

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 MOTIVE**

Physical computing systems are nothing but embedded systems which perform tasks based on preset and predefined logic. Sensors connected to these systems acts like eye, nose, ear, etc. Based upon various types of sensors and their outputs, embedded systems perform tasks accordingly. In most of the cases, information gathered from the physical world using sensors is used to perform some task locally on the system. However, when we start thinking in terms of many such interconnected systems and making sense out of all the information generated, we start working on a totally different level.

On a broad level, there are local embedded systems including sensors, which sense various environmental parameters as well as physical and chemical parameters of the pipeline. These systems would get deployed over many kilometers of pipeline at predefined intervals. All these physical computers send individual data to a central server located in the head office of their company where the bigger picture out of multiple data-points gets visualized. The logical system part consists of a Web application that would do the number crunching job and then provide meaningful and actionable output to maintenance staff.

### **1.2 OBJECTIVES**

The objective of Internet Of Things is to extend the quantity of devices as the amount of the connected devices will sharply rise from several billions to over hundreds of billions, including a multitude of equipment's, sensors, actuators, vehicles, and devices attached with

RFID and to extend the type of devices as networking devices (networking elements) may be powered by the electronic power directly or by batteries; the computation and communication capacity may be greatly different, e.g., some devices even may not have any computational ability and also extensiveness in the connection mode to make the devices connected in a wired or wireless mode; the communication could be a single hop or multiple hops; the connection can be strong state routing or statistical weak state routing. Thus, in such a large-scale heterogeneous network, we must meet the challenge of highly-efficient interconnection of network elements. This project is a prototype of Internet of Things (IOT) using Raspberry Pi.

### 1.3 THESIS ORGANIZATION

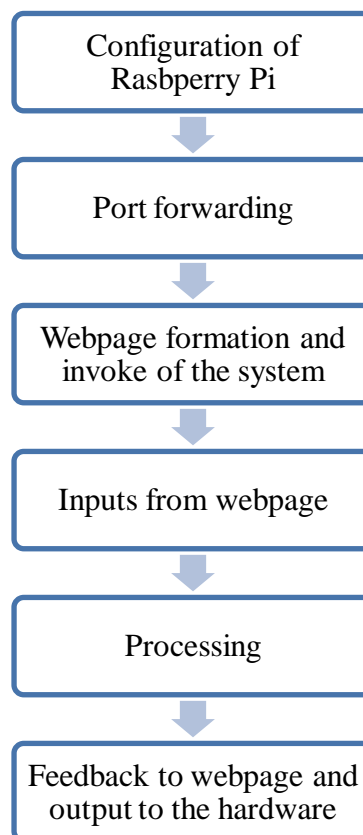


Figure 1.1 Block diagram of the entire system

In this chapter we have discussed about the overview of the project and the main objective behind using this project.

**Chapter 2:** It deals with real time systems its architecture and how the project is divided. Here the details of arrival to the employed complete embedded system for the purpose of project is described with each of the blocks of the system discussed in detail in further chapters.

**Chapter 3:** The moment the system is switched on, a local IP address is assigned to Raspberry using which a web page is seen on the screen. The web page seen has columns where the input buttons have been provided. This input buttons on click runs the shell script commands on Raspberry Pi. The commands are sent to pi using PHP.

**Chapter 4:** A firewall is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. By using the IP address of the DHCP router few ports are opened.

**Chapter 5:** The inputs are processed in the processing unit of the Raspberry Pi and the values are passed to the hardware connected to its GPIO pins.

**Chapter 6:** It speaks about one of the most important topics of the project, communication of Raspberry Pi with a robot which evidently covers the architecture of the robot too. The commands generated by Raspberry Pi are to be communicated to the robot. The way the received information is perceived by the robot and all other parameters will also be discussed in this chapter.

**Chapter 7:** It deals with the description of the Hardware used in this project i.e., complete description of all the components used and specifications of them. This chapter does not

include complete datasheet of any of these components but lists only their features and of them the features being made use of in the chapter are discussed.

**Chapter 8:** It deals with the description of software we have used and steps to use them to make the project.

**Chapter 9:** It deals with the results of all the work related to the project and conclusion of how the present system can be implemented in real-time.

**Chapter 10:** It deals with the Scope for future work.

Bibliography deals with text books referred along with the websites which provided us useful content for the project.

