periodic events make it easier to schedule processing to guarantee performance. Aperiodic events are harder to schedule. The maximum event arrival rate must be estimated in order to accommodate worst case situations. Most embedded systems have a significant reactive component. One of the biggest challenges for embedded system designers is performing an accurate worst case design analysis on systems with statistical performance characteristics (e.g., cache memory on a DSP or other embedded processor). Real time system operation means that the correctness of a computation depends, in part, on the time at which it is delivered. Systems with this requirement must often design to worst case performance. But accurately predicting the worst case may be difficult on complicated architectures. This often leads to overly pessimistic estimates erring on the side of caution. Many embedded systems have a significant requirement for real time operation in order to meet external I/O and control stability requirements. Many real-time systems are also reactive systems.

## **2.6 DISTRIBUTED SYSTEMS**

A common characteristic of an embedded system is one that consists of communicating processes executing on several CPUs or ASICs which are connected by communication links. The reason for this is economy. Economical 4 8-bit microcontrollers may be cheaper than a 32-bit processors. Even after adding the cost of the communication links, this approach may be preferable. In this approach, multiple processors are usually required to handle multiple time-critical tasks. Devices under control of embedded systems may also be physically distributed.

## **2.7 HETEROGENEOUS ARCHITECTURES**

Embedded systems often are composed of heterogeneous architectures (Fig 2.7). They may contain different processors in the same system solution. They may also

be mixed signal systems. The combination of I/O interfaces, local and remote memories, and sensors and actuators makes embedded system design truly unique. Embedded systems also have tight design constraints, and heterogeneity provides better design flexibility.



Fig 2.2 Embedded Systems having Heterogeneous Architectures

### 2.8 Harsh environment

Many embedded systems do not operate in a controlled environment. Excessive heat is often a problem, especially in applications involving combustion (e.g., many transportation applications). Additional problems can be caused for embedded computing by a need for protection from vibration, shock, lightning, power supply fluctuations, water, corrosion, fire, and general physical abuse.

## **CHAPTER-III**

### **3.1 SYSTEM SAFETY AND RELIABILITY**

As embedded system complexity and computing power continue to grow, they are starting to control more and more of the safety aspects of the overall system. These safety measures may be in the form of software as well as hardware control. Mechanical safety backups are normally activated when the computer system loses control in order to safely shut down system operation. Software safety and reliability is a bigger issue. Software doesn't normally "break" in the sense of hardware. However software may be so complex that a set of unexpected circumstances can cause software failures leading to unsafe situations. Discussion of this topic is outside the scope of this book, but the challenges for embedded designers include designing reliable software and building cheap, available systems using unreliable components. The main challenge for embedded system designers is to obtain low-cost reliability with minimal redundancy.

### **3.2 CONTROL OF PHYSICAL SYSTEMS**

One of the main reasons for embedding a computer is to interact with the environment. This is often done by monitoring and controlling external machinery. Embedded computers transform the analog signals from sensors into digital form for processing. Outputs must be transformed back to analog signal levels. When controlling physical equipment, large current loads may need to be switched in order to operate motors and other actuators. To meet these needs, embedded systems may need large computer circuit boards with many non-digital components. Embedded system designers must carefully balance system tradeoffs among analog components, power, mechanical, network, and digital hardware with corresponding software.

### **3.3 SMALL AND LOW WEIGHT**

Many embedded computers are physically located within some larger system. The form factor for the embedded system may be dictated by aesthetics. For example, the form factor for a missile may have to fit inside the nose of the missile. One of the challenges for embedded systems designers is to develop non-rectangular geometries for certain solutions. Weight can also be a critical constraint. Embedded automobile control systems, for example, must be light weight for fuel economy. Portable CD players must be light weight for portability purposes.

### **3.4 COST SENSITIVITY**

Cost is an issue in most systems, but the sensitivity to cost changes can vary dramatically in embedded systems. This is mainly due to the effect of computer costs

have on profitability and is more a function of the proportion of cost changes compared to the total system cost.

### **3.5 POWER MANAGEMENT**

Embedded systems have strict constraints on power. Given the portability requirements of many embedded systems, the need to conserve power is important to maintain battery life as long as possible. Minimization of heat production is another obvious concern for embedded systems.

### **3.6 CURRENT SENSORS**

Current measurement is of vital importance in many power and instrumentation systems. Traditionally, current sensing was primarily for circuit protection and control. However, with the advancement in technology, current sensing has emerged as a method to monitor and enhance performance.



Fig:3.1 current sensor

Knowing the amount of current being delivered to the load can be useful for wide variety of applications. Current sensing is used in wide range of electronic systems, viz., Battery life indicators and chargers, 4-20 mA systems, over-current protection and supervising circuits, current and voltage regulators, DC/DC converters, ground fault detectors, programmable current sources, linear and switch-mode power supplies, communications devices, automotive power electronics, motor speed controls and overload protection, etc.

### 3.7 CURRENT SENSING PRINCIPLES

A current sensor is a device that detects and converts current to an easily measured output voltage, which is proportional to the current through the measured path.

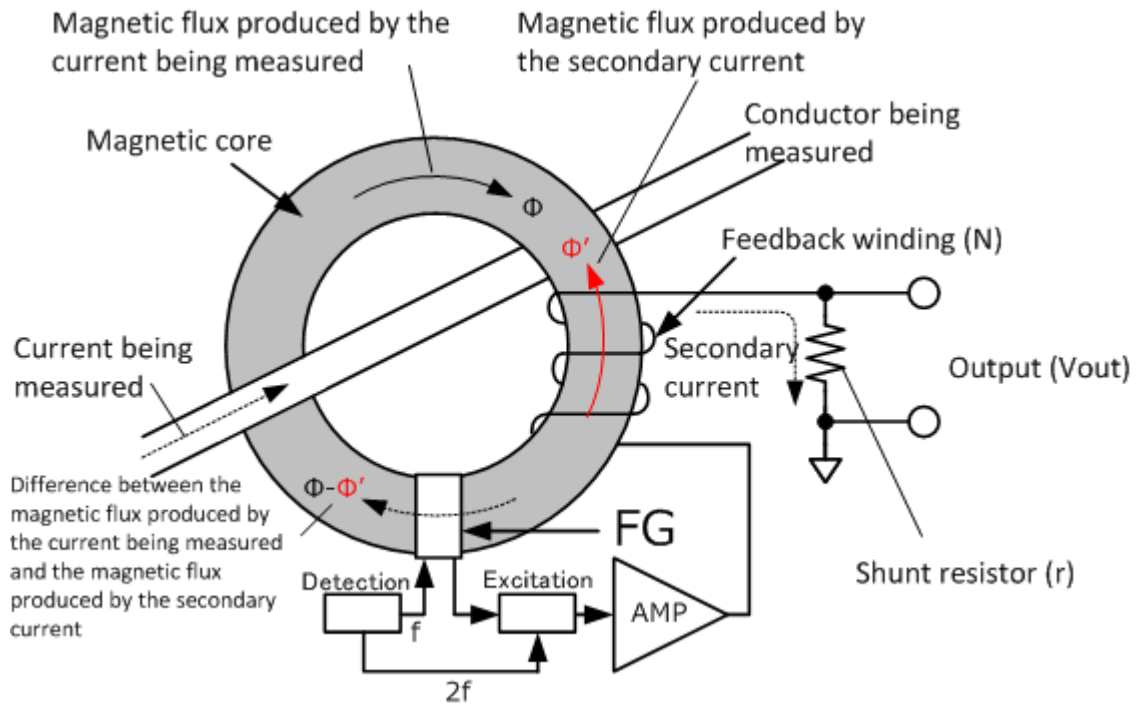


Fig:3.2 current sensing principles

When a current flows through a wire or in a circuit, voltage drop occurs. Also, a magnetic field is generated surrounding the current carrying conductor. Both of these phenomena are made use of in the design of current sensors. Thus, there are two types of current sensing: direct and indirect. Direct sensing is based on Ohm's law, while indirect sensing is based on Faraday's and Ampere's law.

