

# **IOT BASED SMART HOME USING BLYNK FRAME WORK**

**A Project Report Submitted to**

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA KAKINADA**

**In partial fulfillment of the award of degree of**

**Bachelor of Technology**

**In**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**Submitted by**

<b>K.NAGANNA</b>	<b>(18X91A0427)</b>
<b>A.SANDHYA RANI</b>	<b>(18X91A0401)</b>
<b>C.VENKATESH</b>	<b>(18X91A0409)</b>
<b>A.SAURAV</b>	<b>(17X91A0403)</b>

**Under the esteemed guidance of**

**Mr.G.VIJAYA KIRAN** **M.TECH(Ph.D),**

**Associate Professor**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**SRI SATYANARAYANA ENGINEERING COLLEGE**

**(An ISO 9001:2008 Certified Institution)**

**(Approved by AICTE, Affiliated to J.N.T.U.K KAKINADA)**

**Ongole-523001, A.P, INDIA.**

**YEAR-2021-2022**

# **SRI SATYANARAYANA ENGINEERING COLLEGE::ONGOLE**

**(AN ISO 9001:2008 Certified Institution)**

**(Approved by AICTE, Affiliated to J.N.T.U.K, KAKINADA)**

**Ongole-523001, A.P, INDIA.**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

## **CERTIFICATE**

This is to certify that the Project report titled “ **IOT BASED SMART HOME USING BLYNK FRAME WORK**” is the bonafide work carried out by

**K.NAGANNA (18X91A0427)**

**A.SANDHYA RANI (18X91A0401)**

**C.VENKATESH (18X91A0409)**

**A.SAURAV (17X91A0403)**

in partial fulfillment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering by J.N.T.U.K, Kakinada during the academic year 2021-2022.

**Project Guide**

**MR.G.VIJAYA KIRAN,M.TECH(Ph.D)**

**ASSOCIATE PROFESSOR**

**HOD**

**MR.G.VIJAYA KIRAN,M.TECH(Ph.D)**

**ASSOCIATE PROFESSOR**

**Project Co-ordinator**

**MR.SWARNA RAJESH,M.TECH**

**ASSISTANT PROFESSOR**

**EXTERNAL EXAMINER**

# ACKNOWLEDGMENT

I would like to express our profound sense of gratitude and indebtedness to our project guide **Mr.G.VijAY KIRAN<sup>MTECH PHD</sup>**. Associate professor of ECE Department, SSN Engineering College, Ongole for the valuable guidance and inspiration rendered by him during this project work. Mere words would not be sufficient to place on record the erudite guidance, sustained encouragement, constructive comments and inspiring discussions with him during course of this project work.

I honorably express our thanks to our project coordinator **Mr. S.RAJESH<sup>M.Tech</sup>**, Assistant professor for providing us an utmost congenial atmosphere to carry out our project work peacefully.

I are highly grateful and indebted to **Mr. G.VIJAYA KIRAN<sup>M.Tech, [Ph.D]</sup>**, HOD of the ECE Department for his support and valuable suggestions in completing this project work.

I would like to express our deep sense of gratitude to all **staff members of Electronics and Communication Engineering Department**, SSN Engineering College, Ongole, for their support in various aspects in completing this project work.

I would like to place on record the valuable academic support given by our principal **T.RAMA PITCHAIAH<sup>,M.Tech,[Ph.D]</sup>** for completion of the course.

I express our sincere gratitude and thanks to the honorable director **Mr. Y.SATYA NARAYANA REDDY** for encouraging us in all aspects and for helping us for all round development and guiding us to take right direction to become professionals with moral values.

At the outset we extend our special thanks to our honourable Secretary and Correspondent **Mr. Y.RAMA KRISHNA REDDY** for providing us good faculty and infrastructure for his moral support throughout the course.

I would like to convey gratitude to our parents whose prayers and blessings are always with us.

In this regard we would like to thank our friends and others who helped us directly or indirectly for successful completion of the project.

<b>K.NAGANNA</b>	<b>(18X91A0427)</b>
<b>A.SANDHYA RANI</b>	<b>(18X91A0401)</b>
<b>C.VENKATESH</b>	<b>(18X91A0409)</b>
<b>A.SAURAV</b>	<b>(17X91A0403)</b>

# **ABSTRACT**

The project discussed about solving some problems faced by present people in their daily life. It is designed to control and monitor appliances via smartphone using Wi-Fi as communication protocol and raspberry pi as private server. All the appliances and sensors are connected to the internet via NodeMcu microcontroller, which serves as the gateway to the internet. Even if the user goes offline, the system is designed to switch to automated state controlling the appliances automatically as per the sensors readings. Also, the data are logged on to the server for future data mining. The core system of this project is adopted from the Blynk framework\*\*.

# INDEX

CONTENTS	PAGE NO
ACKNOWLEDGEMENT	
ABSTRACT	
LIST OF FIGURES	
CHAPTER-1: INTRODUCTION	
1.1 INTRODUCTION	1
1.2 AIM OF THE PROJECT	2
1.3 METHODOLOGY	2
1.4 SIGNIFICANT OF THE WORK	2
1.5 INTRODUCTION TO EMBEDDED SYSTEMS	2
1.6 EMBEDDED SYSTEM DESIGN CYCLE	3
1.6.1 CHARACTERSITICS OF EMBEDDED SYSTEM	3
1.6.2 APPLICATIONS	4
1.7 OVERVIEW OF EMBEDDED SYSTEM ARCCITECTURE	4
1.8 APPLICATION -SPECIFIC CIRCUITRY	5
1.8.1 CENTRAL PROCESSING UNIT	5
1.8.2 MEMORY	5
1.8.3 INPUT DEVICES	6
1.8.4 OUPUT DEVICES	6
1.8.5 COMMUNICATION INTERFACES	6
1.8.6 APPLICATION SPECIFIC CIRCUITARY	6
1.9 MICRO CONTROLLER	7
CHAPTER-2: DESCRIPTION OF BLOCK DIAGRAM	
2.1 BLOCK DIAGRAM	9
CHAPTER-3: NODEMCU	
3.1 NODEMCU	10
3.2 HISTORY	10
3.3 SPECIFICATIONS	10
3.4 NODEMCU PIN OUT	11

## **CHAPTER -4: DHT11 SESNOR**

4.1 INTRODUCTION	12
4.2 HOW THE DHT11 MEASURES HUMIDITY AND TEMPERATURE	12

## **CHAPTER-5 PASSIVE INFRARED SENSOR**

5.1 PIR SENSOR BRIEF INTRODUCTION	13
5.2 ELECTRICITY	13
5.3 SENSOR	13
5.4 WORKING PRINCIPLE	14
5.5 APPLICATIONS	14

## **CHAPTER-6: RELAY**

6.1 CHOOSING A RELAY	16
6.1.1 PHYSICAL SIZE AND PIN ARRANGEMENT	16
6.1.2 COIL VOLTAGE	16
6.1.3 COIL RESISTANCE	16
6.1.4 SWITCHING RATINGS (VOLTAGE AND CURRENT)	16
6.1.5 SWITCH CONTACTS ARRANGEMENT (SPDT, DPDT ETC)	16
6.2 ADVANTAGES OF RELAYS	16
6.3 DISADVANTAGES OF RELAYS	16

## **CHAPTER-7 ATMEGA328 MICRO CONTROLLER**

7.1 MICRO CONTROLLER	17
7.2 INTRODUCTION TO MICRO CONTROLLERS	17
7.3 ARCHITECTURE	18
7.4 EEPROM	19
7.5 MCU SPEED	19
7.6 FEATURES	19
7.7 POWER-ON RESET	21
7.8 EXTERNAL RESET	22
7.9 APPLICATIONS	25

## **CHAPTER-8 ARDUINO UNO**

8.1 INTRODUCTION TO ARDUINO UNO	26
8.2 CONCEPT OF ARDUINO	27

8.3 HISTORY	28
8.4 OPENSOURCE LICENSE	28
8.5 ATMEL ATMEGA328P	28
8.6 ARDUINO UNO	30
8.7 SUMMARY	31
8.8 PIN CONFIGURATION	32
8.8.1 MEMORY	33
8.8.2 INPUT AND OUTPUT	33
8.8.3 COMMUNICATION	34
8.9 ARDUINO IDE	36
<b>CHAPTER -9 BLYNK</b>	
9.1 BLYNK PLAT FORM	50
9.2MCU LIBRARY	51
<b>CHAPTER -10 RESULT AND DISCUSSION</b>	
<b>WORKING</b>	<b>54</b>
<b>CONCLUSION &amp; FUTURESCOPE</b>	
CONCLUSION	55
FUTURE SCOPE	55
<b>BIBILOGRAPHY</b>	56
<b>SOFTWARE CODING</b>	<b>57</b>

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.6	V DIAGRAM	3
1.8	APPLICATION SPECIFIC CIRCUITARY	5
1.9	MICRO CONTROLLER	8
2.1	WORKING PLAN OF THE PROPOSED SYSTEM	9
3.3	SPECIFICATION OF NODE MCU	11
3.4	NODE MCU PIN OUT	11
4.2	DHT11 SENSOR	12
6.1	DIAGRAM OF RELAY	15
7.1	MICRO CONTROLLER`	17
7.3	ARCHITECTURE	18
7.8	ATMEGA 328	22
7.8.1	PIN DIAGRAM OF ATMEGA328	23
8.6	ARDUINO	30
8.8	PIN CONFIGURATION	35
8.9.1	OPEN ARDUINO IDE	39
8.9.2	SELECT THE COM PORT FROM TOOLS	41
8.9.3	SELECT THE REQUIRED ARDUINO FROM TOOLS	42
8.9.4	WRITE THE SKETCH IN ARDUINO IDE	43
8.9.5	COMPILE AND UPLOAD THE SKETCH TO ARDUINO BOARD	44
9.1	BLYNK	45
10	PROJECT OUTPUT	53



# CHAPTER1

## INTRODUCTION

### 1.1 INTRODUCTION

The first quarter of the 21st century has pushed the 4th industrial revolution with the increasingly widespread internet of things (IoT). One of the most popular applications on IoT is smart home. Smart home as a home which is smart enough to assist the inhabitants to live independently and comfortably with the help of technology is termed as smart home.

In a smart home, all the mechanical and digital devices are interconnected to form a network, which can communicate with each other and with the user to create an interactive space.

With the rapid development of technology and price reductions of devices and sensors make smart home more easily and inexpensively obtained and implemented in a real home. In this paper, we discuss designing a smart home in an easy way on a building that has been established and does not have a smart home installation plan, and has minimal costs through a smartphone.

#### ***A . Problem Statement and Significance***

In villages, load shedding has been an incorrigible and impasse problem for more than a decade now. During dry seasons, electricity cut-off reaches up to 16 hours a day [3]. Hence, people are obliged to make troublesome decisions to comply with the power-cut schedule. In cities, most of the people are job indulged and are busy at the day time, which, unfortunately, is low peak-load hour when electricity is available at home. On the other hand, at the time when people reach their home, everybody turn ON their appliances that makes peak load hour, where load-shedding is scheduled to maximum extent. So, there is a need of a system that can automate household tasks such as filling water tanks, charging devices and such, in the absence of the house owners.

Many incidents of robbery are witnessed in cities such as Houses, shops, offices, industries, business complexes, etc. are no longer safe. Illminded people are always in search of right opportunity to rob these places. Thus, information regarding intruder detection and alarms are needed for security. Above all, security is critical and also is the foremost priority of this research project.

## **1.2 AIM OF THE PROJECT**

The main aim of the function is to improve the quality of life and convenience in the home. Other goals are greater security and efficient use of energy tanks to connected remote controllable devices. Home appliances, such as washing machines, lights or the coffee maker can be controlled.

## **1.3 METHODOLOGY**

In designing the smart home, in this study used sensors, actuators, NodeMCU, Raspberry Pi, Smartphones, and Blynk framework are Sensors and actuators are connected to NodeMCU. Each room is installed with one NodeMCU as a client or more depending on I / O requirements designed. In a house the entire NodeMCU is connected wirelessly to the Raspberry pi as a server and is used as a data center and control and bridge between NodeMCU and the internet. With wireless installations, homeowners do not need additional costs for renovation and installation of cables. After connecting to the internet, the homeowner can control and monitor the condition of his home through a smartphone equipped with the Blynk applications.

## **1.4 SIGNIFICANT OF THE WORK**

This automated control systems proved to be versatile tools for many problems in human-and mobile interface systems. Basically, they are used for providing better usability of a computer or a system for people, including disabled people.