Appendix of the project deals with the algorithm related to the project, circuitry and circuit board used, its layout, USB programmer and USB to TTL converter.

## 1.4 CONCLUSION

After providing a brief description on this project and what this project will achieve, it gives the insights of Internet of Things with the exemplified applications. We also hope that this research can come up with a system that can access anything from anywhere at any time.

## CHAPTER 2

# ARCHITECTURE

### 2.1 INTRODUCTION

The working of the Internet of Things using Raspberry Pi is divided into 4 stages, they are

1. Web page formation and invoke of the system.
2. Port forwarding in the router.
3. Processing the inputs in the Raspberry Pi.
4. Outputs from the hardware.

#### Stage 1:

The moment the system is switched on the IP address of the personal network is found and by using this IP as URL a web page is opened on the screen.

The web page seen has columns where the input buttons have been provided. This input buttons on click runs the shell script command. The command is routed to Pi using PHP.

#### Stage 2:

A firewall is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. Using the IP address of the DHCP router, It can be configured to open few ports.

#### Stage 3:

On opening ports, inputs that are given on the web page are received by the router and been transferred to its client i.e. Raspberry Pi.

The inputs are processed in the processing unit of the Raspberry Pi and passed the values to the hardware connected to its GPIO pins.

#### Stage 4:

This is the final stage of the process where commands are executed by Raspberry Pi and result is reflected to the hardware.

Since the commands include giving and extraction of data, we also need a correct feedback system to display the appropriate sensors value on the webpage and also status of the pins can be displayed.

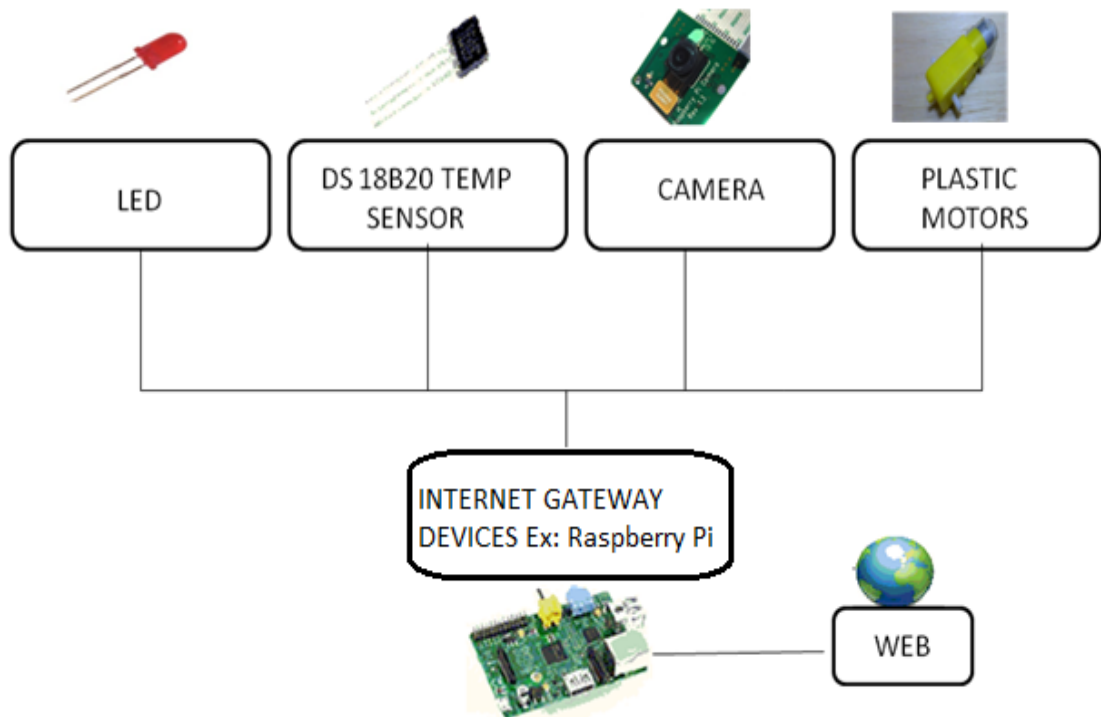


Fig 2.1 Block diagram

From the above figure (Fig 2.1) it is quite clear that the Raspberry Pi is the core of the project

We have divided the system into two main divisions

- Webpage and internet.
- Central system

## 2.2 WEBPAGE AND INTERNET

Here the web page (or webpage) is a web document that is suitable for the World Wide Web and the web browser, which collects input from user and transfer that to hardware over internet. In the proto type we have used a simple webpage that uses html, css for the formation and PHP for the transfer of data.