Direct Sensing involves measuring the voltage drop associated with the current passing through passive electrical components.

Indirect Sensing involves measurement of the magnetic field surrounding a conductor through which current passes.

Generated magnetic field is then used to induce proportional voltage or current which is then transformed to a form suitable for measurement and/or control system.

## **3.8 PASSIVE ELEMENT BASED CURRENT SENSING TECHNIQUES**

### **3.8.1 Sense Resistors**

Current sensing means developing a voltage signal which is representative of the current flowing at the particular place of interest in the circuit. The traditional way of current sensing introduces a resistor in the path of the current to be sensed. The sense resistor can be placed in series with the inductor, switches, and the load. Thus, a current sensing resistor should be considered as a current-to-voltage converter.

The current sensing resistor should have following attributes

### **3.8.2 Low value in order to minimize power losses**

Value of the current sense resistors primarily depend upon the voltage threshold of the following circuitry which is going to operate based upon the sensed current information. In circuits where amplification is available, emphasis is to minimize the voltage drop across the resistor.

Typical resistance values utilized in various control ICs are 20m $\Omega$  to 25m $\Omega$ .

### **3.8.3 Low inductance because of high di/dt.**

Any inductance in the resistor, when exposed to high slew rate (di/dt), an inductive step voltage is superimposed upon the sense voltage and may be a cause of concern in many circuits. Hence sense resistors should have very low inductance.

### **3.8.4 Tight tolerance**

For maximizing the current supply within the limit of acceptable current, the tolerance of the sense resistor must be  $\pm 1\%$  or tighter.

### **3.8.5 Low temperature coefficient for accuracy**

Normally specified in units of parts per million per degree centigrade (ppm/ $^{\circ}\text{C}$ ), temperature coefficient of resistance (TCR) is an important parameter for accuracy. Resistors with TCRs closer to zero, in the entire operating range should be used.

### **3.8.6 High peak power rating to handle short duration high current pulses.**

Power rating is a driving factor for the selection of appropriate technology for sense resistors. Though the device may be intended to sense DC current, it may often experience transients.

Power derating curve provides allowable power at different temperatures. But peak power capability is a function of energy; hence energy rating curve should be taken into account.



### **3.8.7 High temperature rating for reliability**

Pros and Cons of current sensing resistors include:

**Pros:**

- Low cost
- High measurement accuracy
- Measurable current range from very low to medium
- Capability to measure DC or AC current

**Cons:**

- Introduces additional resistance into the measured circuit path, which may increase source output resistance and result in undesirable loading effect.
- Power loss due to power dissipation. Therefore, current sensing resistors are rarely used beyond the low and medium current sensing applications.

## CHAPTER-IV

### CURRENT SENSING TECHNIQUES

#### 4.1 Current Sensing With a Copper Resistor

Instead of using a separate discrete resistor for current sensing, it is often useful to use copper trace in a printed circuit board as a low-value resistor for the purpose of current sensing. This method will have smaller power loss and also it will save on the efforts towards buying and installing a discrete resistor. But, since resistance of copper is very low, sensed voltage will also be very requiring significant amplification, or increase in the length of the resistor at the cost of PCB area. Another factor of significance is the TCR of copper ( $0.39 \% / ^\circ\text{C}$ ) which amounts to approx. 20 % change for 50% temperature rise.

#### 4.2 MOSFET-R

MOSFETs act as resistors when they are “on” and are biased in the Ohmic (nonsaturated) region. The current is determined by sensing the voltage across the drain-source of the MOSFET, if  $R_{DS}$  of the MOSFET is known. The main drawbacks of this technique are low accuracy and switching noise from non-zero gate currents during transients, nonlinearity of  $R_{DS}$  of the MOSFET, dependence of  $R_{DS}$  on  $C_{ox}$ ,  $V_T$  and temperature.

#### 4.3 Sense-FET Technique

This method is a practical technique used for current sensing in many new power MOSFET applications. A current-sensing FET in parallel with the power MOSFET is used. The effective width of the sense MOSFET (sense-FET) is significantly smaller ( $\sim 10000$  times) than the power FET. The accuracy of the sense-FET technique is about  $\pm 20\%$ .

#### • Sensorless (Observer) Approach

This method uses the inductor voltage to measure the inductor current. Since the voltage-current relation of the inductor is  $v = L \cdot di/dt$ , the inductor current can be estimated by integrating the voltage over time. To avoid saturation in the integrator, it is reset periodically, and therefore only AC ripple current is estimated. The value of  $L$  also should be known for this technique.

#### 4.4 Average Current

This current-sensing technique uses an RC low-pass filter at the junction of the converter switches. Therefore, the voltage at filter output capacitor is the average voltage of the phase node. Consequently, the differential voltage at the input of the amplifier is the DC voltage across the inductor.  $V_{I-Average}$  is a function of  $R_{ESR}$  (Inductor Resistance) and  $I_{L\_DC}$  (Inductor DC Current).

#### 4.5 Filter-Sense the inductor

This current-sensing technique uses a simple low-pass RC network to filter the voltage across the inductor and sense the current through the equivalent series resistance (ESR) of the inductor.

#### 4.6 Magnetic Field Based Sensing Technique

While resistive current sensing techniques are useful in many applications, they suffer from three inherent drawbacks:

- Supply-line voltage drop
- Insertion power loss
- Common-mode errors

Most of these issues can be taken care of when sensing low to moderate amounts of current on low-voltage supply lines, but they can become significant as either currents or voltages increase. When trying to measure currents at higher levels ( $>10A$ ) or where the supply line is at a high (e.g. 48V) voltage, preferred solution is to use magnetic current sensors. One of the significant and obvious benefits of using magnetic coupling for sensing current is the electrical isolation.

In these sensors, magnetically permeable core is used which concentrates the conductor's magnetic field generated due to flow of current in the conductor. The magnetic field is sensed using different methods:

##### 4.6.1 Hall Effect Sensors

The Hall Effect principle states that when a current carrying conductor is placed in a magnetic field, a voltage will be generated perpendicular to the direction of the field and the flow of current.

When a constant current is passed through a thin sheet of semiconducting material, there is no potential difference at the output contacts if the magnetic field is zero. However, when a perpendicular magnetic field is present, the current flow is distorted. The uneven distribution of electron density creates a potential difference across the output terminals. This voltage is called the Hall voltage. If the input current is held

constant the Hall voltage will be directly proportional to the strength of the magnetic field.

The Hall voltage is a low level signal of the order of 20 to 30 microvolt's in a magnetic field of one gauss. A signal of this magnitude requires a low noise, high impedance, moderate gain amplifier.

Hall sensors are based on following technologies. They can be used for measurement of DC, AC and impulse currents, with galvanic isolation between primary and secondary circuits

#### **4.6.2 Open Loop Hall effect technology**

Current sensors based on this technology are electronic transformers. Primary current  $I_p$  creates magnetic flux and the hall probe placed in the airgap of the magnetic circuit provides a voltage proportional to the magnetic flux. This voltage itself is proportional to  $I_p$  is amplified and is used for further processing.

The linearity of the open-loop sensor is determined by the characteristics of the magnetic core and the Hall generator. Offset drift over temperature is determined primarily by the temperature sensitivity of the Hall generator.