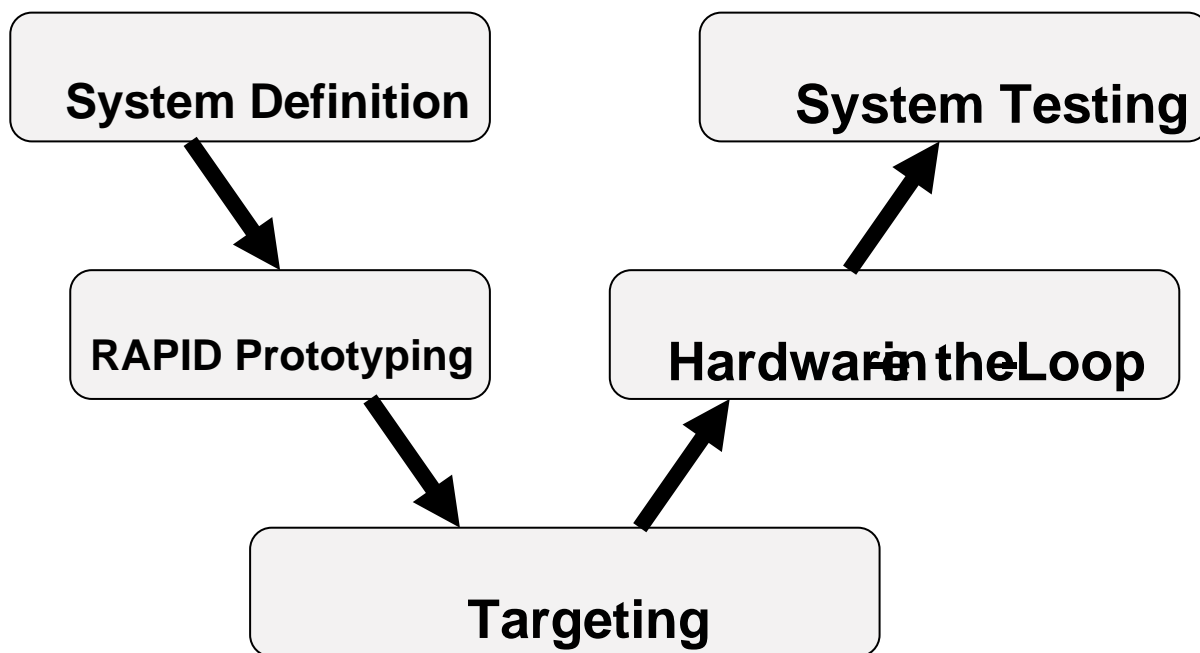
## **1.5 INTRODUCTION TO EMBEDDED SYSTEMS**

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, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc. Lower end embedded systems - Generally

8,16 Bit Controllers used with an minimal operating systems and hardware layout designed for the specific purpose.

## 1.6 EMBEDDED SYSTEM DESIGN CYCLE



**Figure 1.6 “V Diagram”**

### 1.6.1 Characteristics of Embedded System

- An embedded system is any computer system hidden inside a product other than a computer.
- They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications
  - Throughput – Our system may need to handle a lot of data in a short period of time.
  - Response – Our system may need to react to events quickly
  - Testability – Setting up equipment to test embedded software can be difficult
  - Debugability – Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem
  - Reliability – embedded systems must be able to handle any situation without human intervention
  - Memory space – Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists

- Program installation – you will need special tools to get your software into embedded systems
  - Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power
  - Processor hogs – computing that requires large amounts of CPU time can complicate the response problem
  - Cost – Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.
- Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.

### **1.6.2 APPLICATIONS**

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

### **CLASSIFICATION**

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

### **RTS CLASSIFICATION**

- Hard Real Time Systems
- Soft Real Time System

### **HARD REAL TIME SYSTEM**

- "Hard" real-time systems have very narrow response time.

- Example: Nuclear power system, Cardiac pacemaker.

## **SOFT REAL TIME SYSTEM**

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

## **1.7 OVERVIEW OF EMBEDDED SYSTEM ARCHITECTURE**

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the 'firmware'. The embedded system architecture can be represented as a layered architecture as shown in Fig.

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time you don't need to reload new software.

Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are;

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random Access Memory)
- Input Devices
- Output devices
- Communication interfaces

## 1.8 APPLICATION-SPECIFIC CIRCUITRY

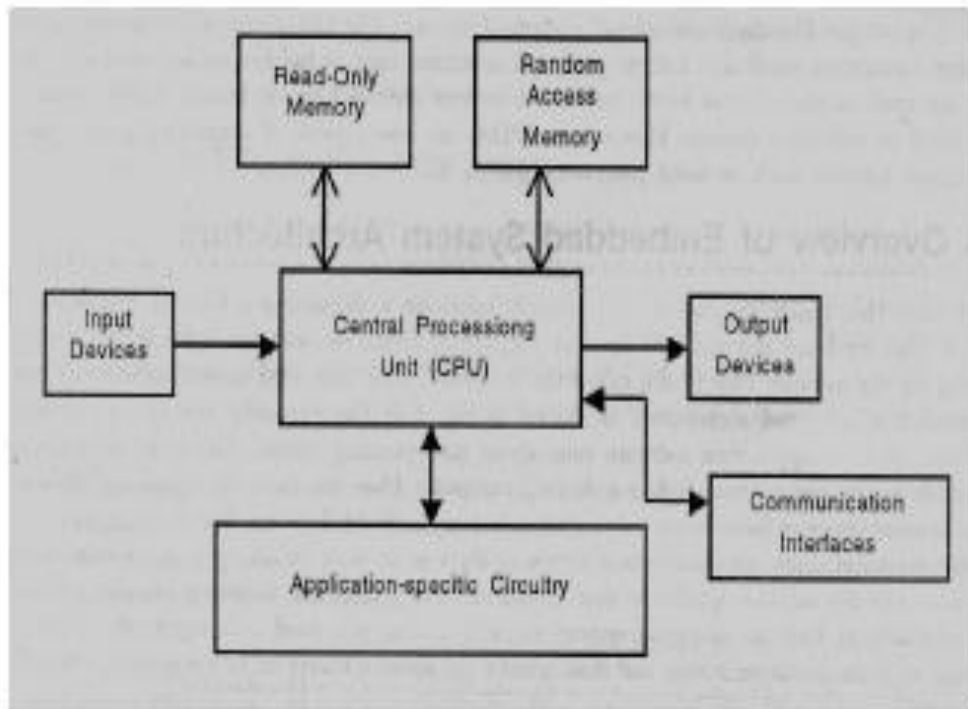


Fig 1.8:APPLICATION SPECIFIC CIRCUITRY

### 1.8.1 CENTRAL PROCESSING UNIT (CPU):

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. DSP is used mainly for applications in which signal processing is involved such as audio and video processing.

### 1.8.2 MEMORY:

The memory is categorized as Random Access Memory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is executed.

### 1.8.3 INPUT DEVICES:

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors or transducers and produce electrical signals that are in turn fed to other systems.

#### **1.8.4 OUTPUT DEVICES:**

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

#### **1.8.5 COMMUNICATION INTERFACES:**

The embedded systems may need to, interact with other embedded systems as they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc.

#### **1.8.6 APPLICATION-SPECIFIC CIRCUITRY:**

Sensors, transducers, special processing and control circuitry may be required for an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

#### **APPLICATION AREAS**

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

##### **Consumer appliances:**

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today's high-tech car has about 20 embedded systems for

transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

### **Office automation:**

The office automation products using em embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

### **Industrial automation:**

Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

### **Medical electronics:**

Almost every medical equipment in the hospital is an embedded system. These equipments include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; equipment used in blood analysis, radiation, colonoscopy, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

### **Computer networking:**

Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks. The two networks may be running different protocol stacks. The router's function is to obtain the data packets from incoming pores, analyze the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipments, other than the end systems (desktop computers) we use to access the networks, are embedded systems



**. Telecommunications:**

In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Disassemblers (PADs), satellite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

**Wireless technologies:**

Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the 20<sup>th</sup> century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia services over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

**Insemination:**

Testing and measurement are the fundamental requirements in all scientific and engineering activities. The measuring equipment we use in laboratories to measure parameters such as weight, temperature, pressure, humidity, voltage, current etc. are all embedded systems. Test equipment such as oscilloscope, spectrum analyzer, logic analyzer, protocol analyzer, radio communication test set etc. are embedded systems built around powerful processors. Thank to miniaturization, the test and measuring equipment are now becoming portable facilitating easy testing and measurement in the field by field personnel.

**Security:**

Security of persons and information has always been a major issue. We need to protect our homes and offices; and also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in~ embedded systems.

## Finance:

Financial dealing through cash and cheques are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; and it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society. Well, the list goes on. It is no exaggeration to say that eyes wherever you go, you can see, or at least feel, the work of an embedded system.

Embedded systems find applications in . every industrial segment- consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defense, security etc. Used to encrypt the data/voice being transmitted on communication links such as telephone lines.

## 1.9 MICROCONTROLLERS

A microcontroller (sometimes abbreviated  $\mu C$ ,  $\mu C$  or MCU) is a small computer on a single [integrated circuit](#) containing a processor core, memory, and programmable [input/output](#) peripherals. Program memory in the form of [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.

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 (milli watts or micro watts). 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.

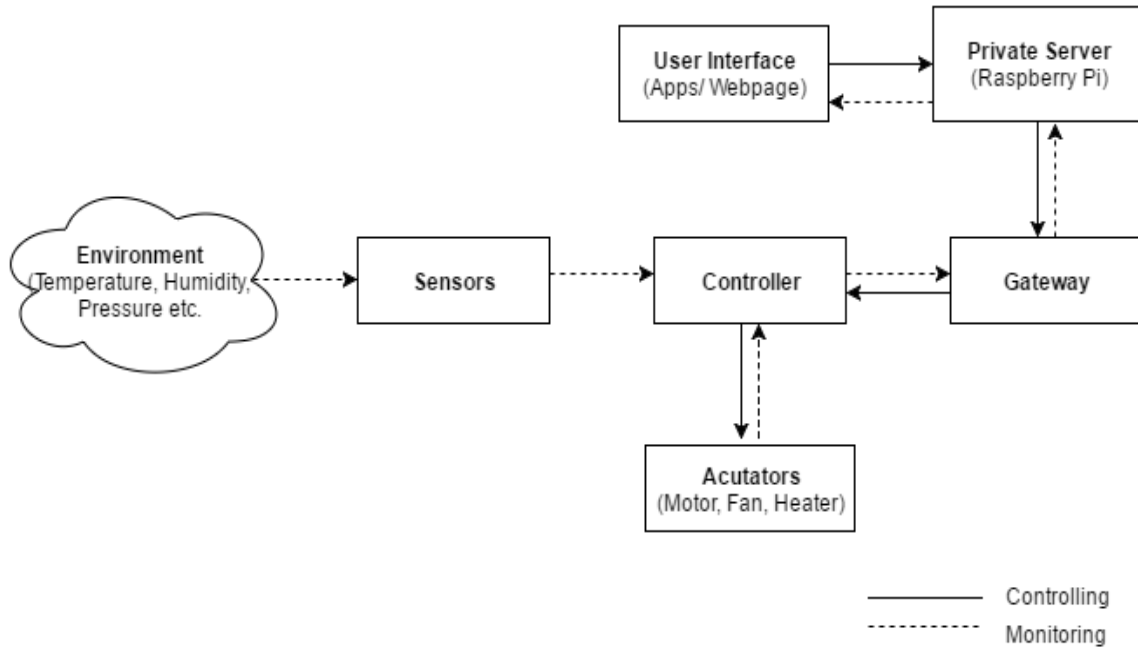
Many embedded systems need to read sensors that produce analog signals. This is the purpose of the [analog-to-digital converter](#) (ADC). Since processors are built to interpret and process digital data, i.e.

1s and 0s, they are not able to do anything with the analog signals that may be sent to it by a device. So the analog to digital converter is used to convert the incoming data into a form that the processor can recognize. A less common feature on some microcontrollers is a [digital-to-analog converter](#) (DAC) that allows the processor to output analog signals or voltage levels.

## CHAPTER -2

### DESCRIPTION OF BLOCK DIAGRAM

#### 2.1 BLOCK DIAGRAM



**Fig2.1 Working Plan of the Proposed System**

#### 2.2 REQUIRED MODULES

- Node MCU
- DHT11 Sensor
- PIR Sensor
- Relay
- ATmega328
- Arduino UNO
- DC Fan

## CHAPTER 3

### 3.NODEMCU

#### 3.1 NODEMCU:

**NodeMCU** is an open source IOT platform. It includes firmware which runs on the ESP8266 WiFi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.

#### 3.2 HISTORY:

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Express if Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IOT applications. NodeMCU started on 13 Oct 2014, when Hong committed the first file of NodeMCU-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IOT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent but dedicated contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

#### 3.3 SPECIFICATIONS:

NodeMCU Dev Board is based on widely explored esp8266 System on Chip from Express if. It combined features of WIFI access point and station + microcontroller and uses simple LUA based programming language. ESP8266NodeMCU offers-

- Arduino-like hardware IO
- Event-driven API for network applications
- 10 GPIOs D0-D10, PWM functionality, IIC and SPI communication, 1-Wire and ADC A0 etc. all in one board
- Wi-Fi networking (can be uses as access point and/or station, host a webserver), connect to internet to fetch or upload data.
- excellent few \$ system on board for Internet of Things (IOT) projects.