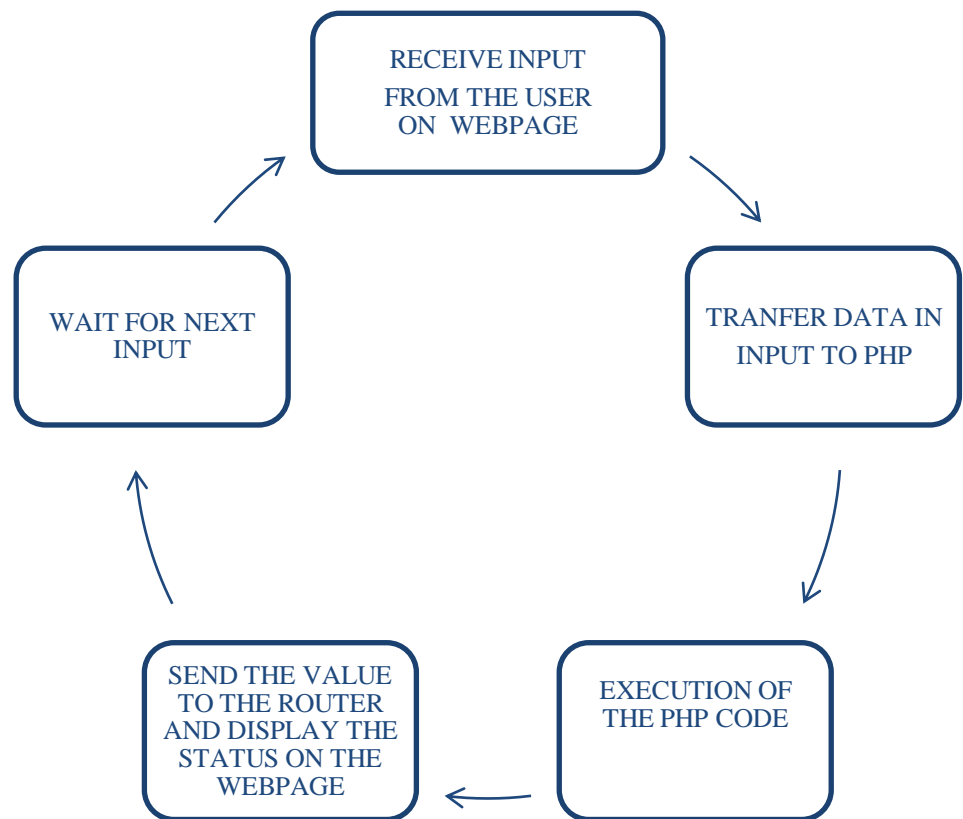


Fig 2.2 Process behind Webpage

A Web page is a simple text file that contains not only text, but also a set of HTML tags that describe how the text should be formatted when a browser displays it on the screen. The tags are simple instructions that tell the Web browser how the page should look when it is displayed. The tags tell the browser to do things like change the font size or color, or arrange things in columns. The Web browser interprets these tags to decide how to format the text onto the screen.

PHP is a programming language that is used to send inputs from a web page to the server and the commands are executed on the server.

### 2.2.1 ROUTER

Router acts like a gate way between our two systems. The input that is been given through the webpage has to be transmitted to the hardware either wired or wireless this will be done by a router. In order to access the system globally that is from anywhere on the world we need to open https port in the router that controls output of the system.