#### **4.6.3 Closed Loop Hall effect technology**

Current sensors based on this technology are also electronic transformers. Primary current  $I_p$  creates magnetic flux and the hall probe placed in the airgap of the magnetic circuit provides a voltage proportional to the magnetic flux. This voltage is fed into a push-pull driver stage that drives the coil wound in series opposition on the magnetic core. Thus, it creates a magnetic field equal to and opposite to the field of the sensed current: maintaining the core flux level near zero. The secondary current cancels out the primary magnetic flux that created it (contra reaction). The output of the closed loop sensor is proportional to the aperture current and the number of turns of the coil. Closed loop approach allows significant improvements in sensor performance by eliminating the influence of non-linearities in the magnetic core and by reducing the effects of temperature sensitivity in the Hall element

#### **4.6.4 Electronic technology**

In contrast to open loop and closed loop technology, they do not use magnetic circuit. Primary current  $I_p$  creates magnetic flux and different hall probes included in the sensor provides a voltage proportional to the magnetic flux.

Hall Effect based sensors do not suffer from Insertion loss (and related heating, etc.). However, frequency range, cost, DC offset, and external power represent the potential

disadvantages of Hall-effect IC technology when compared to the resistive sensing methods.

#### **4.6.5 Rogowski Coil**

This device consists of a single-layer coil, uniformly wound on a non-magnetic core which is either flexible or formed into a circle which surrounds the conductor of the current to be measured. An AC current through the wire changes polarity. Changing polarity causes expanding and collapsing magnetic field which in turn induces current in the windings. The current is then processed to make suitable them suitable for measurement or control system.

Practical implementations of this technique typically also incorporate a low-frequency roll-off to eliminate thermal noise and drift. The major benefit of a Rogowski Coil is that since the core is effectively air, there is no magnetic material to saturate and the coil's output remains linear for extremely high currents. This device is used for measuring high-energy current pulses or transients with high-frequency harmonic content since the upper bandwidth can extend into the megahertz range.

#### **4.6.6 Transformer Technique**

Transformer Technique is an extension of the Rogowski Coil technology wherein the air core is replaced with a material which concentrates the magnetic flux inside the coil. With the flux contained within the coil instead of passing through it, a direct relationship between the coil current and the current in the conductor generating the field is obtained.

Current sense transformers provide important benefits over simple resistive sensing. They offer electrical isolation, avoid insertion loss and they do not require external power. The lower power dissipation of a current sense transformer allows a much higher signal level, significantly improving the signal-to-noise environment of the control system.

Current transformers (CT) are commonly used in high-power systems to measure current. Major drawbacks are large size and cost and also the inability to detect DC Current.

#### **4.6.7 Fiber optic current sensors**

Development of magneto-optic sensors has demonstrated their use in current and magnetic field measurements applications. The principle of magneto-optic effects is based on the interaction between magnetic field and the phenomenon of light refraction and reflection in transparent medium and on its surface. They offer inherent immunity against EMI and good isolation against high voltages

Current sensors employ Faraday magneto-optic effect. Faraday effect causes the electromagnetic wave polarization rotation due to the magnetic field intensity in transparent material. The magnetic field induced by the current leads to an angle rotation in the plane of polarization of the linearly polarized light propagating across the ferromagnetic material. The rotation is detected by polarisers and analyzers at the input and output. By monitoring the rotation of incident polarization, magnetic field and hence current can be estimated. The magnitude of the effect depends further on the magneto-optic material constant (Verdet constant) and on the interaction length through which the wave travels in magnetized material.

#### **4.7 VOLTAGE SENSORS**

This sensor is used to monitor, calculate and determine the voltage supply. This sensor can determine the AC or DC voltage level. The input of this sensor can be the voltage whereas the output is the switches, analog voltage signal, a current signal, an audible signal, etc. Some sensors provide sine waveforms or pulse waveforms like output & others can generate outputs like AM (Amplitude Modulation), PWM (Pulse Width Modulation) or FM (Frequency Modulation). The measurement of these sensors can depend on the voltage divider. This sensor includes input and output. The input side mainly includes two pins namely positive and negative pins. The two pins of the device can be connected to the positive & negative pins of the sensor. The device positive & negative pins can be connected to the positive & negative pins of the sensor. The output of this sensor mainly includes supply voltage (Vcc), ground (GND), analog o/p data



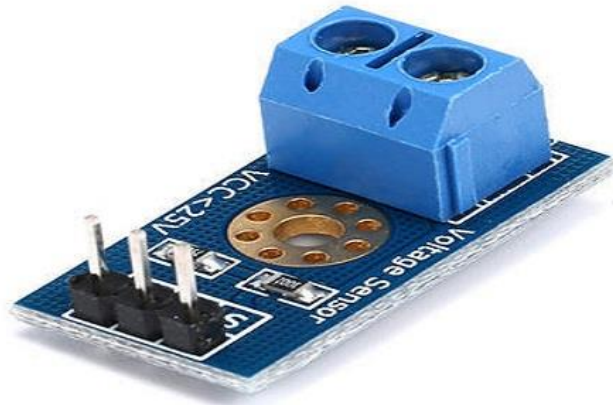


Fig:4.1 Voltage Sensor

This sensor includes input and output. The input side mainly includes two pins namely positive and negative pins. The two pins of the device can be connected to the positive & negative pins of the sensor. The device positive & negative pins can be connected to the positive & negative pins of the sensor. The output of this sensor mainly includes supply voltage (Vcc), ground (GND), analog o/p data.

## CHAPTER-V

### 5.1 INTRODUCTION TO CONTROLLERS

The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output(I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

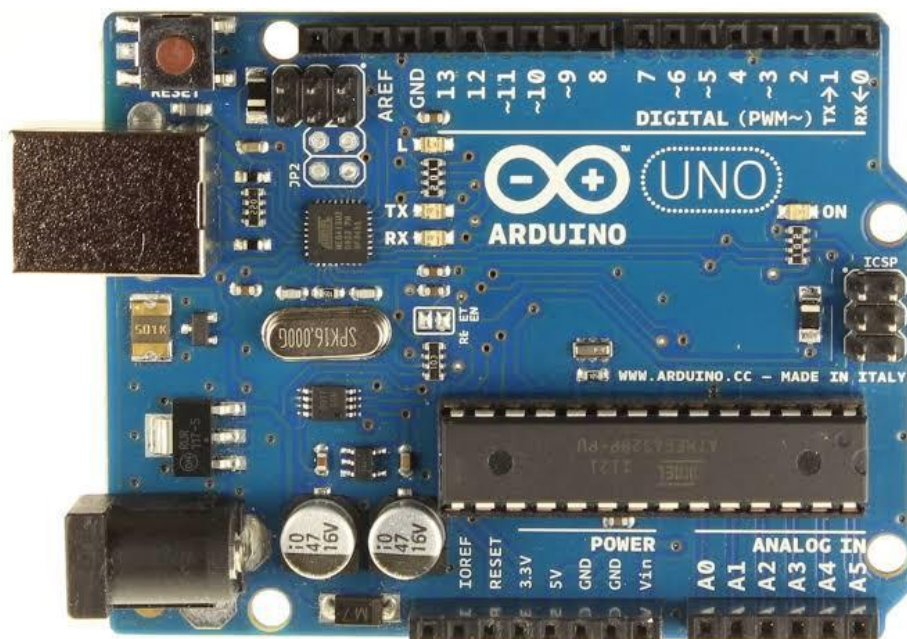


Fig:5.1 Arduino UNO

- The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

- While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

## 5.2 BASIC ARDUINO

**setup( ):** A function present in every Arduino sketch. Run once before the loop( ) function. Often used to set pinmode to input or output. The setup( ) function looks like:

```
void setup( ){
    //code goes here
}
```

**loop( ):** A function present in every single Arduino sketch. This code happens over and over again. The loop( ) is where (almost) everything happens. The one exception to this is setup( ) and variable declaration. ModKit uses another type of loop called “forever( )” which executes over Serial. The loop( ) function looks like:

```
void loop( ){
    //code goes here
}
```