Recently, there has been interest in programming ESP8266 systems using Arduino IDE.

Programming, of ESP8266 using Arduino IDE is not very straight forward, until it is properly configured. Especially because, the Input and output pins have different mapping on NodeMCU than those on actual ESP8266 chip.

I had request about showing how to program ESP-12E NodeMCU using Arduino IDE. I struggled myself earlier in the beginning, so thought of making this Instruct able for beginners. This is quick guide/tutorial for getting started with Arduino and ESP8266 NodeMCU V2 ESP-12Ewifi module. (I think, this method can be used for other NodeMCU boards too. (or only ESP8266 boards, but with necessary hardware modifications and using FTDI modules for programming- not covered in this tutorial because, this is only for NodeMCU development boards).

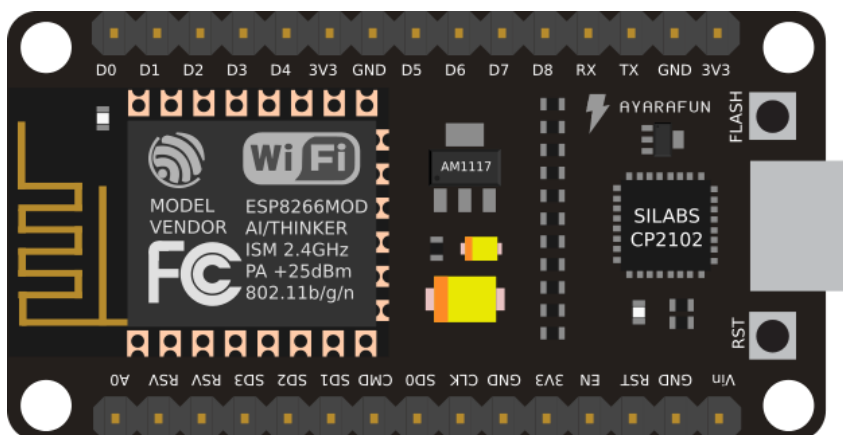
This Instruct able gives quick intro to-

- 1) Installing Arduino core for ESP8266 Wi-Fi chip in Arduino IDE and Getting started with sketches written using Latest stable Arduino IDE 1.6.7
- 2) Run/modify basic LED blink sketch to blink onboard LED and/or externally connected LED at pin D0 or GPIO-16 as per the pin configuration mentioned here and [here](#).

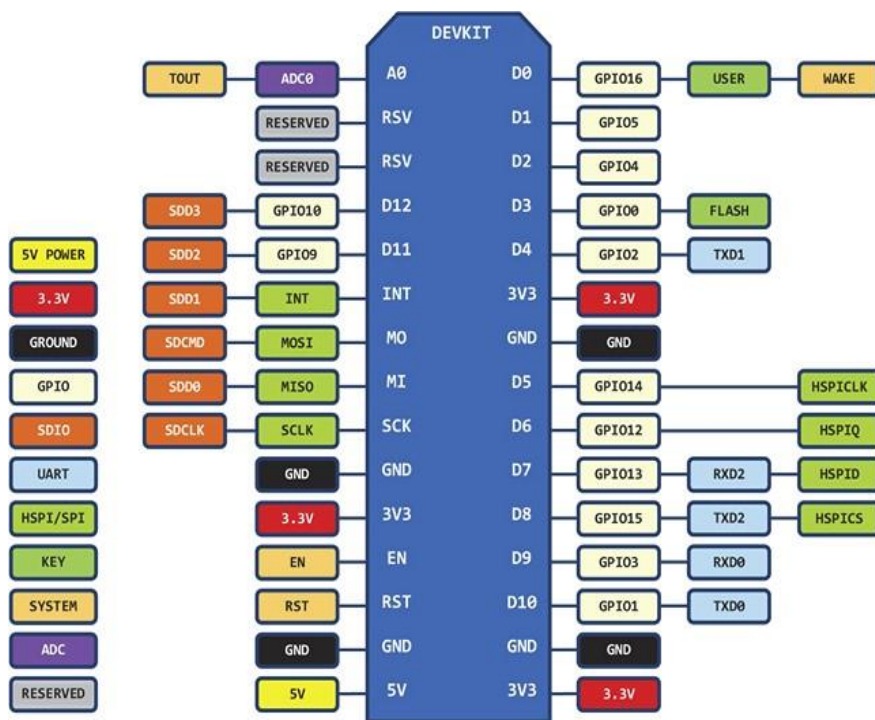
**NOTE-** To use NodeMCU V1 or V2 or V3 development boards using Arduino IDE, we do not need to flash it with firmware using NodeMCU flasher. It is required only if we intend to program NodeMCU using Lua script with explorer etc.

First and foremost word of - CAUTION!

- \* The ESP8266 chip requires 3.3V power supply voltage. It should not be powered with 5 volts like other Arduino boards.
- \* NodeMCU ESP-12E development board can be connected to 5V using **micro USB** connector or **Vin** pin available on board.
- \* The I/O pins of ESP8266 communicate or input/output max 3.3V only. The pins are **NOT** 5V tolerant inputs.



### 3.4 NODEMCU PIN OUT:



*D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.*

Fig:3.4 NODE MCu

## CHAPTER 4

### DHT11 SENSOR

#### 4.1 INTRODUCTION:

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmers in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

#### 4.2 HOW THE DHT11 MEASURES HUMIDITY AND TEMPERATURE

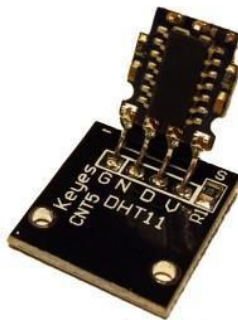
The DHT11 calculates relative humidity by measuring the electrical resistance between two electrodes. The humidity sensing component of the DHT11 is a moisture holding substrate (usually a salt or conductive plastic polymer) with the electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes while lower relative humidity increases the resistance between the electrodes. Inside the DHT11 you can see electrodes applied to a substrate on the front of the chip:



Front view with  
cover removed



The DHT11 converts the resistance measurement to relative humidity on an IC mounted to the back of the unit and transmits the humidity and temperature readings directly to the Arduino. This IC also stores the calibration coefficients and controls the data signal transmission between the DHT11 and the Arduino:

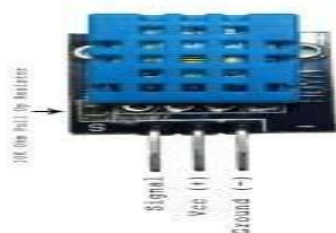


Rear view with cover removed

The temperature readings from the DHT11 come from a surface mounted NTC temperature sensor (thermistor) built into the unit. To learn more about thermistors and how to use them on the Arduino, check out our [Arduino Thermistor Temperature Sensor Tutorial](#).

The DHT11 uses one signal wire to transmit sensor readings to the Arduino digitally. The power comes from separate 5V and ground wires. A 5K – 10K Ohm pull-up resistor is connected from the signal line to 5V to make sure the signal level stays high by default (see the datasheet for specifics on how the signal is sent).

There are two different variations of the DHT11 sensor you might come across. One type has four pins, and the other type is mounted to a small PCB that has three pins. The PCB mounted version with three pins is nice since it includes a surface mounted 10K Ohm pull up resistor for the signal line:



## DHT11 SENSORS

## 5.PASSIVE INFRARED SENSOR

### 5.1 PIR SENSOR BRIEF INTRODUCTION:

This PIR Sensor Switch Can Detect the Infrared Rays released by Human Body Motion within the Detection Area (14 Meters), and Start the Load - Light Automatically. This Unit is Suitable for Outdoor Use (Corridor, Staircase, Courtyard etc.)



### 5.2 ELECTRICITY:

It has been estimated that a single unit of energy saved at the end use point is equal to 2.3 units of energy produced.

If energy efficient methods are implemented properly about 25000mw equivalent capacity of power can be created through promotion of energy efficient measures.

### 5.3 PIR SENSOR:

O A PIR Sensor is a Passive Infrared Sensor which controls the switching on/off of the lighting load when it detects a moving target.

O The built in sensor turns on/off the connected lighting load when it detects motion in the coverage area. It has different working principle during the day time and the night time.

O During the day, the built in photocell sensor saves electricity by deactivating the lighting load connected to the sensor.

O During the night the connected lighting load is turned on by adjusting the luminosity knob (LUX).

An adjustable time knob lets you select how long the light stays on after

#### **5.4 WORKING PRINCIPLE :-**

- O The PIR Sensor senses the motion of a human body by the change in surrounding ambient temperature when a human body passes across.
- O Then it turns on the lighting load to which it is connected.
- O The lighting load remain on until it senses motion.
- O Once the motion is seized it switches off the lighting load.
- O During the night, the LUX adjustment knob allows you to adjust the luminosity based on which the lighting load will either switch on/off automatically.

#### **5.5 APPLICATIONS:**

- O Common toilets, for lights & exhaust fans
- O Common staircases
- O For parking lights
- O For garden lights
- O For changing rooms in shops
- O For corridors & many more

#### **5.6 COST EFFECTIVENESS :**

- O In just Rs. 750 you save the most precious form of energy, electricity.

## CHAPTER 6

### RELAY

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 they are double throw switches.

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 electrically connection inside the relay between the two circuits; the link is magnetic and mechanical.

The coil of a relay passes 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 chips cannot provide this current and transistor. Usually used to amplify the small IC current to the larger value required for the relay coil. The maximum output current for the popular 555timer IC is 200ma, so these devices can supply relay coils directly without amplifications.

Relays usually SPDT or DPDT but they can have more sets of switch contacts, for example relays with four sets of change over contacts are readily available.

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 relay connection. 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 be destroying transistors and IC's in the circuit. To prevent damage, we must connect a protection diode across a relay coil. The figure shows a working relay with its coil and switch contacts. You can see the lever on the being attracted by magnetism when the coil is switched on. This lever moves the switches contacts.



**Fig6.1 Diagram of relay**

The relay's switch connections are usually labeled 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 relay is off.
- **NO**=normally open, COM is connected to this when relay is no.
- Connect to COM and NO if you want the switched circuit to be on when the relay coil is on.
- Connect to COM and NC if you want the switched circuit to be on when the relay coil is off.

### 6.1 CHOOSING A RELAY:

You need to consider several features in choosing a relay.

#### 1. PHYSICAL SIZE AND PIN ARRANGEMENT:

If you are choosing a relay for an existing PCB you will need to ensure that its dimension and pin arrangement are suitable.

#### 2. COIL VOLTAGE:

The relay's coil voltage rating and resistance must suit the circuit powering the relay coil. Many relay coils. Many relays have a coil rated for a 12v supply but 5v and 12v relays are readily available. Some relays operate perfectly well with a supply voltage which is little lower than rated value.

#### 3. COIL RESISTANCE:

The circuit must be able to supply the current required by the relay coil. We can use ohms law to calculate:

Relay coil current= Supply voltage/ Coil resistance

For example: A 12v supply relay with a coil resistance of 400ohms passes a current of 30ma. This is ok for 555timer IC (maximum output current 200ma), but it is too much for most ICs and they will require a transistor to amplify the current.

#### 4. SWITCH RATINGS (VOLTAGE AND CURRENT):

The relay's switch contacts must be suitable for the circuit they are to control. We will need to check the voltage and current ratings. Note that the voltage rating is usually higher for AC, for example: "5A AT 24VDC OR 125V AC".

## **5. SWITCH CONTACTS ARRANGEMENT (SPDT, DPDT ETC):**

Most relays are SPDT or DPDT which are often described as 'single pole changeover' (SPCO) or 'double pole changeover' (DPCO).

### **6.2 ADVANTAGES OF RELAYS:**

- Relays can switch AC and CD, transistors can only switch CD.
- Relays can switch high voltages, transistors cannot.
- Relay are a better choice for switching large currents ( $>5a$ ).
- Relay can switch many contacts at once.

### **6.4 DISADVANTAGES OF RELAYS:**

- Relays are bulkier than transistors for switching small currents.
- Relays cannot switch rapidly (except reed relays), transistors can switch many times per second. Relays use more power due to the current following through their coil. Relays require more current than many chips can provide, so a low power transistor may be needed to switch the current for the relays coil.

## CHAPTER 7

### ATMEGA328 MICRO CONTROLLER

#### 7.1 Micro controller:



**Fig:7.1 Microcontrollers**

#### 7.1.2 Introduction to Microcontrollers:

Circumstances that we find ourselves in today in the field of microcontrollers had their beginnings in the development of technology of integrated circuits. This development has made it possible to store hundreds of thousands of transistors into one chip. That was a prerequisite for production of microprocessors, and the first computers were made by adding external peripherals such as memory, input/output lines, timers and other. Further increasing of the volume of the package resulted in creation of integrated circuits. These integrated circuits contained both processor and peripherals. That is how the first chip containing a microcomputer, or what would later be known as a microcontroller came about.