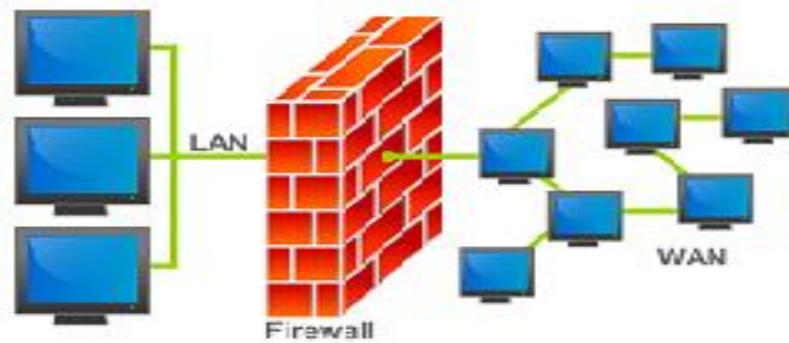


Fig 2.3 Firewall

Central system (Raspberry Pi) and router can be connected wired by using wire between them or wireless using a Wi-Fi dongle which is connected to Raspberry Pi.

## 2.3 CENTRAL SYSTEM

This can also be called as server with hardware connected, of the entire Production/Assembly system, where the central system will be having the full authority on all the assembly equipment, Objects and the Robot will be functioning on the commands given by the web system.

Initially central system connected to objects (equipment or material) is connected to the router using wire. Router is configured accordingly so the commands can be passed to our Raspberry Pi which is converted to a web server.

### 2.3.1 CONFIGURATION OF CENTRAL SYSTEM

The configuration of central system represented below consist of a Raspberry Pi installed with Raspbian operating system , wiring pi library, apache and PHP software and hardware such as temperature sensor, dot matrix, camera and an embedded robot system. Raspberry Pi is connected to a network from the router using wire, the values of input are transmitted from the webpage system to the hardware after processing in the Raspberry Pi.

Raspberry Pi can be accessed, that is given commands using an SSH putty software that should be installed in the laptop or PC. By knowing the IP address of the Raspberry Pi from the router configurations the SSH system can be operated and the desktop can also be



seen using another software named VNC server. This software will avoid Raspberry Pi to be connected to any display unit directly from HDMI port. The hardware is connected to GPIO (General Purpose Input Output) pins on the Raspberry Pi.

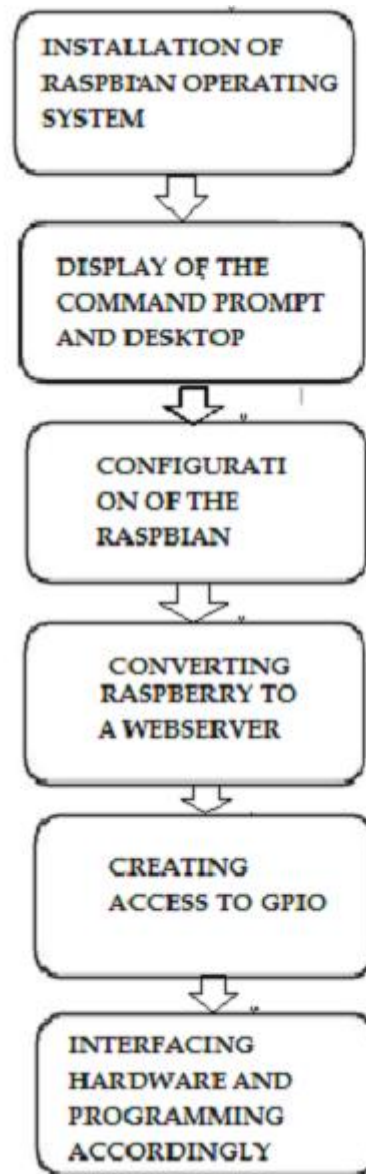


Fig 2.4 configuration of central system

## 2.4 CONCLUSION

The architecture of project and an over view of all the building blocks of the project are discussed.

## CHAPTER 3

### WEBPAGE FORMATION AND INVOKE OF THE SYSTEM

#### 3.1 INTRODUCTION

A webpage is a web document that is suitable for the World Wide Web and the web browser. A web browser displays a web page on a monitor or mobile device. This chapter includes the formation of webpage and source code running behind the html page.

#### 3.2 PURPOSE

When it comes to a website a webpage which is accessible to users ,clients, should be well designed in which they should be provided with all the facilities to operate hardware effectively.

#### 3.3 WEB PAGE

A webpage is a web document that is suitable for the World Wide Web and the web browser. A web browser displays a web page on a monitor or mobile device. The web page is what displays, but the term also refers to a computer file, usually written in HTML or comparable markup language. Web browsers coordinate the various web resource elements for the written web page, such as style sheets, scripts and images, to present the web page.