**input:** A pin mode that intakes information.

**output:** A pin mode that sends information.

**HIGH:** Electrical signal present (5V for Uno). Also ON or True in boolean logic.

**LOW:** No electrical signal present (0V). Also OFF or False in boolean logic.

**digitalRead:** Get a HIGH or LOW reading from a pin already declared as an input.

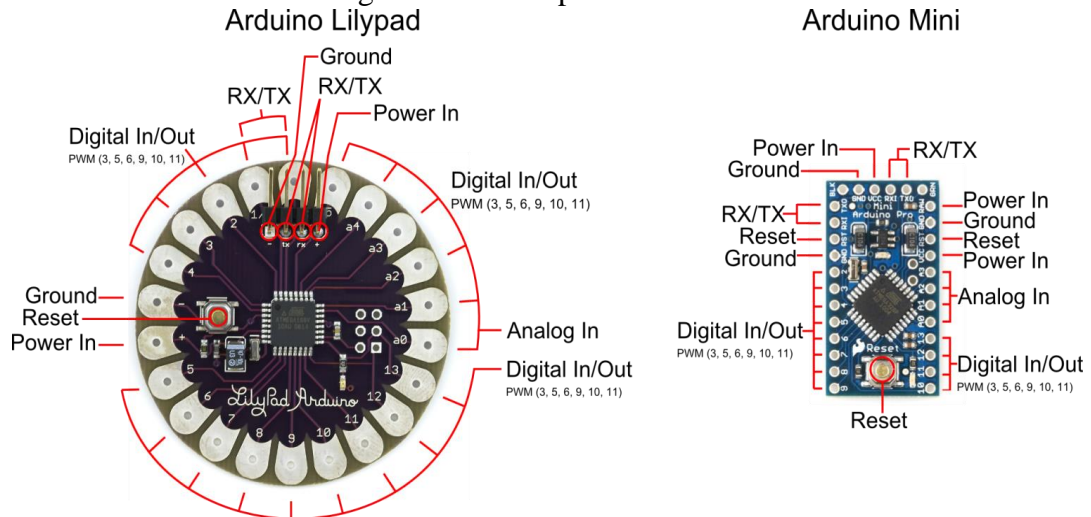
**digitalWrite:** Assign a HIGH or LOW value to a pin already declared as an output.

**analogRead:** Get a value between or including 0 (LOW) and 1023 (HIGH). This allows you to get readings from analog sensors or interfaces that have more than two states.

**analogWrite:** Assign a value between or including 0 (LOW) and 255 (HIGH). This allows you to set output to a PWM value instead of just HIGH or LOW.

**PWM:** Stands for Pulse-Width Modulation, a method of emulating an analog signal through a digital pin. A value between or including 0 and 255. Used with `analogWrite`.

Fig:5.2 Pin description of arduino



If you wish to use the voltage divider as a sensor reading device first you need to know the maximum voltage allowed by the analog inputs you are using to read the signal. On an Arduino this is 5V. So, already we know the maximum value we need for  $V_{out}$ . The  $V_{in}$  is simply the amount of voltage already present on the circuit before it reaches the first resistor. You should be able to find the maximum voltage your sensor outputs by looking on the Datasheet, this is the maximum amount of voltage your sensor will let through given the voltage in of your circuit. Now we have exactly one variable left, the value of the second resistor. Solve for  $R_2$  and you will have all the components of your voltage divider figured out! We solve for  $R_1$ 's highest value because a smaller resistor will simply give us a smaller signal which will be readable by our analog inputs.

Powering an analog Reference is exactly the same as reading a sensor except you have to calculate for the Voltage Out value you want to use as the analog Reference.

All of the electrical signals that the Arduino works with are either Analog or Digital. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

Lilypad is designed for use with conductive thread instead of wire and the Arduino Mini is simply a smaller package without the USB, Barrel Jack and Power Outs. It depends on what you want to do with it really. There are two different purposes outlined above for the voltage divider, we will go over both.

### 5.3 DIGITAL

An electronic signal transmitted as binary code that can be either the presence or absence of current, high and low voltages or short pulses at a particular frequency.

Humans perceive the world in analog, but robots, computers and circuits use Digital. A digital signal is a signal that has only two states. These states can vary depending on the signal, but simply defined the states are ON or OFF, never in between.

In the world of Arduino, Digital signals are used for everything with the exception of Analog Input. Depending on the voltage of the Arduino the ON or HIGH of the Digital signal will be equal to the system voltage, while the OFF or LOW signal will always equal 0V. This is a fancy way of saying that on a 5V Arduino the HIGH signals will be a little under 5V and on a 3.3V Arduino the HIGH signals will be a little under 3.3V.

To receive or send Digital signals the Arduino uses Digital pins # 0 - # 13. You may also setup your Analog In pins to act as Digital pins. To set up Analog In pins as Digital pins use the command:

```
pinMode(pinNumber, value);
```

where pinNumber is an Analog pin (A0 – A5) and value is either INPUT or OUTPUT. To setup Digital pins use the same command but reference a Digital pin for pinNumber instead of an Analog In pin. Digital pins default as input, so really you only need to set them to OUTPUT in pinMode. To read these pins use the command:

```
digitalRead(pinNumber);
```

where pinNumber is the Digital pin to which the Digital component is connected. The digitalRead command will return either a HIGH or a LOW signal. To send a Digital signal to a pin use the command:

```
digitalWrite(pinNumber, value);
```

where pinNumber is the number of the pin sending the signal and value is either HIGH or LOW.

The Arduino also has the capability to output a Digital signal that acts as an Analog signal, this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, #

5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

*analogWrite(pinNumber, value);*

where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

#### **5.4 THINGS TO REMEMBER ABOUT DIGITAL:**

- Digital Input/Output uses the Digital pins, but Analog In pins can be used as Digital
- To receive a Digital signal use: *digitalRead(pinNumber);*
- To send a Digital signal use: *digitalWrite(pinNumber, value);*
- Digital Input and Output are always either HIGH or LOW

All of the electrical signals that the Arduino works with are either Analog or Digital. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

#### **5.5 ANALOG**

Humans perceive the world in analog. Everything we see and hear is a continuous transmission of information to our senses. The temperatures we perceive are never 100% hot or 100% cold, they are constantly changing between our ranges of acceptable temperatures. (And if they are out of our range of acceptable temperatures then what are we doing there?) This continuous stream is what defines analog data. Digital information, the complementary concept to Analog, estimates analog data using only ones and zeros.

In the world of Arduino an Analog signal is simply a signal that can be HIGH (on), LOW (off) or anything in between these two states. This means an Analog signal has a voltage value that can be anything between 0V and 5V (unless you mess with the Analog Reference pin). Analog allows you to send output or receive input about devices that run at percentages as well as on and off. The Arduino does this by sampling the voltage signal sent to these pins and comparing it to a voltage reference signal (5V). Depending on the voltage of the Analog signal when compared to the Analog Reference signal the Arduino then assigns a numerical value to the signal somewhere between 0 (0%) and 1023 (100%). The digital system of the Arduino can then use this number in calculations and sketches.

To receive Analog Input the Arduino uses Analog pins # 0 - # 5. These pins are designed for use with components that output Analog information and can be used for Analog Input. There is no setup necessary, and to read them use the command:

*analogRead(pinNumber);*

where pinNumber is the Analog In pin to which the the Analog component is connected. The analogRead command will return a number including or between 0 and 1023.