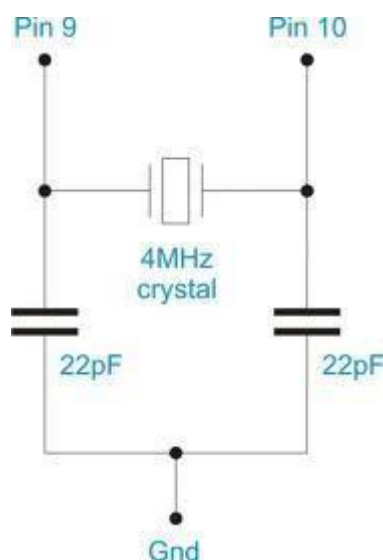
Microprocessors and microcontrollers are widely used in embedded systems products. Microcontroller is a programmable device. A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports and a timer embedded all on a single chip. The fixed amount of on-chip ROM, RAM and number of I/O ports in microcontrollers makes them ideal for many applications in which cost and space are critical.

The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to One-Time Programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

### 7.1.3 CRYSTAL OSCILLATOR:

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, Either a quartz

Crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably.



This mode has a limited frequency range and it cannot be used to drive other clock buffers. For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. For ceramic resonators, the capacitor values given by the manufacturer should be used. The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1

### 7.1.3 ARCHITECTURE:

**Memory:** It has **8 Kb** of Flash program memory (10,000 Write/Erase cycles durability), **512 Bytes** of



EEPROM (100,000 Write/Erase Cycles). **1Kbyte** Internal SRAM

**I/O Ports:** 23 I/ line can be obtained from three ports; namely Port B, Port C and Port D.

**Interrupts:** Two External Interrupt source, located at port D. 19 different interrupt vectors supporting 19 events generated by internal peripherals.

**Timer/Counter:** Three Internal Timers are available, two 8 bit, one 16 bit, offering various operating modes and supporting internal or external clocking.

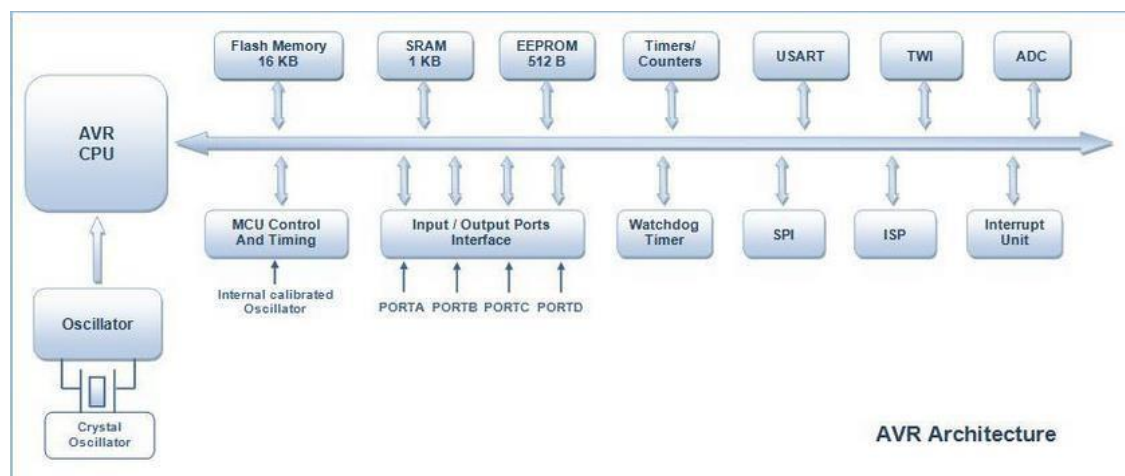
**SPI (Serial Peripheral interface):** ATmega8 holds three communication devices integrated. One of them is Serial Peripheral Interface. Four pins are assigned to Atmega8 to implement this scheme of communication.

**USART:** One of the most powerful communication solutions is USART and ATmega8 supports both synchronous and asynchronous data transfer schemes. It has three pins assigned for that. In many projects, this module is extensively used for PC-Micro controller communication.

**TWI (Two Wire Interface):** Another communication device that is present in ATmega8 is Two Wire Interface. It allows designers to set up a commutation between two devices using just two wires along with a common ground connection, As the TWI output is made by means of open collector outputs, thus external pull up resistors are required to make the circuit.

**Analog Comparator:** A comparator module is integrated in the IC that provides comparison facility between two voltages connected to the two inputs of the Analog comparator via External pins attached to the micro controller.

**Analog to Digital Converter:** Inbuilt analog to digital converter can convert an analog input signal into digital data of **10bit** resolution. For most of the low end application, this much resolution is enough.



**Microcontroller:** Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, Timers etc., required to perform some predefined task.

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store

some multimedia file or to access internet through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

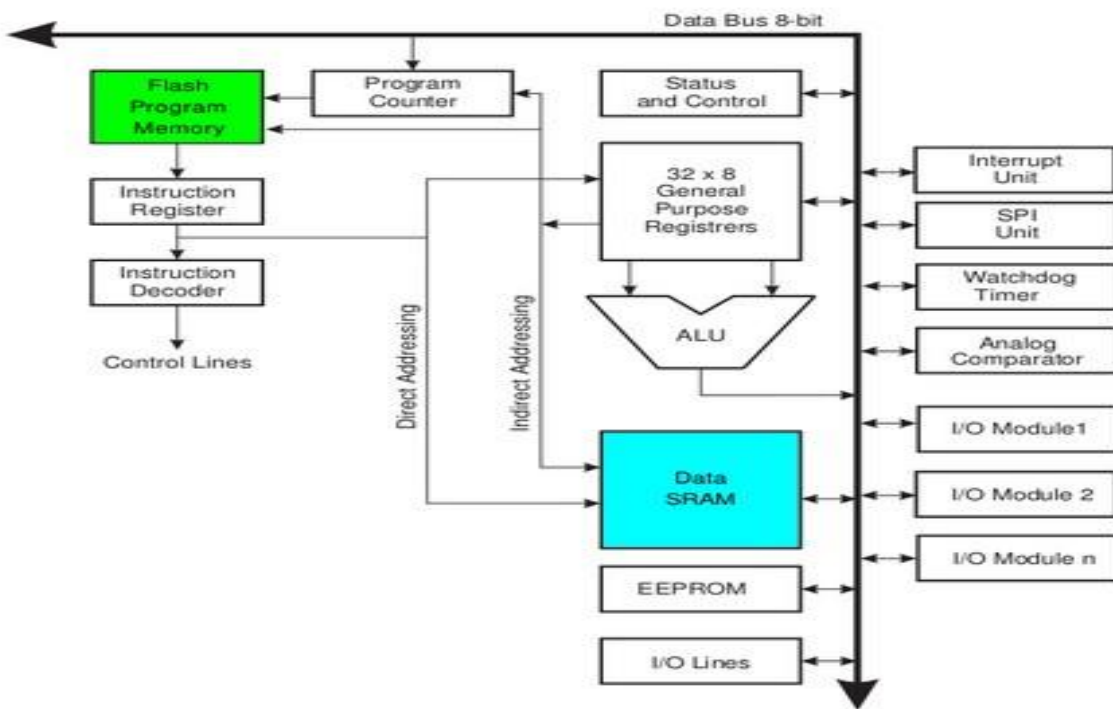
There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of these are 8051, **AVR** and PIC microcontrollers. In this article we will introduce you with **AVR** family of microcontrollers.

**AVR** was developed in the year 1996 by Atmel Corporation. The architecture of **AVR** was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for **Alf-Egil Bogen Vegard Wollan RISC microcontroller**, also known as **Advanced Virtual RISC**. The AT90S8515 was the first microcontroller which was based on **AVR architecture** however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

**AVR microcontrollers** are available in three categories:

1. **TinyAVR** – Less memory, small size, suitable only for simpler applications
2. **MegaAVR** – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.
3. **XmegaAVR** – Used commercially for complex applications, which require large program memory and high speed.

### 7.3 ARCHITECTURE:



### 7.4 DEVICE ARCHITECTURE:

Flash, EEPROM, and SRAM are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

#### *Program memory*

Program instructions are stored in non-volatile flash memory. Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.

The size of the program memory is usually indicated in the naming of the device itself. If (e.g., the ATmega64x line has 64 kB of flash while the ATmega32x line has 32 kB).

There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

## Internal data memory

The data address space consists of the register file, I/O registers, and SRAM.

## Internal registers

The AVR has 32 single-byte registers and is classified as 8-bit RISC device.

In most variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses ( $0000_{16}$ – $001F_{16}$ ) followed by the 64 I/O registers ( $0020_{16}$ – $005F_{16}$ ).

Actual SRAM starts after these register sections (address  $0060_{16}$ ). (Note that the I/O register space may be larger on some more extensive devices, in which case the memory mapped I/O registers will occupy a portion of the SRAM address space.)

Even though there are separate addressing schemes and optimized opcodes for register file and I/O register access, all can still be addressed and manipulated as if they were in SRAM.

In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes ( $0000_{16}$ – $0FFF_{16}$ ). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space which can be used optionally for mapping the internal EEPROM to the data address space ( $1000_{16}$ – $1FFF_{16}$ ). The actual SRAM is located after these ranges, starting at  $2000_{16}$ .

## 7.5 EEPROM

Almost all AVR microcontrollers have internal EEPROM for semi-permanent data storage. Like flash memory, EEPROM can maintain its contents when electrical power is removed.

In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions which makes EEPROM access much slower than other internal RAM.

However, some devices in the SecureAVR (AT90SC) family use a special EEPROM mapping to the data or program memory depending on the configuration. The XMEGA family also allows the EEPROM to be mapped into the data address space.

Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well designed EEPROM write routine should compare the contents of an EEPROM address with desired contents and only perform an actual write if the contents need to be changed.

Note that erase and write can be performed separately in many cases, byte-by-byte, which may also help prolong life when bits only need to be set to all 1s (erase) or selectively cleared to 0s (write).

#### Program execution

Atmel's AVR's have a two stage, single level pipeline design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVR's relatively fast among eight-bit microcontrollers.

The AVR processors were designed with the efficient execution of compiled C code in mind and have several built-in pointers for the task.

## 7.6 MCU SPEED

The AVR line can normally support clock speeds from 0 to 20 MHz, with some devices reaching 32 MHz. Lower powered operation usually requires a reduced clock speed. All recent (Tiny, Mega, and Xmega, but not 90S) AVR's feature an on-chip oscillator, removing the need for external clocks or resonator circuitry. Some AVR's also have a system clock prescaler that can divide down the system clock by up to 1024. This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.

Since all operations (excluding literals) on registers R0 - R31 are single cycle, the AVR can achieve up to 1 MIPS per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS. Loads and stores to/from memory take two cycles, branching takes two cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices

## 7.7 FEATURES:

- High-performance, Low-power Atmel®AVR® 8-bit

#### Microcontroller • Advanced RISC Architecture

- 130 Powerful Instructions – Most Single-clock Cycle Execution
- $32 \times 8$  General Purpose Working Registers
- Fully Static Operation
- Up to 16MIPS Throughput at 16MHz
- On-chip 2-cycle Multiplier

- **High Endurance Non-volatile Memory segments**

- 8Kbytes of In-System Self-programmable Flash program memory
- 512Bytes EEPROM
- 1Kbyte 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

- **Peripheral Features**

- Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture

Mode

- Real Time Counter with Separate Oscillator
- Three PWM Channels
- 8-channel ADC in TQFP and QFN/MLF package

Eight Channels 10-bit Accuracy

- 6-channel ADC in PDIP package

Six Channels 10-bit Accuracy

- Byte-oriented Two-wire Serial Interface
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator

- **Special Microcontroller Features**

- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated RC Oscillator
- External and Internal Interrupt Sources
- Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby

- **I/O and Packages**

- 23 Programmable I/O Lines
- 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF

- **Operating Voltages**

- 2.7V - 5.5V (ATmega8L)
- 4.5V - 5.5V (ATmega8)

- **Speed Grades**

- 0 - 8MHz (ATmega8L)
- 0 - 16MHz (ATmega8)

- **Power Consumption at 4Mhz, 3V, 25oC**

- Active: 3.6mA
- Idle Mode: 1.0mA
- Power-down Mode: 0.5μA **Brown-out Detector:**

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Brown-out Detection” on page 38 for details on how to configure the Brown-out Detector.

Internal Voltage Referencethe Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the

reference is kept on in sleep mode, the output can be used immediately. Refer to “Internal Voltage Reference” on page 40 for details on the start-up time. Watchdog Timer If the Watchdog Timer is not needed in the application, this module should be turned off.

If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Watchdog Timer” on page 41 for details on how to configure the Watchdog Timer. Port Pins When entering a sleep mode, all port pins should be configured to use minimum power.

The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock (clkI/O) and the ADC clock (clkADC) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section “Digital Input Enable and Sleep Modes” on page 53 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to  $VCC/2$ , the input buffer will use excessive power.