On a network, a web browser can retrieve a web page from a remote web server. On a higher level, the web server may restrict access to only a private network such as a corporate intranet or it provides access to the World Wide Web. On a lower level, the web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server-side software or client-side scripting. Dynamic web pages help the browser (the client) to enhance the web page through user input to the server.

In order to graphically display a web page, a web browser is needed. This is a type of software that can retrieve web pages from the Internet. Most current web browsers include the ability to view the source code. Viewing a web page in a text editor will also display the source code.

### **3.4 CREATION OF WEB PAGE**

To create a web page, a text editor or a specialized HTML editor is needed. In order to upload the created web page to a web server, traditionally an FTP client is needed.

The design of a web page is highly personal. A design can be made according to one's own preference, or a premade web template can be used. Web templates let web page designers edit the content of a web page without having to worry about the overall aesthetics. Many people turn to all-in-one sites for web domain purchases, web hosting service and templates to build customized websites. Web publishing tools such as Tripod and Wordpress offer free page creation and hosting up to a certain size limit. Other ways of making a web page is to download specialized software, like a Wiki, CMS, or forum. These options allow for quick and easy creation of a web page which is typically dynamic. Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts.

### **3.5 SOURCE CODE**

On clicking on a button on the html page the source running behind is PHP script. In computing, source code is any collection of computer instructions written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer.

The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.

Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or to understand how it works.

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page; for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used instand alone graphical applications.

### **3.6 CONCLUSION**

The webpage formation and the process of running source code behind the HTML page have been described.

## **CHAPTER 4**

### **PORT FORWARDING**

#### **4.1 INTRODUCTION**

Firewall is a security system that controls incoming and outgoing network traffic in order to access the internet system remotely (globally) the firewall has to be removed for two-way traffic. The way of removing it is described below.

#### **4.2 FIREWALL**

In computing, a firewall is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. A firewall establishes a barrier between a trusted, secure internal network and another network (e.g., the Internet) that is assumed not to be secure and trusted. Firewalls exist both as software to run on general purpose hardware and as a hardware appliance. Many hardware-based firewalls also offer other functionality to the internal network they protect, such as acting as a DHCP server for that network.

Many personal computer operating systems include software-based firewalls to protect against threats from the public Internet. Many routers that pass data between networks contain firewall components and, conversely, many firewalls can perform basic routing functions,

#### **4.3 PORT FORWARDING**

In computer networking, port forwarding or port mapping is an application of network address translation (NAT) that redirects a communication request from one address and port number combination to another while the packets are traversing a network gateway, such as a router or firewall. This technique is most commonly used to make services on a host residing on a protected or internal network available to hosts on the opposite side of the

gateway (external network), by remapping the destination IP address and port number of the communication to an internal host.

### **4.3.1 PURPOSE OF PORT FORWARDING**

Port forwarding allows remote computers to connect to a specific computer or service within a private local-area network (LAN). In a typical residential network, nodes obtain Internet access through a cable modem connected to a router. Hosts on the private network are connected to an Ethernet switch or communicate via a wireless LAN. The NAT device's external interface is configured with a public IP address. The computers behind the router, on the other hand, are invisible to hosts on the Internet as they each communicate only with a private IP address.

When configuring port forwarding, the network administrator sets aside one port number on the gateway for the exclusive use of communicating with a service in the private network, located on a specific host. External hosts must know this port number and the address of the gateway to communicate with the network-internal service. Often, the port numbers of well-known Internet services, such as port number 80 for web services (HTTP), are used in port forwarding, so that common Internet services may be implemented on hosts within private networks.

Typical applications include the following:

- Running a public HTTP server within a private LAN
- Permitting Secure Shell access to a host on the private LAN from the Internet
- Permitting FTP access to a host on a private LAN from the Internet
- Running a publicly available game server within a private LAN

### 4.3.2 How to Set Up Port Forwarding on a Router

Port forwarding opens certain ports on our home or small business network, usually blocked from access by our router, to the Internet. Opening specific ports can allow games, servers, Bit Torrent clients, and other applications to work through the usual security of our router that otherwise does not permit connections to these ports.