The Arduino also has the capability to output a digital signal that acts as an Analog signal, this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, # 5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

*analogWrite(pinNumber, value);*

where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). On the Arduino UNO PWM pins are signified by a ~ sign. For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

## **5.6 THINGS TO REMEMBER ABOUT ANALOG**

- Analog Input uses the Analog In pins, Analog Output uses the PWM pins
- To receive an Analog signal use: *analogRead(pinNumber);*
- To send a PWM signal use: *analogWrite(pinNumber, value);*
- Analog Input values range from 0 to 1023 (1024 values because it uses 10 bits,  $2^{10}$ )
- PWM Output values range from 0 to 255 (256 values because it uses 8 bits,  $2^8$ )

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

## **5.7 OUTPUT SIGNALS**

Output to the Arduino pins is always Digital, however there are two different types of Digital Output; regular Digital Output and Pulse Width Modulation Output (PWM). Output is only possible with Digital pins # 0 - # 13. The Digital pins are preset as Output pins, so unless the pin was used as an Input in the same sketch, there is no reason to use the pinMode command to set the pin as an Output. Should a situation arise where it is necessary to reset a Digital pin to Output from Input use the command:

*pinMode(pinNumber, OUTPUT);*

where `pinNumber` is the Digital pin number set as Output. To send a Digital Output signal use the command:

```
digitalWrite(pinNumber, value);
```

where `pinNumber` is the Digital pin that is outputting the signal and `value` is the signal.

When outputting a Digital signal value can be either HIGH (On) or LOW (Off).

```
analogWrite(pinNumber, value);
```

where `pinNumber` is a Digital Pin with PWM capabilities and `value` is a number between 0 (0%) and 255 (100%). For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

Output can be sent to many different devices, but it is up to the user to figure out which kind of Output signal is needed, hook up the hardware and then type the correct code to properly use these signals.

## 5.8 THINGS TO REMEMBER ABOUT OUTPUT

- Output is always Digital
- There are two kinds of Output: regular Digital or PWM (Pulse Width Modulation)
- To send an Output signal use `analogWrite(pinNumber, value);` (for analog) or `digitalWrite(pinNumber, value);` (for digital)
- Output pin mode is set using the `pinMode` command: `pinMode(pinNumber, OUTPUT);`
- Regular Digital Output is always either HIGH or LOW
- PWM Output varies from 0 to 255

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

## 5.9 INPUT SIGNALS

Analog Input enters your Arduino through the Analog In pins # 0 - # 5. These signals originate from analog sensors and interface devices. These analog sensors and devices use voltage levels to communicate their information instead of a simple yes (HIGH) or no (LOW). For this reason you cannot use a digital pin as an input pin for these devices. Analog Input pins are used only for receiving Analog signals. It is only possible to read the Analog Input pins so there is no command necessary in the setup(



) function to prepare these pins for input. To read the Analog Input pins use the command:

```
analogRead(pinNumber);
```

where pinNumber is the Analog Input pin number. This function will return an Analog Input reading between 0 and 1023. A reading of zero corresponds to 0 Volts and a reading of 1023 corresponds to 5 Volts. These voltage values are emitted by the analog sensors and interfaces. If you have an Analog Input that could exceed  $V_{cc} + .5V$  you may change the voltage that 1023 corresponds to by using the Aref pin. This pin sets the maximum voltage parameter your Analog Input pins can read. The Aref pin's preset value is 5V.

Digital Input can enter your Arduino through any of the Digital Pins # 0 - # 13. Digital Input signals are either HIGH (On, 5V) or LOW (Off, 0V). Because the Digital pins can be used either as input or output you will need to prepare the Arduino to use these pins as inputs in your

setup( )function. To do this type the command:

```
pinMode(pinNumber, INPUT);
```

inside the curly brackets of the setup( ) function where pinNumber is the Digital pin number you wish to declare as an input. You can change the pinMode in the loop( )function if you need to switch a pin back and forth between input and output, but it is usually set in the setup( )function and left untouched in the loop( )function. To read the Digital pins set as inputs use the command:

```
digitalRead(pinNumber);
```

where pinNumber is the Digital Input pin number.

Input can come from many different devices, but each device's signal will be either Analog or Digital, it is up to the user to figure out which kind of input is needed, hook up the hardware and then type the correct code to properly use these signals.

X1:

DE-9 serial connector

Used to connect computer (or other devices) using RS-232 standard. Needs a serial cable, with at least 4 pins connected: 2, 3, 4 and 5. Works only when JP0 is set to 2-3 position.

DC1:

2.1 mm. power jack

Used to connect external power source. Centre positive. Voltage Regulator Works with regulated +7 to +20 volts DC (9v. to 12v. is recommended). It is possible to alternatively connect external power using 9v. pin or 5v. pin. (see POWER PINOUT) ICSP:

2x3 pin header Used to program Atmega with bootloader. The number 1 on both sides of the board indicates cable pin1 position. Used to upload sketches on Atmega ICs without bootloader (available only in Arduino IDE versions 0011 and 0012).

JP0

3 pins jumper When in position 2-3, this jumper enables serial connection (through X1 connector) to/from computer/devices. Use this as default position. When in position 1-2, it disables serial communication, and enables external pull-down resistors on pin0 (RX) and pin1 (TX). Use this only to prevent noise on RX (that seems incoming data to Atmega), that sometimes makes sketch not starting. When removing this jumper, serial communication is disabled, and pin0 and pin1 work as a normal (floating) digital pin. Useful when more digital pins are needed, but only when serial communication is not necessary. External pull-down/pull-up resistor is required.

JP4

2 pins jumper When in position 1-2, this jumper enables auto reset feature, useful when uploading a sketch to Arduino, resetting Atmega automatically. It makes unnecessary to press reset button (S1) when uploading sketches. Be sure that computer COM Port speed is set to 19200bps otherwise auto reset will not work properly. If removed, disables auto reset feature. Very useful to prevent undesired Atmega reset when using sketches that needs serial communication. Auto reset works with DTR pulse on serial pin4. Sometimes Arduino senses a DTR pulse when connecting X1 (serial connector) and some softwares sends a DTR pulse when it starts or when it closes, that makes Atmega reset when not desired.

S1

Tactile button This button resets Atmega, to restart uploaded sketch or to prepare Arduino to receive a sketch through serial connector (when auto reset is not active).

LEDS

Indicative leds POWER led Turns on when Arduino is powered through DC1, +9v. pin or +5v. pin. RX led Blinks when receiving data from computer/device through

serial connection. TX led Blinks when sending data to computer/device through serial connection. L led This led is connected to digital pin13 with a current limiter resistor (that doesn't affect pin13). Useful to test sketches. It is normal to blink when bootloading too.

## POWER PINOUT

6 pin header

### RST pin

Makes Atmega reset when connected to GND. Useful for Shield Boards, or to connect external reset.

### NC pin

This pin is not connected in Arduino S3v3. Arduino Diecimila has a 3.3 volts pin in the same position.

### +9v. pin

When Arduino DC1 is powered (with battery or DC adaptor), this pin is used as Vout, with the same voltage supplied on DC1 (see DC1), minus 0,7 volts. The total supplied current depends on external power source capacity When Arduino DC1 is not powered, +9v. pin can be used as Vin, connecting it to a external regulated power source (+7 to +20 volts) and connecting 0v. pin to external power source GND. In this case, +5v. pin

can be used as Vout, supplying +5 volts. +5v. pin When Arduino DC1 is powered (with battery or DC adaptor), +5v. pin supplies +5 volts as a Vout pin. The total supplied current depends on Voltage Regulator (7805 supplies up to 1A). This applies only to +5v. pin: Atmega in/out pins only supplies max. 40mA on each pin. When Arduino DC1 is not powered, this pin can be used as Vin, connecting it to a regulated +5v. and connecting 0v. pin to power source GND. In this case, +9v. pin is inactive. 0v. pin (GND) Two 0v. pins between +5v. and +9v. / One

### 0v. pin

beside AREF pin. When Arduino DC1 is powered, 0v. pin supplies 0 volts reference (GND) for +5v. pin and +9v. pin. When DC1 is not powered, and Arduino is powered through +5v. pin or +9v. pin, 0v. pin must be used as GND reference, connecting it to the external power source GND.

### GND pin

see 0v. pin (GND).