## **7.7 POWER-ON RESET:**

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever VCC is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

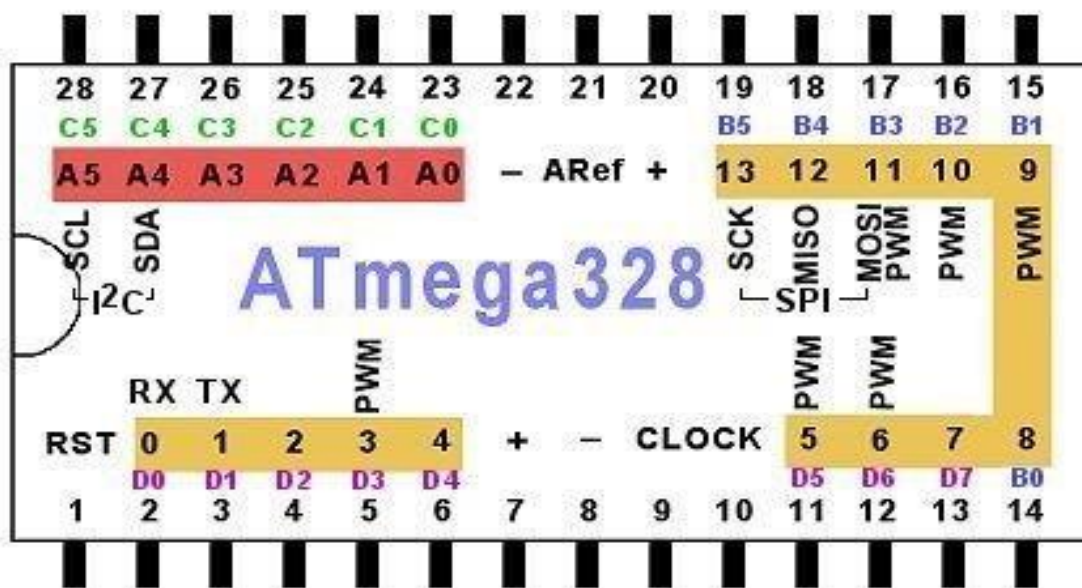
A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after VCC rise. The RESET signal is activated again, without any delay, when VCC decreases below the detection level.



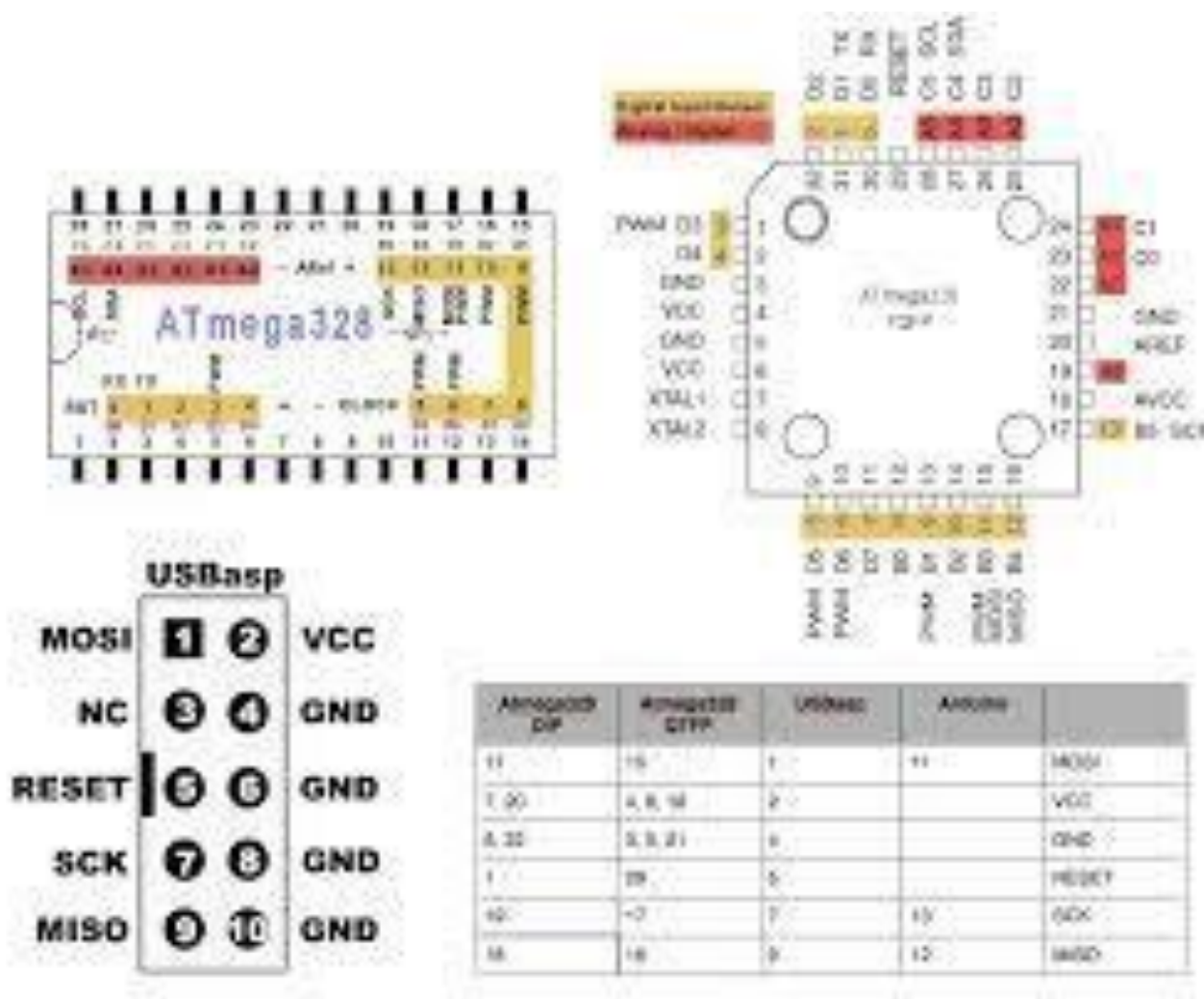
## 7.8 EXTERNAL RESET:

An External Reset is generated by a low level on the RESET pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage  $V_{RST}$  on its positive edge, the delay counter starts the MCU after the time-out period  $t_{TOUT}$  has expired.

## 7.9 Pin diagram:



Digital Input/Output  
Analog / Digital



**Fig.3.2.4.PIN DIAGRAM OF ATMEGA328**

### VCC

Digital supply voltage magnitude of the voltage range between 4.5 to 5.5 V for the ATmega 8 and 2.7 to 5.5 V for ATmega8L

### GND

Ground Zero reference digital voltage supply.

### PORTB (PB7.. PB0)

PORTB is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTB pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

### PORTC (PC5.. PC0)

PORTC is a port I / O two-way (bidirectional) 7-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the

pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTC pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

### **PC6/RESET**

If RSTDISBL Fuse programmed, PC6 then serves as a pin I / O but with different characteristics. PC0 to PC5 If Fuse RSTDISBL not programmed, then serves as input Reset PC6. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running.

### **PORTD (PD7.. PD0)**

PORTD is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTD pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

### **RESET**

Reset input pin. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running. Signal with a width of less than 1.5 microseconds does not guarantee a Reset condition.

### **AVCC**

AVCC is the supply voltage pin for the ADC, PC3 .. PC0, and ADC7..ADC6. This pin should be connected to VCC, even if the ADC is not used. If the ADC is used, AVCC should be connected to VCC through a low-pass filter to reduce noise.

### **Aref**

Analog Reference pin for the ADC.

### **ADC7 .. ADC6**

ADC analog input there is only on ATmega8 with TQFP and QFP packages / MLF.

### **PORTS**

Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status. Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller. Ports represent physical connection of Central Processing Unit with an outside world.

Microcontroller uses them

The Atmega8 has 23 I/O ports which are organized into 3 groups:

- Port B (PB0 to PB7)
- Port C (PC0 to PC6)
- Port D (PD0 to PD7)

We will use mainly 3 registers known as **DDRX**, **PORTX** & **PINX**. We have total four PORTs on my ATmega16. They are **PORTA**, **PORTB**, **PORTC** and **PORTD**. They are multifunctional pins. Each of the pins in each port (total 32) can be treated as input or output pin.

## APPLICATIONS

AVR microcontroller perfectly fits many uses, from automotive industries and controlling home appliances to industrial instruments, remote sensors, electrical door locks and safety devices. It is also ideal for smart cards as well as for battery supplied devices because of its low consumption.

EEPROM memory makes it easier to apply microcontrollers to devices where permanent storage of various parameters is needed (codes for transmitters, motor speed, receiver frequencies, etc.). Low cost, low consumption, easy handling and flexibility make ATmega8 applicable even in areas where microcontrollers had not previously been considered (example: timer functions, interface replacement in larger systems, coprocessor applications, etc.).

In System Programmability of this chip (along with using only two pins in data transfer) makes possible the flexibility of a product, after assembling and testing have been completed. This capability can be used to create assembly-line production, to store calibration data available only after final testing, or it can be used to improve programs on finished products.

## **8ARDUINO UNO**

### **8.1INTRODUCTION TO ARDUINO UNO**

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicating with software running on your computer. The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free. The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

### **8.2 WHAT IS ARDUINO?**

Arduino is an open source electronics platform accompanied with a hardware and software to design, develop and test complex electronics prototypes and products. The hardware consists of a microcontroller with other electronic components which can be programmed using the software to do almost any task. The simplicity of the Arduino language makes it very easy for almost everyone who has an interest in electronics to write programs without the understanding of complex algorithms or codes.

Arduino is intended for an artist, tinker, designer or anyone, interested in playing with electronics without the knowhow of complex electronics and programming skills. Arduino is an excellent designed open source platform. It has specially designed boards which can be programmed using the Arduino Programming Language (APL).

The presence of Arduino is not only spreading between hobbyists, but it has also expanded its roots in industries and used by experts for making prototypes of commercial products. Arduino takes off the efforts required in complex coding and designing hardware.

The open source nature of Arduino has been the main reason for its rapid horizontal growth. Since it is an Open Source project, all the files related to hardware and software is available for personal or commercial use. The development cost of the hardware is very small as against the costly similar proprietary products by the industrial giants. The open source nature doesn't require any licenses to develop, use, redistribute or even sell the product. But the Arduino name is trade mark protected (Arduino™) i.e., you are free to sell the Arduino board under any other name however in order to sell

it under the name “Arduino” you need to take permission from the founders and follow their quality terms.

The Software files which includes all the source code library are also open sourced. A user can modify them to make the project more versatile and improve its capabilities. This provides a strong online community support.

### 8.3 CONCEPT OF ARDUINO:

The root of Arduino goes deep down to the development of Processing Language by MIT researchers. Processing language is an open source language designed to introduce the software development environment for the artistic people without the need of deep knowledge of programming of algorithms. Processing is based on java.

In early year of 21<sup>st</sup> century, designing an electronics gadget was nearly impossible for a common man. The requirement of specific skill set and hefty prices of software and hardware created a full stop in the path of their creativity.

In year 2003 Hernando Barragan, a programmer developed an open source electronics development platform with software IDE, where anyone with a small knowledge in electronics and programming could use his project to give wings to their creativity. His focus was to reduce the burden of complexity in designing electronics hardware and software. The project was named as Wiring. The software IDE of the Wiring used processing language to write the codes.

As the program written in C\C++ is named as *Project*, in the same way the code written in Wiring (even in Processing and Arduino) is termed as *Sketch*. The name sketch gives a familiar look for an artist.

The principle idea behind Wiring is that one can make the sketch of their idea on Wiring software and implement it using specially designed Wiring board. You need to write a few lines of codes on the software IDE and then download the program to the onboard microcontroller to see the output.

Wiring has predefined libraries to make the programming language easy. Arduino uses these libraries. The predefined libraries are written in C and C++. One can even write his software in C\C++ and use them on *wiring* boards. The difference between writing a program in C/C++ and Wiring is that the Wiring *Application Programmable Interface* (API) has simplified programming style and the user doesn't require detailed knowledge of the concepts like classes, objects, pointers, etc. While sketching hardware you need to call the predefined functions and rest will be handled by the Wiring software.

The basic difference between the Processing and the Wiring is that the Processing is use to write the program which can be used on other computers while Wiring program is used on microcontrollers.

### 8.3 HISTORY:

Wiring is the predecessor of Arduino. Arduino was developed in Ivrea, Italy by Massimo Banzi and David Cuartielles in year 2005. The Project was named after Arduin of Ivrea (King of Italy). The project Arduino uses the Wiring language. The concept of Wiring Language was created by Hernando

Barragan, and under his supervision Massimo Banzi and David Cuartielles developed the Project Arduino.

### 8.4 OPEN SOURCE LICENSE

Arduino is an open source project which is probably the root cause reason for its popularity.

Arduino hardware design is an Open Source Hardware, distributed under *Creative Common Attribution* Share-Alike license. Creative Common, a non-profitable organization has released several copyleft licenses as free of charge, so that the creativity/ knowledge can be shared to the rest of the world while having the copyright to the authorized person. The originally designed files, like layout and schematics of Arduino products are available as Eagle CAD files.