#### 1. Enter the router's IP address in the address bar of a web browser.

This will open your router's configuration page. For most routers, this will be 192.168.0.1, 192.168.1.1, or 192.168.2.1. However, if you want to figure out the IP, below are the steps

- For Windows: Open the command prompt and enter `ipconfig /all`. The router's IP address is usually the same as the Default Gateway.
- For Mac: Open the terminal and enter `netstat -nr`.
- For Linux: Open the terminal and enter `route`.

#### 2. Enter username and password.

If we have already configured the security settings for our router, enter the username and password we chose then.

#### 3. Finding the Port Forwarding section and enabling the ports.

Each router will be slightly different. Common labels are Port Forwarding, Applications, Gaming and Virtual Servers. If we don't see one of these or something similar, try Advanced Settings and look for a Port Forwarding subsection. Enable the HTTP server and FTP server ports.

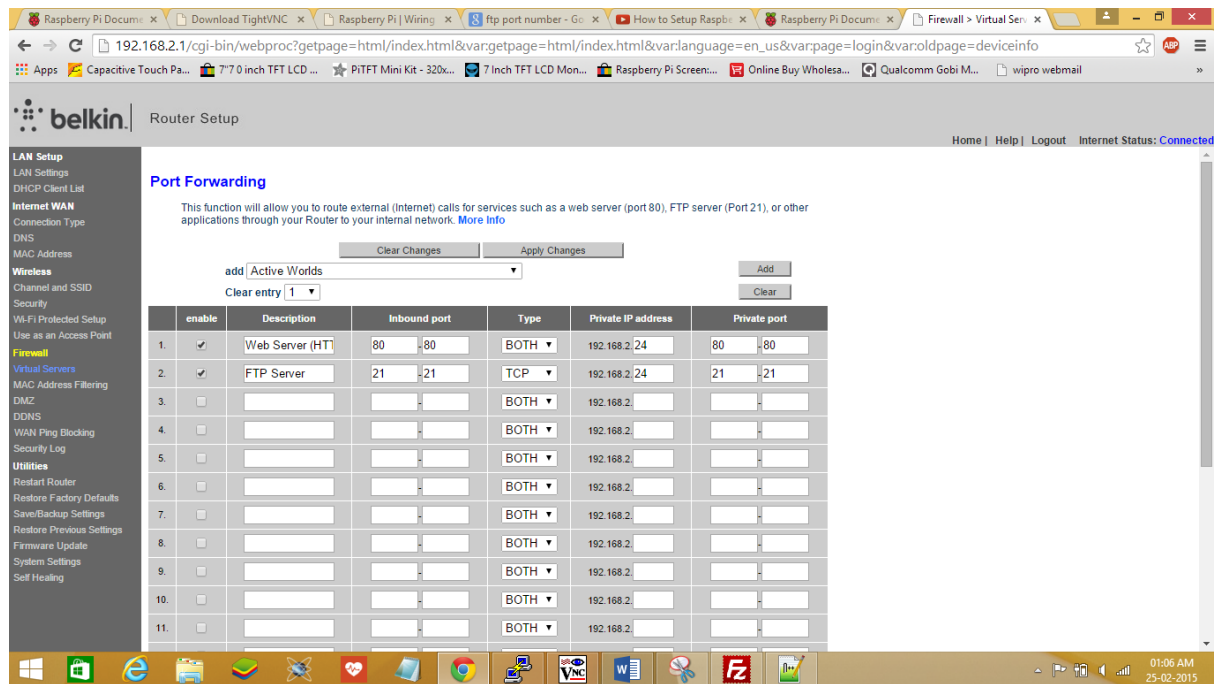


Fig 4.1 Opening ports for port forwarding

## 4.4 CONCLUSION

By enabling ports in the port forwarding section of the router it removes the firewall so that the server in the network of this router can be accessed globally from anywhere in the world over the internet



## CHAPTER 5

### PROCESSING OF INPUTS IN THE RASPBERRY PI

#### 5.1 INTRODUCTION

There should be a gateway between the internet and the hardware that are to be accessed using that internet. There are many gateway devices such as arduino, banana pro, Raspberry Pi etc., this project uses Raspberry Pi as the gateway structure because of the low cost and the compatible size of it.

Initially Raspberry Pi has to be installed with raspbian operating system and configuration is done as per the use and required software has to be included as shown in the chapter ahead.