### AREF pin

The AREF can be set to AVcc (default), internal 2.56 volts (Atmega8), internal 1.1 volts (Atmega168), or external AREF. In case of AVcc or internal AREF, AREF pin can be used to attach an external capacitor to decouple the signal, for better noise performance. In case of external AREF, AREF pin is used to attach the external reference voltage. Remember that it is necessary to change the fuses (wiring.c file), and re-upload sketch, before connecting external voltage to AREF

### **5.10 SOFTWARE TIPS**

When bootloading an Atmega8 chip with Arduino 0010, there is a command (-i800) that makes bootloader delay 10 minutes. So, if you need to use bootloader, use command line instead of IDE, removing “-i800” command and adding “-F” command, or use Arduino 0007 IDE. To upload sketches Arduino 0010 works fine.

### **5.11 ARDUINO S3v3 NEW FEATURES**

- full compatible with Shield Boards (Version 2 is the only Arduino Board not compatible with Shield Boards because of ICSP header wrong position, and tall components);
- AVcc LP filter (to reduce diodes, transistors, LEDs, capacitors) has the same board orientation (to make easier to mount with less mistakes);
- no wires between pads, more space between wires, larger wires, larger pads (better for etching, soldering and drilling, with no short circuits, soldering bridges or open wires in corrosion);
- only 3 wire bridges;
- electrolytic capacitor (in serial to TTL circuit) changed to bipolar type (to avoid inverted voltage problem when serial cable is not connected);
- All jumpers are right angle type, to allow Shield Boards use.

## CHAPTER-VI

### 6.1 INTRODUCTION TO NODE MCU

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.



Fig:6.1 Node MCU

Since NodeMCU is open source platform, their hardware design is open for edit/modify/build. Node MCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 WiFi Module There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.

NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc.

Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

#### 6.1.1 How to start with NodeMCU?

NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols.

To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement.

### **6.1.2 How to write codes for NodeMCU?**

After setting up ESP8266 with Node-MCU firmware, let's see the IDE (Integrated Development Environment) required for development of NodeMCU.

#### **NodeMCU with ESPlorer IDE**

Lua scripts are generally used to code the NodeMCU. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.

#### **NodeMCU with Arduino IDE**

Here is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCU using Arduino development environment. This makes easy for Arduino developers than learning new language and IDE for NodeMCU.

#### **Difference in using ESPlorer and Arduino IDE**

Well, there is a programming language difference we can say while developing application for NodeMCU using ESPlorer IDE and Arduino IDE.

We need to code in C\C++ programming language if we are using Arduino IDE for developing NodeMCU applications and Lua language if we are using ESPlorer IDE.

Basically, NodeMCU is Lua Interpreter, so it can understand Lua script easily. When we write Lua scripts for NodeMCU and send/upload it to NodeMCU, then they will get executes sequentially. It will not build binary firmware file of code for NodeMCU to write. It will send Lua script as it is to NodeMCU to get execute.

In Arduino IDE when we write and compile code, ESP8266 toolchain in background creates binary firmware file of code we wrote. And when we upload it to NodeMCU then it will flash all NodeMCU firmware with newly generated binary firmware code. In fact, it writes the complete firmware.

That's the reason why NodeMCU not accept further Lua scripts/code after it is getting flashed by Arduino IDE. After getting flashed by Arduino sketch/code it will be no

more Lua interpreter and we got error if we try to upload Lua scripts. To again start with Lua script, we need to flash it with NodeMCU firmware.

Since Arduino IDE compile and upload/writes complete firmware, it takes more time than ESPlorer IDE.

## 6.2 Arduino IDE

If you want to program your Arduino you need to install the Arduino Desktop IDE The Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards. Before you can move on, you must have installed the Arduino Software (IDE) on your PC

Connect your Uno board with an A B USB cable; sometimes this cable is called a *USB printer cable*



Fig:6.2 USB printer cable

The USB connection with the PC is necessary to program the board and not just to power it up. The Uno automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power LED (labelled PWR) should go on.

### *Install the board drivers*

If you used the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board.

### *Open your first sketch*

Open the LED blink example sketch: File > Examples >01.Basics > Blink.

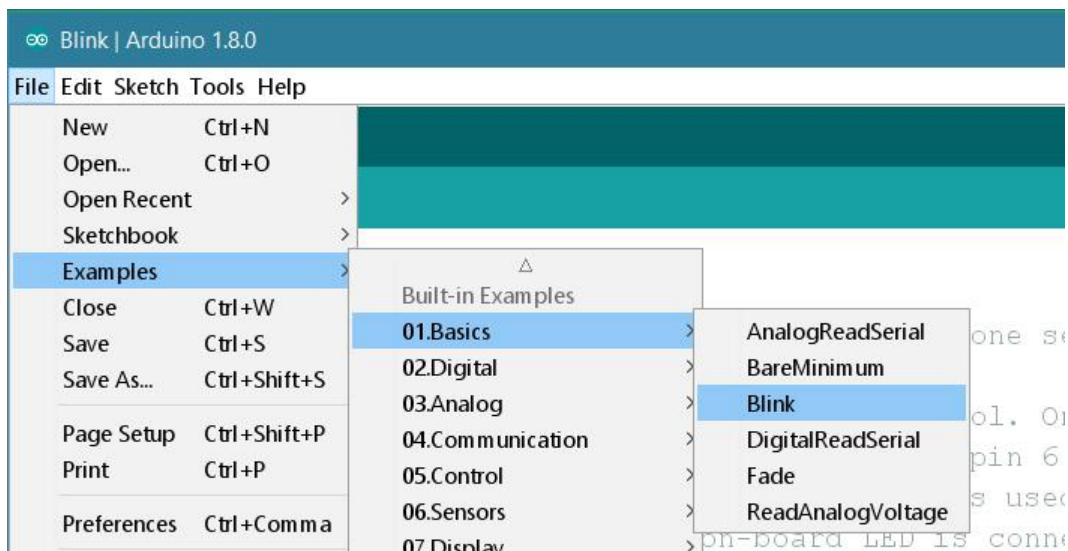


Fig:6.2.1 path to select blink

*Select your board type and port*

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino board.

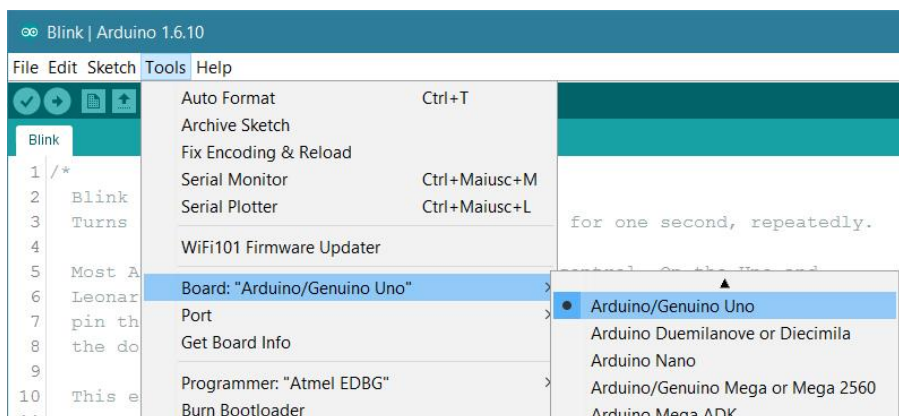


Fig:6.2.2 path to select board type

Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



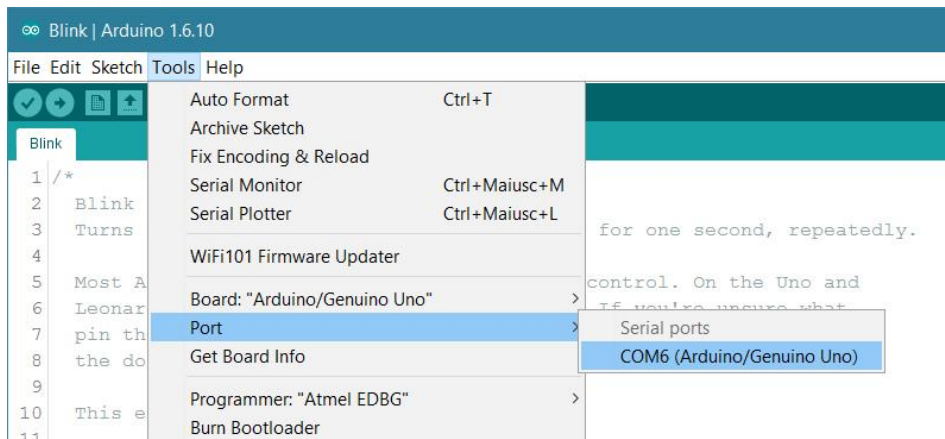


Fig:6.2.3 path to select COM

### *Upload the program*

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange).