The source code for its IDE and libraries are also available and released under GUN General Public License (known as GPL). The GPL is the first copyleft license for general use. The license is granted for the software to ensure the copyleft freedom.

### 8.65 ATMEL ATMEGA328P:

The ATmega328P chip is used in this project as the microcontroller. The significance of the first two digits is to stipulate that the AVR core consists of variety of instruction set with 32 general purpose working registers which are connected directly to the Arithmetic Logical Unit (ALU), tolerating two independent registers to be retrieved in one single instruction executed in one clock cycle. The subsequent architecture is more programmable efficient while attaining data transfer rates up to ten times quicker than other CISC microcontrollers. The last digit is to indicate the 8 bit bi-directional port. It is certainly the head of the system which is controlling the various modules. The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

The Arduino hardware was very skillfully designed to reduce the complexities arising in the circuitry. It has an In System Programmer (ISP), which allows users to transfer the software inside the microcontroller without removing it from the circuit. The basic model of an Arduino board consists of

an 8-bit AVR microcontroller along with some other necessary components like a 5 volt linear regulator IC, a 16 MHZ crystal, ceramic resonator, output connectors, direct adaptor input, etc.

The IO ports on boards are positioned in a way that it can be easily attached with the interchangeable add-on modules, known as shields. Shields are daughter boards that can be externally attached/ plugged with the Arduino boards to extent the board's capabilities. For example an xbee shield can be attached with the Arduino board to establish a wireless communication. A motor control shield can be attached on the top of Arduino board to run the motors or to provide an ease to control the speed of motors. The Arduino Board can easily interface with external sensors, circuits or other peripherals.

Arduino hardware is available in various designs and configurations depending on the use. The different configurations use different AVR chips, Atmega8/168/328/1280/2560. Each board has its own additional feature, like Arduino UNO consists of ATmega328 which communicates to PC via USB using FTDI chip. very comfortable for attaching shields. On the other Arduino NANO uses Atmega168/328 which also uses FTDI chip but is much comfortable to use it on breadboard.

Some non-ATmega Arduino boards are also available. These boards don't contain Atmel's ATmega controller but are compatible with Arduino shield. These microcontrollers cannot be programmed by the standard Arduino IDE but manufacturers do provide some other versions of Arduino IDE which includes the necessary libraries related to the controller. For example Leaf labs Maple based on 32bit arm processor or chip KIT UNO32 based on PIC micro controllers.

The earlier version of the Arduino board had controller with bootloader which communicated with the Arduino IDE mostly via a Serial port. Later a FTDI chip was introduced on the Arduino board which is a USB to serial converter to allow the communication with the USB port. And today the Arduino boards are available with Atmel's microcontroller which have inbuilt capacity to communicate with the USB port.

The high-performance Atmel Pico Power 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock



cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed (Atmel Corporation).

## 8.6 ARDUINO UNO:

To program the ATmega328P Microcontroller a Serial communicator is required. Serial communication is most widespread interface between microcontroller and computer. UART is one of the serial interfaces which are widely used. A Universal Asynchronous Receiver/Transmitter (UART) is a piece of computer hardware that translates data between parallel and serial forms. Classically, most serial interface from microcontroller to computer is done through serial port (DB9). However, since computer serial port used RS232 protocol and microcontroller used TTL UART, a level shifter is needed between these interfaces. There are several level shifters available in the market, some of which supports USB plug and play.

But in most of the times the level shifter are unstable to use due to its design and more than one software is required to convert the programming on C to hex or machine language and maybe another software to interface between the Microcontroller and computer.

Arduino UNO is an alternative to this solution, the internal board of Arduino consists of all the necessary ICs for communication. It is also build compact into a PCB which has connectors for fast and easy prototyping.

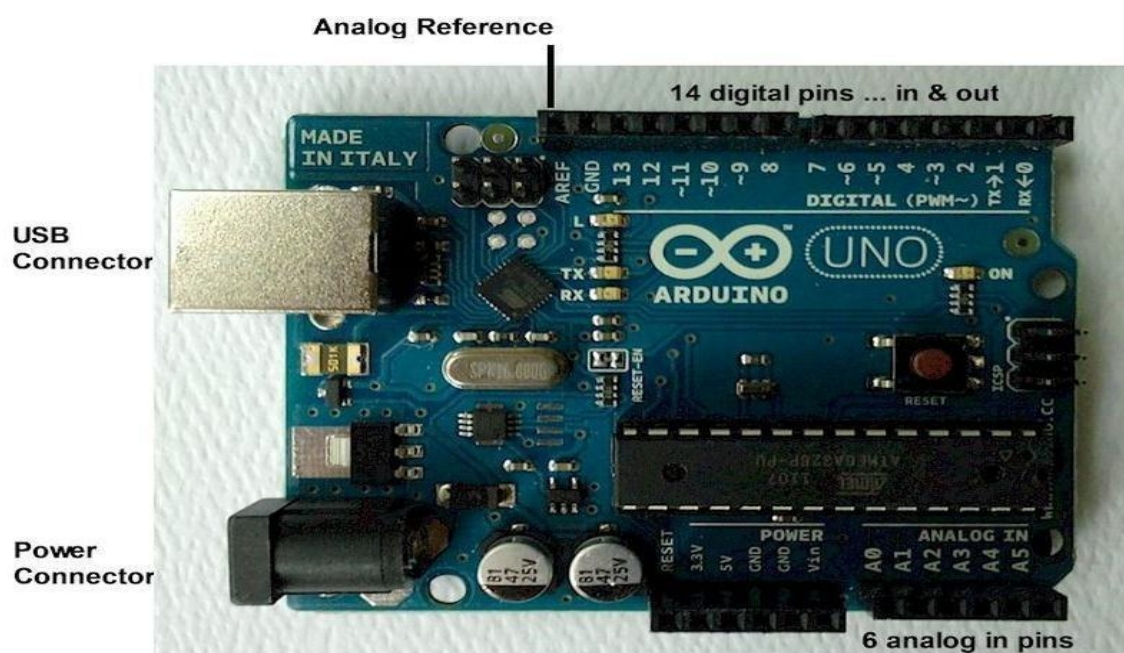


FIG 8.6: ARDUINO UNO

## 8.7 SUMMARY:

Microcontroller ATmega328

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 14 (of which 6 provide PWM output)

Analog Input Pins 6

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 32 KB (ATmega328) of which 0.5 KB used by bootloader

SRAM 2 KB (ATmega328)

EEPROM 1 KB (ATmega328)

Clock Speed 16 MHz

### 8.8.1 PIN CONFIGURATION:

**ATmega328P pin mapping**



**Fig:Pin Diagram**

- The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
- External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
- The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

**The power pins are as follows:**

- ⚡ **VIN:** The input voltage to the Arduino board when it's using an external power source (as Opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- ⚡ **5V:** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board.
- ⚡ **3V3:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- ⚡ **GND:** Ground pins.
- ⚡ **IOREF:** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

**8.8.2 MEMORY:**

The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM.

**8.8.3 INPUT AND OUTPUT:**

Each of the 14 digital pins on the Uno can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kohms. In addition, some pins have specialized functions:

- ✦ **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.
- ✦ These pins are connected to the corresponding pins of the ATmega8U2 USB-to- TTL Serial chip.
- ✦ **External Interrupts:** 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.
- ✦ **PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the `analogWrite()` function.
- ✦ **SPI:** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- ✦ **LED: 13.** There is a built- in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the `analogReference()` function. Additionally, some pins have specialized functionality:

- ✦ **TWI:** A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- ✦ **AREF.** Reference voltage for the analog inputs. Used with `analogReference()`.
- ✦ **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

### 8.8.3 COMMUNICATION:

Microcontrollers depend on a host computer for developing and compiling programs. The software used on the host computer is known as an integrated development environment, or IDE. For the Arduino, the development environment is based on the open source Processing platform ([www.processing.org](http://www.processing.org)) which is described by its creators as a “programming language and environment for people who want to program images, animation, and interactions.

The Arduino programming language leverages an open source project known as Wiring ([wiring.org.co](http://wiring.org.co)). The Arduino language is based on good old- fashioned C. If you are unfamiliar with this language, don't worry; it's not hard to learn, and the Arduino IDE provides some feedback when you make mistakes in your programs.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a info file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

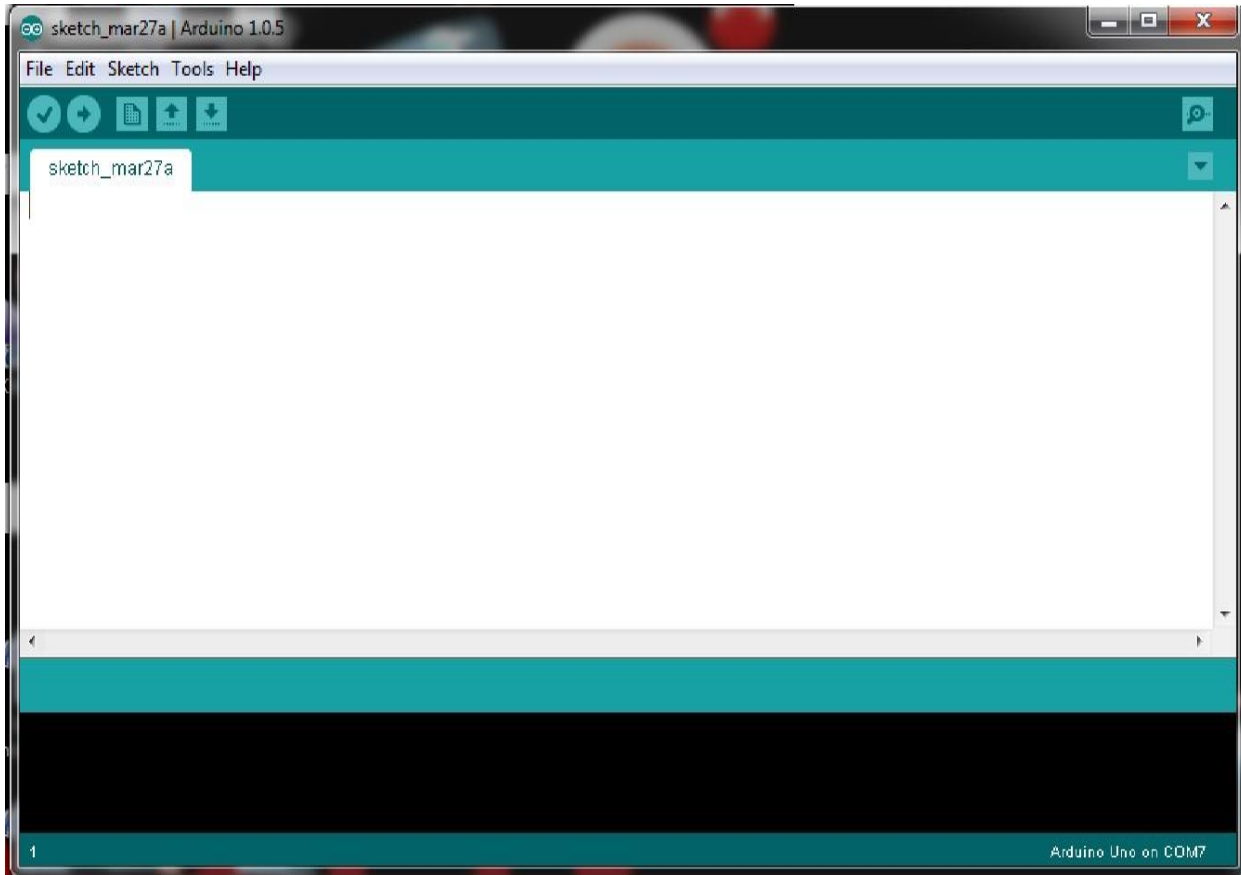
As you go through the list of programming statements available in the Arduino IDE (choose Help->Reference), you might think there isn't much power for doing things like running servos, operating stepper motors, reading potentiometers, or displaying text on an LCD. Like most any language based on C, the Arduino supports the notion of "libraries" code repositories that extend core programming functionality. Libraries let you re-use code without having to physically copy and paste it into all your programs.