#### 5.2 CONVERTING INTO A WEBSERVER

In order to have a link between the gateway device and the user using it over internet the gateway device, Raspberry Pi, must be able to be accessed globally. For this purpose we convert our Raspberry Pi into a Web Server to exploit its advantages. There are three steps included in converting the Raspberry Pi as the web server which includes running some commands on the Raspberry Pi as a result of that an html file is formed where the source code of the project has to be included for the execution of the inputs over the web page through internet. This conversion includes installation of two different software on the Raspberry Pi namely apache and PHP where apache is the worldwide used software for the web server conversion and PHP is a script that is used for running a source code over a webpage. The following are the three steps to be followed:

**Step1:** In this step you have to install Apache and PHP in order to set up the server. For this purpose you must follow the below command

**"sudo apt-get install apache2 php5 libapache2-mod-php5"**

**Step2:** Now reboot the entire program using either of the following commands.

**"sudo service apache2 restart" or "sudo /etc/init.d/apache2 restart"**

**Step3:** Finally enter the I.P. address and the screen will show “It Works”.

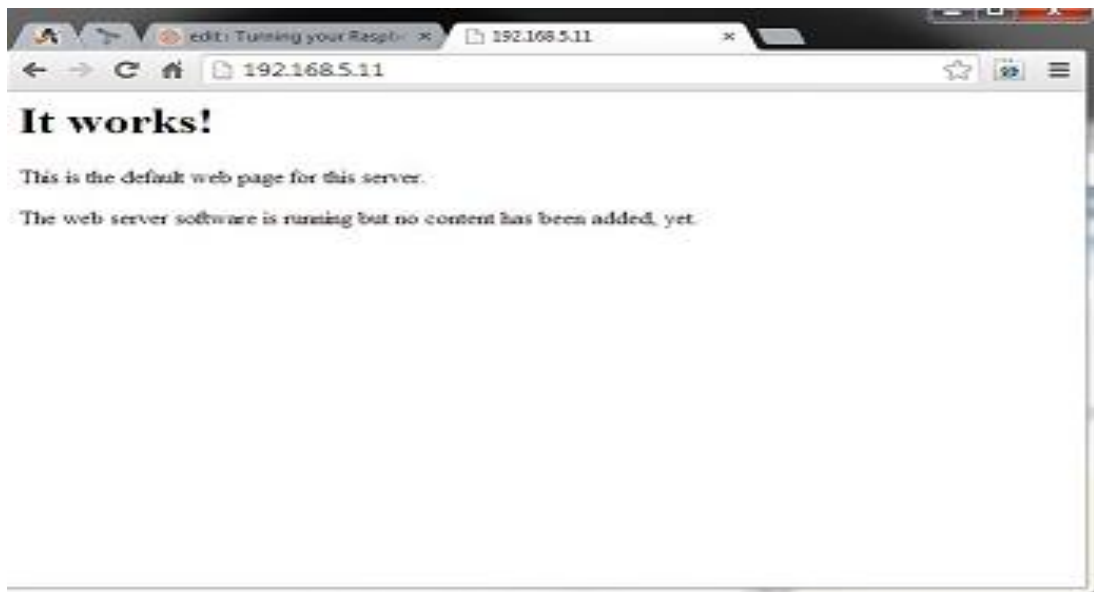


Fig 5.1 Default webpage

## 5.3 ARM PROCESSOR

### Processor / SoC (System on Chip)

The Raspberry Pi has a Broadcom BCM2835 System on Chip module. It has an ARM1176J7F-S processor

The Broadcom SoC used in the Raspberry Pi is equivalent to a chip used in an old Smartphone (Android or Phone). While operating at 700 MHz by default, the Raspberry Pi provides a real world performance roughly equivalent to the 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997-1999, but the GPU, however, provides 1 Gpixel/s, 1.5 Gtexel/s or 24 GFLOPS of general purpose compute and

the graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox of 2001. The Raspberry Pi chip operating at 700 MHz by default, will not become hot enough to need a heat sink or special cooling.