## CHAPTER-VII

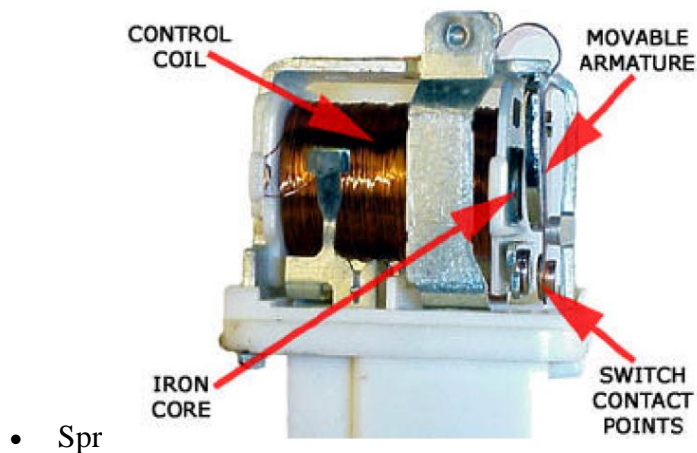
### 7.1 Why is a relay used?

The main operation of a relay comes in places where only a low-power signal can be used to control a circuit. It is also used in places where only one signal can be used to control a lot of circuits. The application of relays started during the invention of telephones. They played an important role in switching calls in telephone exchanges. They were also used in long distance telegraphy. They were used to switch the signal coming from one source to another destination. After the invention of computers they were also used to perform Boolean and other logical operations. The high end applications of relays require high power to be driven by electric motors and so on. Such relays are called contactors.

### 7.2 Relay Design

There are only four main parts in a relay. They are

- Electromagnet
- Movable Armature
- Switch point contacts



• Fig:7.1 Electromagnetic relay

It is an electro-magnetic relay with a wire coil, surrounded by an iron core. A path of very low reluctance for the magnetic flux is provided for the movable armature and also the switch point contacts. The movable armature is connected to the yoke which is mechanically connected to the switch point contacts. These parts are safely held with the help of a spring. The spring is used so as to produce an air gap in the circuit when the relay becomes de-energized.

### 7.3 How relay works?

The relay function can be better understood by explaining the following diagram given below.

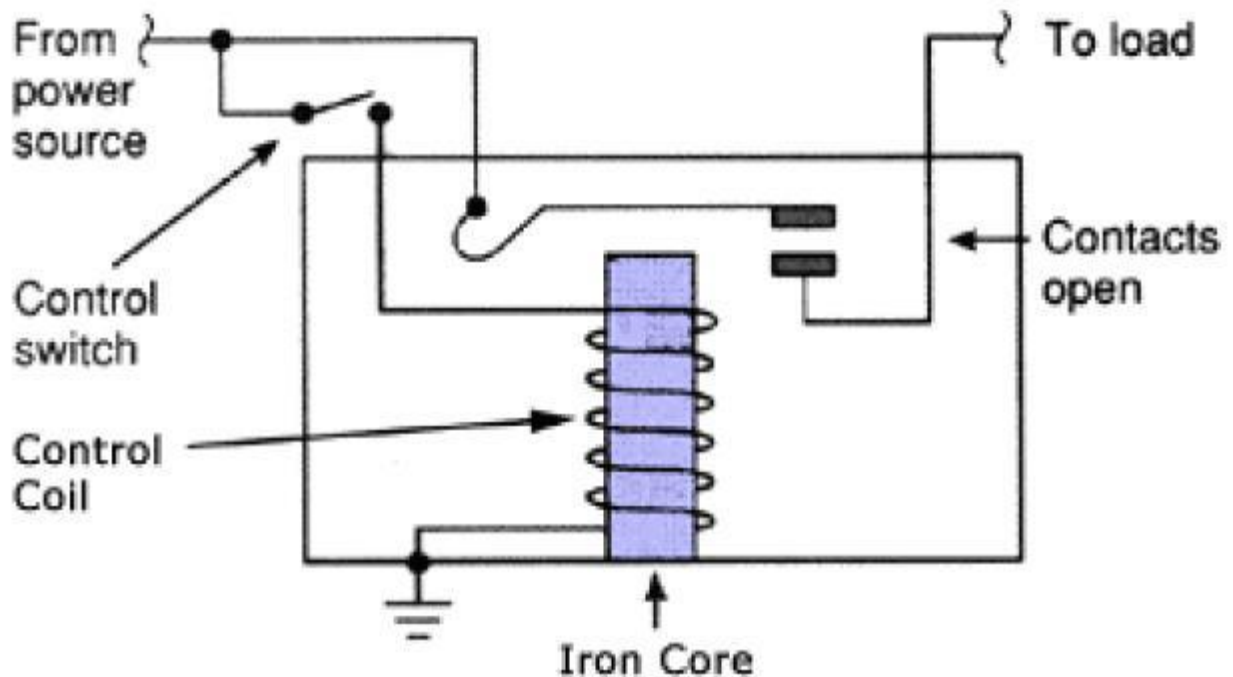


Fig:7.2 Schematic diagram of relay

The diagram shows an inner section diagram of a relay. An iron core is surrounded by a control coil. As shown, the power source is given to the electromagnet through a control switch and through contacts to the load. When current starts flowing through the control coil, the electromagnet starts energizing and thus intensifies the magnetic field. Thus the upper contact arm starts to be attracted to the lower fixed arm and thus closes the contacts causing a short circuit for the power to the load. On the other hand, if the relay was already de-energized when the contacts were closed, then the contact move oppositely and make an open circuit.

As soon as the coil current is off, the movable armature will be returned by a force back to its initial position. This force will be almost equal to half the strength of the magnetic force. This force is mainly provided by two factors. They are the spring and also gravity.

Relays are mainly made for two basic operations. One is low voltage application and the other is high voltage. For low voltage applications, more preference will be given to reduce the noise of the whole circuit. For high voltage applications, they are mainly designed to reduce a phenomenon called arcing.

#### 7.4 Relay Basics

The basics for all the relays are the same. Take a look at a 4 – pin relay shown below. There are two colours shown. The green colour represents the control circuit and the red colour represents the load circuit. A small control coil is connected onto the control circuit. A switch is connected to the load. This switch is controlled by the coil in the control circuit. Now let us take the different steps that occur in a relay.

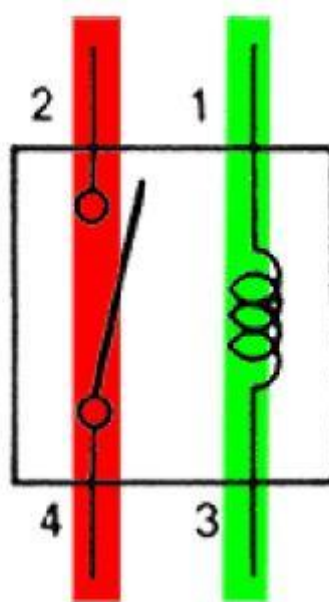


Fig:7.3 4-pin relay

- **Energized Relay (ON)**

As shown in the circuit, the current flowing through the coils represented by pins 1 and 3 causes a magnetic field to be aroused. This magnetic field causes the closing of the pins 2 and 4. Thus the switch plays an important role in the relay working. As it is a part of the load circuit, it is used to control an electrical circuit that is connected to it. Thus, when the electrical relay is energized the current flow will be through the pins 2 and 4.