## **8.9 ARDUINO IDE**

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It includes a code editor which is capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

## FOLLOWING ARE THE STEPS INVOLVED:

### 1. OPEN ARDUINO IDE AS SHOWN BELOW



**FIG:8.9.1:open Arduino Ide**

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much efficient. Users only need define two functions to make a runnable cyclic executive program:

setup(): a function run once at the start of a program that can initialize settings

loop(): a function called repeatedly until the board powers off

## 2. SELECT THE COM PORT FROM TOOLS

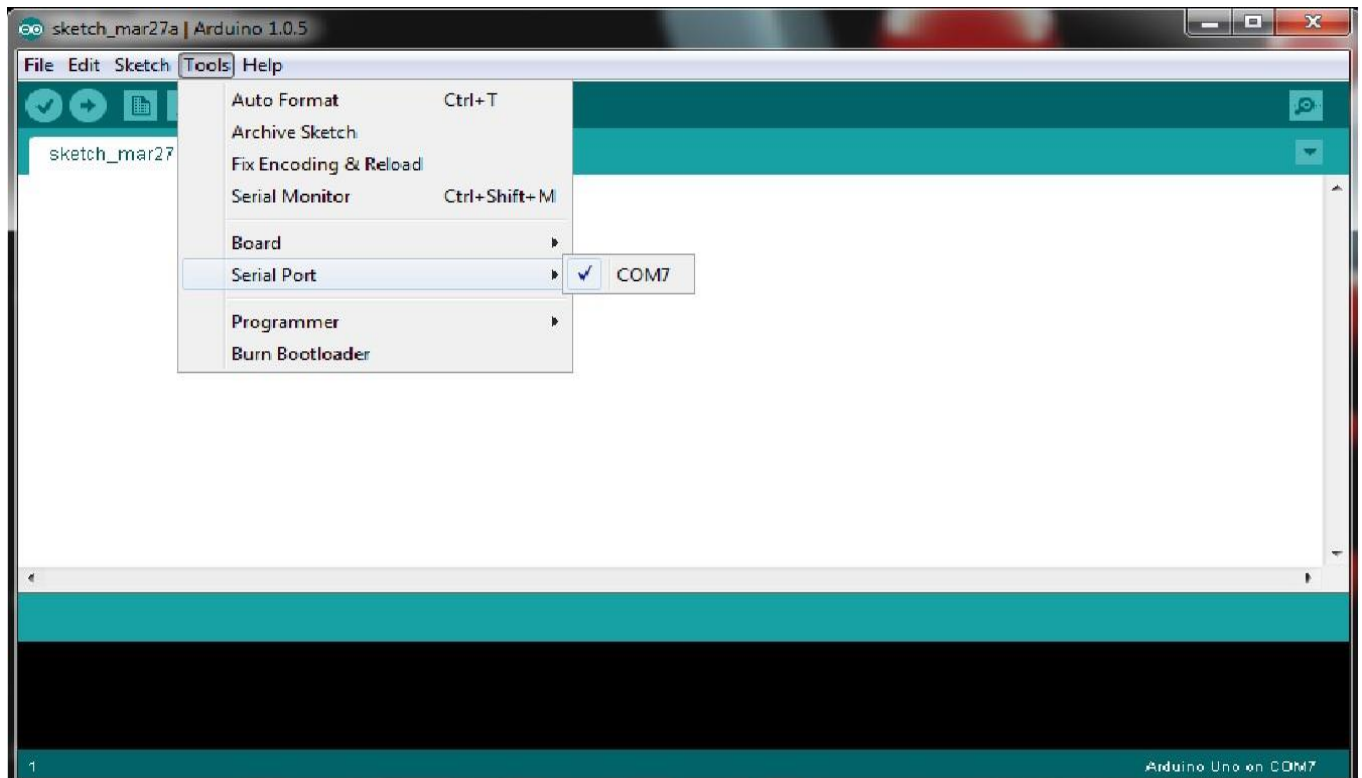
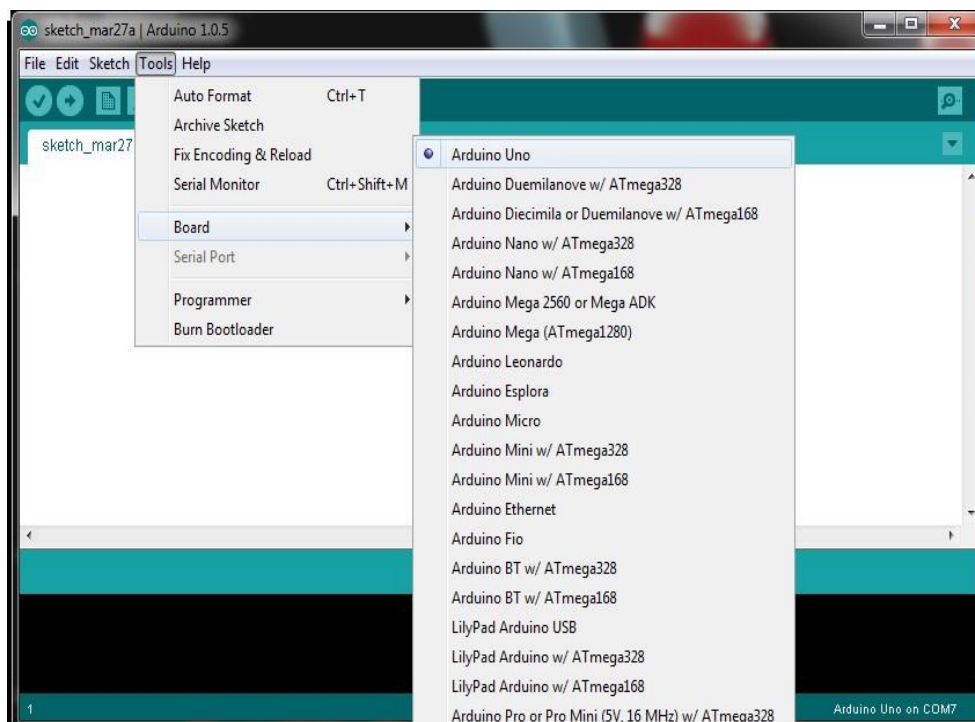
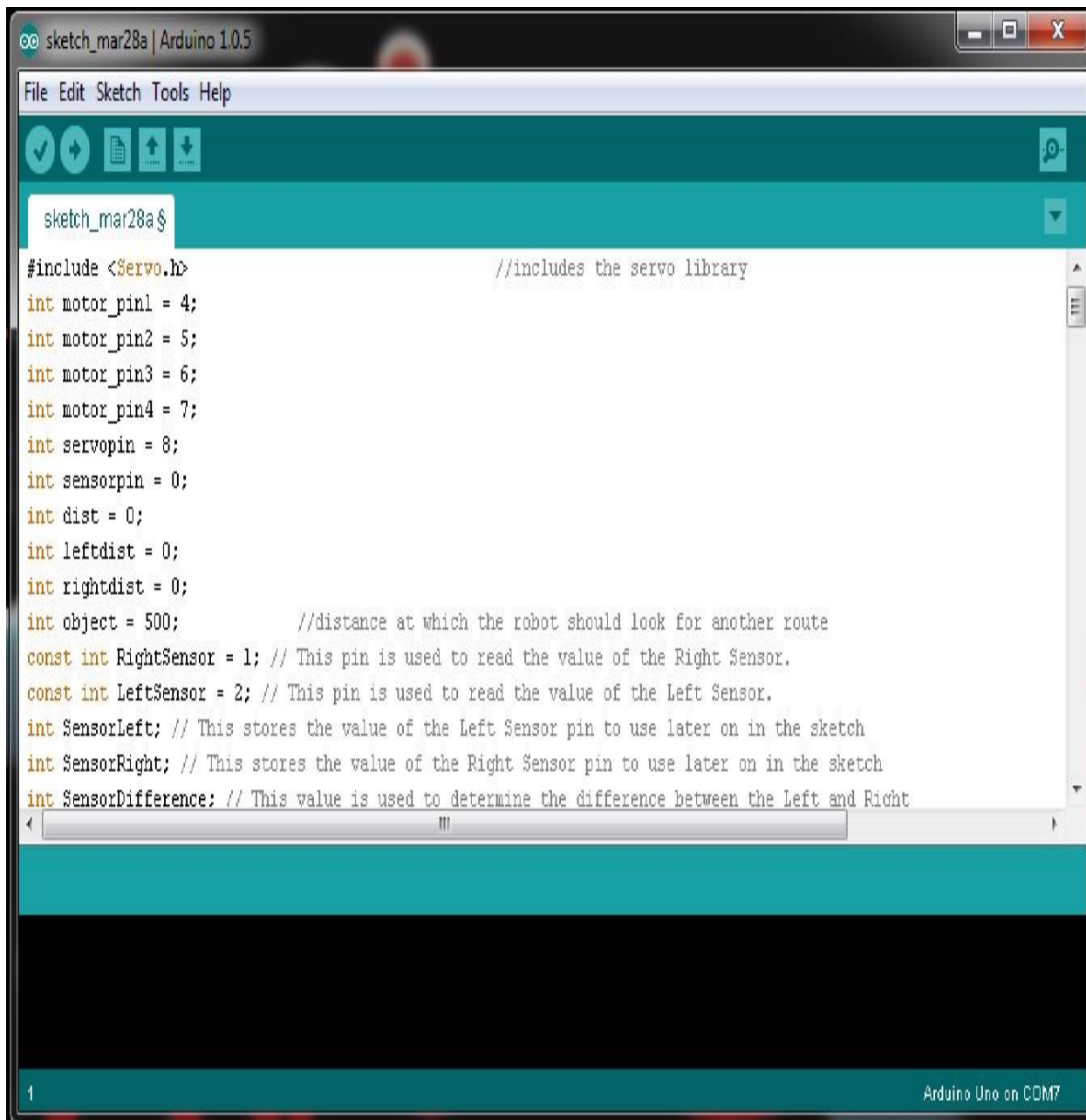


FIG:8.9.2 SELECT THE REQUIRED ARDUINO BOARD FROM TOOLS

## 3. SELECT THE REQUIRED ARDUINO BOARD FROM TOOLS



#### 4. WRITE THE SKETCH IN ARDUINO IDE



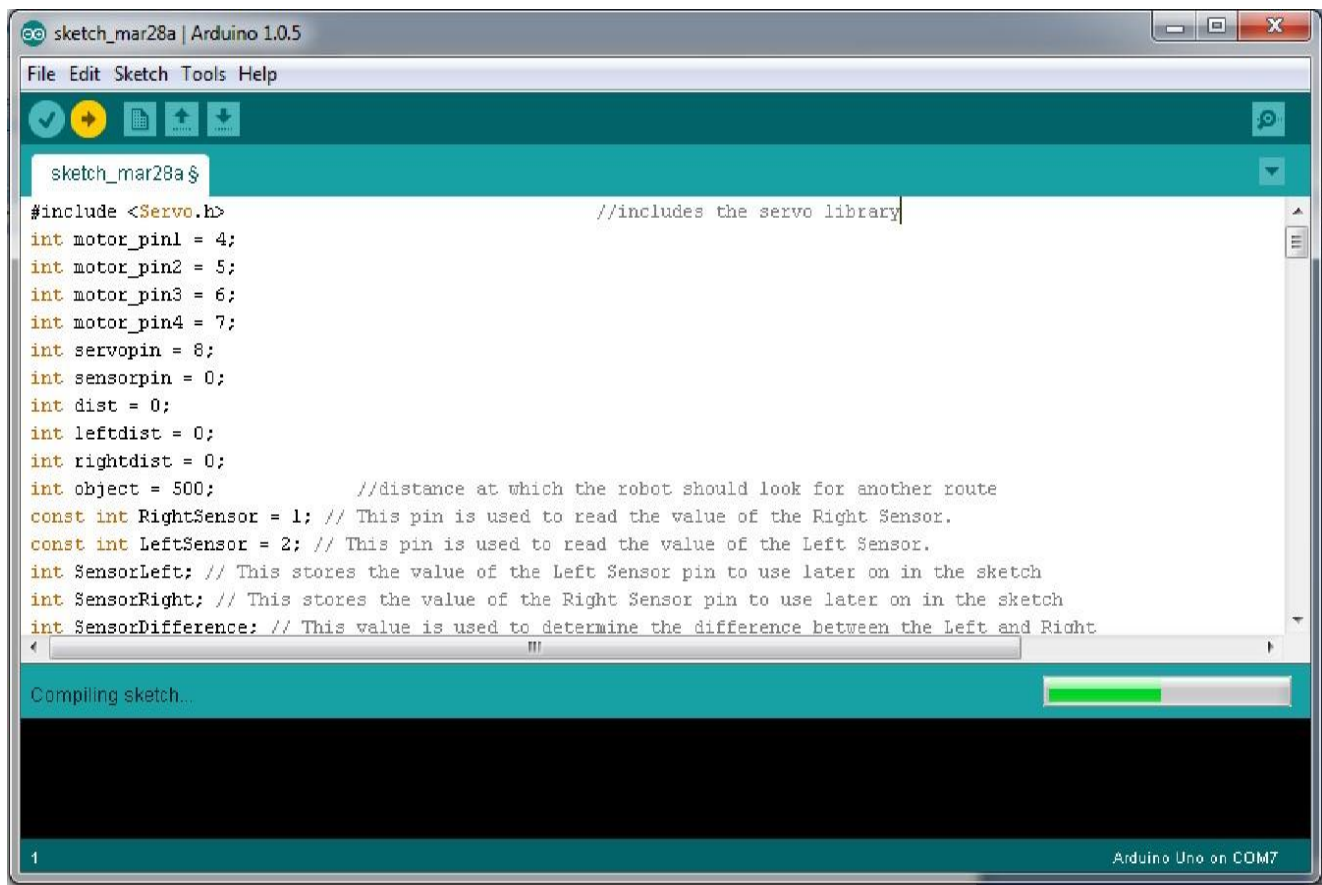
The screenshot shows the Arduino IDE interface with a sketch named 'sketch\_mar28a'. The code includes the Servo library and declares several variables for pins, distances, and sensor values. The code is as follows:

```
sketch_mar28a$
#include <Servo.h>                                //includes the servo library
int motor_pin1 = 4;
int motor_pin2 = 5;
int motor_pin3 = 6;
int motor_pin4 = 7;
int servopin = 8;
int sensorpin = 0;
int dist = 0;
int leftdist = 0;
int rightdist = 0;
int object = 500;                                //distance at which the robot should look for another route
const int RightSensor = 1; // This pin is used to read the value of the Right Sensor.
const int LeftSensor = 2; // This pin is used to read the value of the Left Sensor.
int SensorLeft; // This stores the value of the Left Sensor pin to use later on in the sketch
int SensorRight; // This stores the value of the Right Sensor pin to use later on in the sketch
int SensorDifference; // This value is used to determine the difference between the Left and Right
```

The IDE status bar at the bottom indicates '1' and 'Arduino Uno on COM7'.



## 5. COMPILE AND UPLOAD THE SKETCH TO ARDUINO BOARD



## CHAPTER 9

### BLYNK

#### The Blynk platform

## The Blynk Platform

- Blynk is a platform for the development of smartphone applications that work with a wide range of microcontrollers.
- No mobile programming needed.
- Concentrate on the functionality.
- Free.




Mobile Development with Blynk


With [Blynk](#), you can create smartphone applications that allow you to easily interact with microcontroller ~~sorevenfullcomputers~~ such as the [Raspberry Pi](#).

The main focus of the Blynk platform is to make it super-easy to develop the mobile phone application. As you will see in this course, developing a mobile app that can talk to your Arduino is as easy as dragging a widget and configuring a pin.

With Blynk, you can control an LED or a motor from your mobile phone with literally zero programming. This is actually the first experiment that I will demonstrate in this course.

But don't let this simplicity make you think that Blynk is only useful for trivial applications. Blynk is a robust and scalable tool that is used by hobbyists and the industry alike.

You can use it to monitor the soil humidity of your vegetable garden and turn on the water, or open up your garage door, with your phone.

You can also use it to control smart furniture that can learn from your routines, or embed IoT and AI to traditional industrial products such as a boiler, or for improving the integrity and safety of oilfields.

Blynk is free to use for personal use and prototyping. Their business model generates profits by selling subscriptions to businesses that want to publish Blynk-powered apps for their hardware products or services.

Let's take a closer look at each component of the Blynk Platform.

### 9.1 The Blynk smartphone app

## The Blynk Platform: smartphone app

- The app is really an app editor.
- Use it to create Blynk projects.
- Each project may contain multiple widgets and connect to multiple devices.
- Projects are shareable.



Mobile Development with Blynk





The #1 Blynk innovation is the smartphone app.

The Blynk app is really an app editor.

It allows you to create one or more projects.

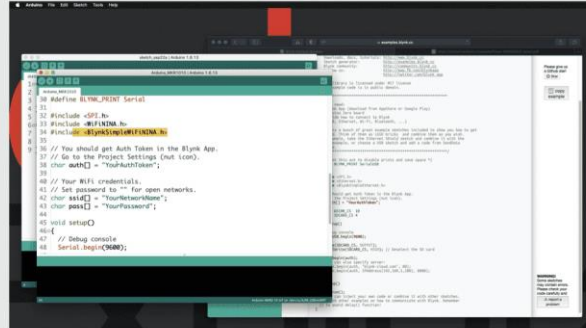
Each project can contain graphical widgets, like virtual LEDs, buttons, value displays and even a text terminal, and can interact with one or more devices. With the help of the Blynk library, it is possible to control [Arduino](#) or [ESP32](#) pins directly from your phone, without having to write any code at all.

It is also possible to share a project with friends and even customers so that they can access the connected devices but not be able to modify the project. Imagine a scenario where you build a smartphone application where you can control lights, window blinds and room temperature from your phone. You can share the project with other family members so that they can also access the functionality.

## 9.2 The Blynk microcontroller libraries

# The Blynk Platform: MCU libraries

- Blynk library available for many devices:
  - Arduino MKR1010, Uno, Nano, Zero etc.
  - ESP32/8266, BBC: micro, Teensy, Microduino, Digistump etc.
  - Particle Core, Photon etc.
- Blynk library available for Javascript, Python or Lua clients.
- Connectivity via USB(!), Ethernet, Wifi, BLE, 3G, LTE etc.



Supported hardware: <https://txplo.re/x5i>

Blynk library (Arduino): <https://txplo.re/voy>

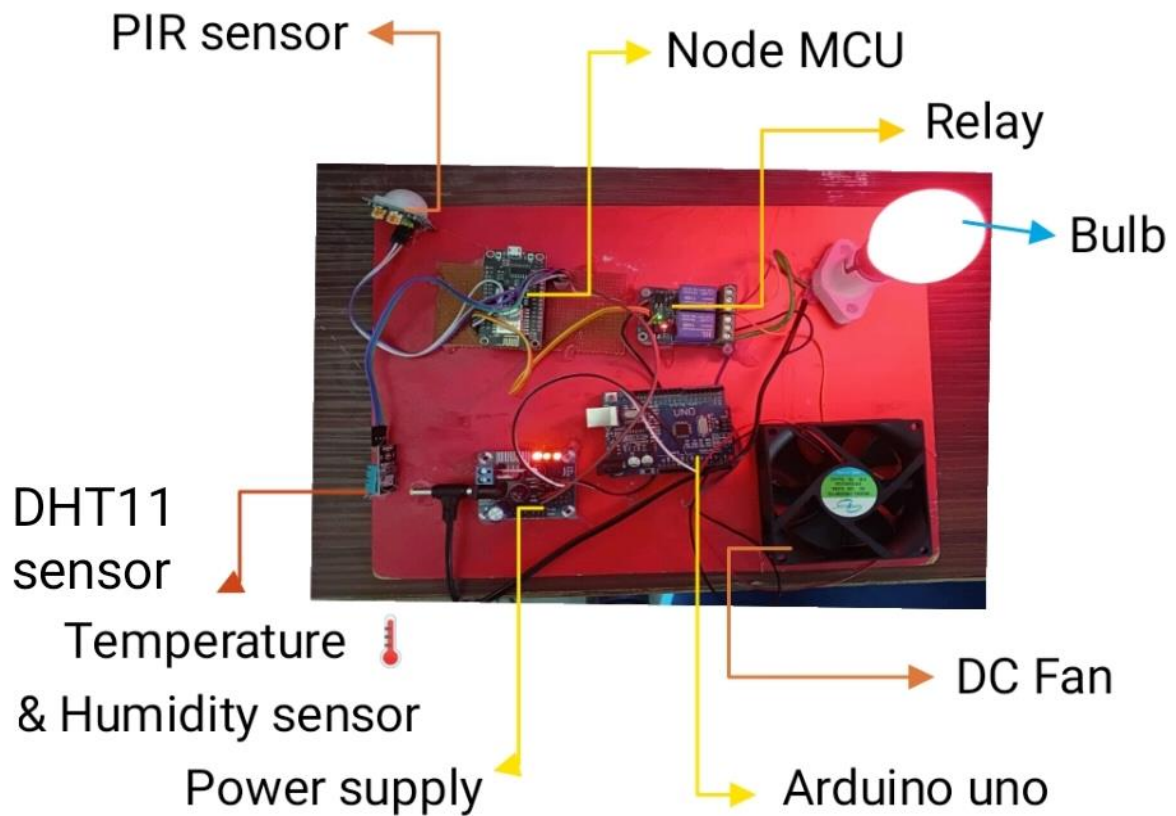
Mobile Development with Blynk

Tech  
Explorations



## CHAPTER 10

### RESULT AND DISCUSSION



○ Circuit prototype

## WORKING

The system consists of three isolated sub-systems: first subsystem consisting of GPS module to get the geo-location, second sub-system consisting of multiple sensors DHT11 temperature sensor to measure temperature, PIR sensor to detect motion and ultrasonic sensor to measure the distance, and the third sub- system consisting of a master microcontroller which function as the central coordinator that communicates with other subsystems via Wi-Fi. The master microcontroller is also interfaced with a relay module to control the appliances at the site. The sensor data are fetched to the user interface facilitated by smartphones or tablets from the various sensors using a raspberry pi as the private server.

Basically, control of turning ON or OFF the whole system is at owners hand. As the system gets powered up, it searches for the preset SSID (Service Set Identifier) and connects automatically to the Internet otherwise remains offline and performs the automated-controlling job that doesn't require commands from the owner.

Sensors accumulate disparate ambient-conditions and transmit them to the Microcontroller which processes the data transmitted by each sensor separately and then concurrently send the acquired data to the web server. The readings of each sensor can be accessed by the user from any place at any time. Additionally, all the sensors data are logged per second for future data analysis purpose. Data-logging is done both in microSD card and in the server.

The system operates in two modes automatic mode and manual mode. When it is set at automatic mode, all the home appliances like fan, heater etc. are automated to operate as per the surrounding environmental conditions sensed by the sensors. On the other hand, when it is set at manual mode, the user can locally or remotely monitor and control each of the home appliances via her smart phone or from her officedesktop PC. To recapitulate, the surveillance and control of entire household appliances is under her finger tip.

## **CHAPTER 11**

### **CONCLUSION & FUTURE SCOPE**

#### **CONCLUSION:**

In This project we present the design and implementation of LDR/IR vehicular robot using Arduino. Robot designed in my project is reaches the final destination (target) by avoiding obstacles in between the robot and the target with the help of Arduino. Arduino uses light dependent resistor which depends on the light intensity of source, Infrared Sensor to detect and avoid objects in its path and servo motor it consists of DC motors with position feedback that means you can tell the Arduino through your code to move the servo to the desired position. An Arduino is attached with the robot gets the incoming message from the LDR and IR SENSOR controls the movement of the robot accordingly.

#### **FUTURE SCOPE:**

This project work can be used to reach the desired location by avoid obstacles in planets like mars to detect metals.

## BIBLIOGRAPHY

1. [www.arduino.cc/unos](http://www.arduino.cc/unos)
2. [http://makeitinyourlibrary.org/technology/basic-arduino-robot-light seeker](http://makeitinyourlibrary.org/technology/basic-arduino-robot-light%20seeker)
3. <http://www.zagrosrobotics.com/shop/item.aspx?itemid=868>
4. [http://communityofrobots.com/tutorial/kawal/how-make-your- first-robot-using-Arduino](http://communityofrobots.com/tutorial/kawal/how-make-your-first-robot-using-Arduino)
5. <http://www.instructables.com/id/Line-seeking-Robot-with-Arduino/>
6. <http://www.instructables.com/id/Robot/>
7. Arduino Cookbook, 2nd Edition by Michael Margolis.
8. [www.ieee.org](http://www.ieee.org)
9. [www.wikipidea.org](http://www.wikipidea.org)
10. [www.google.com](http://www.google.com)



## APPENDEX-A

### SOFTWARE CODING

```

#include <Servo.h>
#include "DHT.h"
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define DHTPIN D1
#define DHTTYPE DHT11
#define pir D0
WidgetLED relay1(D2);//gpio 16
WidgetLED relay2(D3);//gpio 05
int p;
char auth[] = "AFWSOvnAwWHv8VNYJ8XAx1deGnxABAz0";

char ssid[] = "projects";
char pass[] = "projects1234";
BlynkTimer timer;
int cnt =0,d;
DHT dht(DHTPIN,DHTTYPE);
float humidityData;
float temperatureData;
void sendSensor()
{
    humidityData = dht.readHumidity();
    temperatureData = dht.readTemperature();
    p = digitalRead(pir);
    delay(30); // interval
    Serial.print("TEMP:");
    Serial.println(temperatureData);
    Serial.print("HUMIDITY:");

```

```

Serial.println(humidityData);
Blynk.virtualWrite(V0, temperatureData);
Blynk.virtualWrite(V1, humidityData);
if(p == LOW)
{
  Blynk.virtualWrite(V2,"HUMAN DETECTED" );
  Serial.print("HUMAN yes");
  delay(2000);
}
else
{
  Blynk.virtualWrite(V2,"NO HUMAN " );
  Serial.print("NO HUMAN");
}
if (relay1.getValue()) {
  relay1.on();
  Serial.println("relay1 on V1: ON");
} else {
  relay1.off();
  Serial.println("relay1 on V1: OFF");
}
if (relay2.getValue()) {
  relay2.on();
  Serial.println("relay2 on V2: ON");
} else {
  relay2.off();
  Serial.println("relay2 On V2: OFF");
}

}

void setup()
{
  pinMode(pir,INPUT);
  Serial.begin(115200);

```

```
delay(10);  
dht.begin();  
Blynk.begin(auth, ssid, pass);  
timer.setInterval(1000L, sendSensor);}  
void loop()  
{  
  Blynk.run();  
  timer.run();  
}
```