- **De – Energized Relay (OFF)**

As soon as the current flow stops through pins 1 and 3, the relay switch opens and thus the open circuit prevents the current flow through pins 2 and 4. Thus the relay becomes de-energized and thus in off position.

In simple, when a voltage is applied to pin 1, the electromagnet activates, causing a magnetic field to be developed, which goes on to close the pins 2 and 4 causing a closed circuit. When there is no voltage on pin 1, there will be no electromagnetic force and thus no magnetic field. Thus the switches remain open.

### 7.5 Pole and Throw

Relays have the exact working of a switch. So, the same concept is also applied. A relay is said to switch one or more poles. Each pole has contacts that can be thrown in mainly three ways. They are

- **Normally Open Contact (NO)** – NO contact is also called a make contact. It closes the circuit when the relay is activated. It disconnects the circuit when the relay is inactive.
- **Normally Closed Contact (NC)** – NC contact is also known as break contact. This is opposite to the NO contact. When the relay is activated, the circuit disconnects. When the relay is deactivated, the circuit connects.
- **Change-over (CO) / Double-throw (DT) Contacts** – This type of contacts are used to control two types of circuits. They are used to control a NO contact and also a NC contact with a common terminal. According to their type they are called by the names **break before make** and **make before break** contacts.

Relays can be used to control several circuits by just one signal. A relay switches one or more poles, each of whose contacts can be thrown by energizing the coil.

Relays are also named with designations like

- **Single Pole Single Throw (SPST)** – The SPST relay has a total of four terminals. Out of these two terminals can be connected or disconnected. The other two terminals are needed for the coil to be connected.
- **Single Pole Double Throw (SPDT)** – The SPDT relay has a total of five terminals. Out of these two are the coil terminals. A common terminal is also included which connects to either of two others.
- **Double Pole Single Throw (DPST)** – The DPST relay has a total of six terminals. These terminals are further divided into two pairs. Thus they can act as two SPST's which are actuated by a single coil. Out of the six terminals two of them are coil terminals.
- **Double Pole Double Throw (DPDT)** – The DPDT relay is the biggest of all. It has mainly eight relay terminals. Out of these two rows are designed to be change over

terminals. They are designed to act as two SPDT relays which are actuated by a single coil.

### **7.6 Relay Applications**

- A relay circuit is used to realize logic functions. They play a very important role in providing safety critical logic.
- Relays are used to provide time delay functions. They are used to time the delay open and delay close of contacts.
- Relays are used to control high voltage circuits with the help of low voltage signals. Similarly they are used to control high current circuits with the help of low current signals.
- They are also used as protective relays. By this function all the faults during transmission and reception can be detected and isolated.

### **7.7 Application of Overload Relay**

Overload relay is an electro-mechanical device that is used to safeguard motors from overloads and power failures. Overload relays are installed in motors to safeguard against sudden current spikes that may damage the motor. An overload relay switch works in characteristics with current over time and is different from circuit breakers and fuses, where a sudden trip is made to turn off the motor. The most widely used overload relay is the thermal overload relay where a bimetallic strip is used to turn off the motor. This strip is set to make contact with a contactor by bending itself with rising temperatures due to excess current flow. The contact between the strip and the contactor causes the contactor to de-energize and restricts the power to the motor, and thus turns it off.

Another type of overload motor is the electronic type which continuously watches the motor current, whereas the thermal overload relay shuts off the motor depending on the rise of temperature/heat of the strip.

All overload relays available to buy comes in different specifications, the most important of them being the current ranges and response time. Most of them are designed to automatically reset to work after the motor is turned back on.

### **7.8 Relay Selection**

You must note some factors while selecting a particular relay. They are

- Protection – Different protections like contact protection and coil protection must be noted. Contact protection helps in reducing arcing in circuits using inductors. Coil protection helps in reducing surge voltage produced during switching.

- Look for a standard relay with all regulatory approvals.
- Switching time – Ask for high speed switching relays if you want one.
- Ratings – There are current as well as voltage ratings. The current ratings vary from a few amperes to about 3000 amperes. In case of voltage ratings, they vary from 300 Volt AC to 600 Volt AC. There are also high voltage relays of about 15,000 Volts.
- Type of contact used – Whether it is a NC or NO or closed contact.
- Select Make before Break or Break before Make contacts wisely.
- Isolation between coil circuit and contacts

## CHAPTER-VIII

### 8.1 What is IoT?

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:

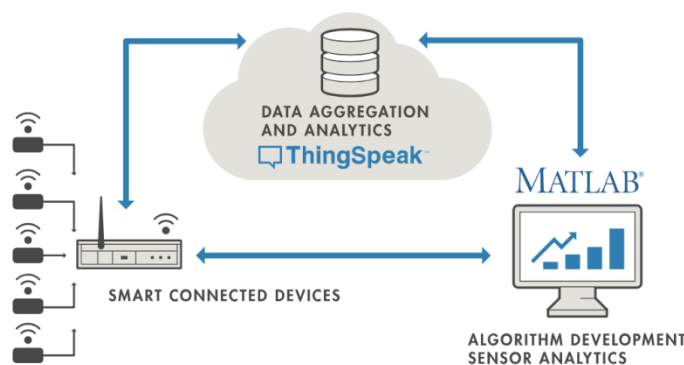


Fig:8.1 Block diagram of IOT

On the left, we have the smart devices (the “things” in IoT) that live at the edge of the network. These devices collect data and include things like wearable devices, wireless temperatures sensors, heart rate monitors, and hydraulic pressure sensors, and machines on the factory floor.

In the middle, we have the cloud where data from many sources is aggregated and analyzed in real time, often by an IoT analytics platform designed for this purpose.

The right side of the diagram depicts the algorithm development associated with the IoT application. Here an engineer or data scientist tries to gain insight into the collected data by performing historical analysis on the data. In this case, the data is pulled from the IoT platform into a desktop software environment to enable the engineer or scientist to prototype algorithms that may eventually execute in the cloud or on the smart device itself.



An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.

### ThingSpeak Key Features

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.
- Automatically act on your data and communicate using third-party.

## 8.2 Create a Channel

1. Sign In to ThingSpeak™ using your MathWorks® Account, or create a new MathWorks account.
2. Click Channels > MyChannels.
3. On the Channels page, click New Channel.
4. Check the boxes next to Fields 1–3. Enter these channel setting values:
  - Name: Dew Point Measurement
  - Field 1: Temperature (F)
  - Field 2: Humidity
  - Field 3: Dew Point
5. Click Save Channel at the bottom of the settings.

You now see these tabs:

- Private View: This tab displays information about your channel that only you can see.
- Public View: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.
- Channel Settings: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.
- Sharing: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.

- **API Keys:** This tab displays your channel API keys. Use the keys to read from and write to your channel.

**Data Import/Export:** This tab enables you to import and export channel data.

## **CONCLUSIONS**

Life today is getting easier and simpler with advancement of automation technology. Manual systems are getting replaced by automatic systems. With the rapid increase in Internet users it has become part of life. One of its kinds is IoT, latest and emerging technology. Things like consumer goods, industrial goods, etc., can be networked to share information and complete the task remotely. Basic home functions and features can be controlled using IoT from anywhere in the world. It is meant to save human and electrical energy. In this system each load is monitored by current and potential transformer.