### BCM2835: CPU Overview

- ARM11J6JZF-S (ARM11 Family)
- ARMv6 Architecture
- Single Core
- 32-Bit RISC
- 700 MHz Clock Rate
- 8 Pipeline Stages
- Branch Prediction

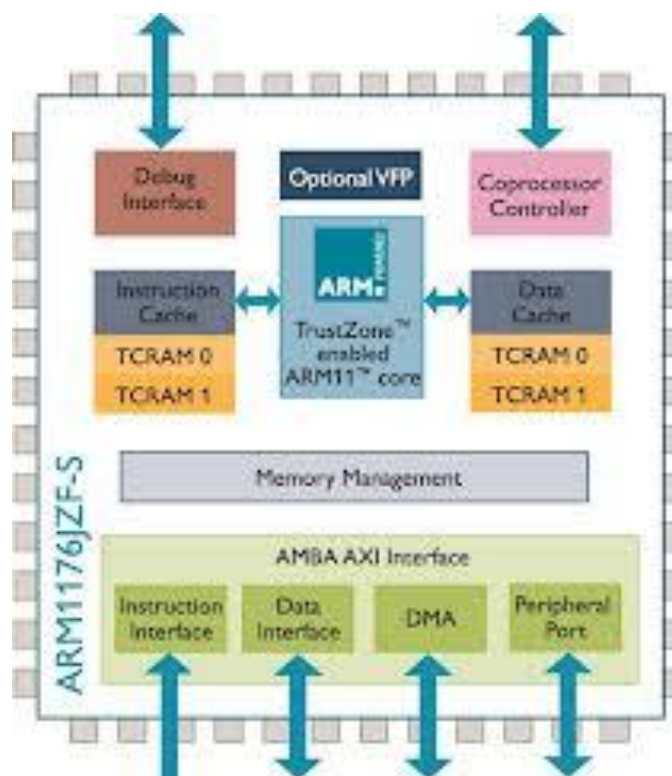


Fig 5.2 Architecture of BCM2835

### 5.4 General Purpose Input Output pins

In order to access the GPIO pins we have to include the library called wiring pi in Raspberry Pi using few commands given in the chapter 7.

GPIO — General Purpose Input Output General-purpose input/output (GPIO) is a generic pin on an integrated circuit whose behavior, including whether it is an input or output pin, can be controlled by the user at run time.

GPIO pins have no special purpose defined, and go unused by default. The idea is that sometimes the system designer building a full system that uses the chip might find it useful to have a handful of additional digital control lines, and having this available on the chip can save the hassle of having to arrange additional circuitry to provide them.

GPIO capabilities may include:

- GPIO pins can be configured to be input or output
- GPIO pins can be enabled/disabled
- Input values are readable (typically high, low)
- Output values are writable/readable
- Input values can often be used as IRQs (typically for wakeup events)

The production Raspberry Pi board has a 26-pin 2.54 mm (100 mil) expansion header, marked as PI, arranged in a 2x13 strip. They provide 8 GPIO pins plus access to PC, SPI, UART), as well as +3.3 V, +5 V and GND supply lines. Pin one is the pin in the first column and on the bottom row.

Raspberry Pi B+ J8 Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Fig 5.3 GPIO Pin Layout for Raspberry Pi B+

Using this gpio pins hardware are connected such as sensors, LEDS, RF module, level shifter and Camera module is connected to the provided on the Raspberry Pi.

## 5.4 CONCLUSION

The description of the gateway system that is Raspberry Pi is been described with proper software's used for the purpose of accessing the given hardware.

## **CHAPTER 6**

### **COMMUNICATION**

#### **6.1 INTRODUCTION**

Communication has also a very important role to play in the project, as we need user and central system to communicate. We need an efficient communication system.

This communication can be wired or wireless.

- Wired Communication.
- Wireless Communication

#### **6.2 COMMUNICATION SYSTEM**

Modulate the RF carrier to be sent to the receiver where RF signal received is demodulated and decoded to get back the commands.

Instead of an encoder and decoder pair in this prototype project we used a microcontroller in transmitting end to receive the data from UART Bridge and reconnect to an RF transmitter module of carrier frequency 433MHz whose sister circuit the super heterodyne RF receiver is connected to microcontroller of the Robot.

#### **6.3 USART**

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. Here we use USART of the microcontroller on the robot to receive the commands. In case of wired communication we use another microcontroller to communicate with the UART device using USART and then RF module comes into the picture where receiver still uses USART communication to receive data from RF receiver.

## 6.4 MODES OF OPERATION OF USART

The USART of the AVR can be operated in three modes, namely

- Asynchronous Normal Mode
- Asynchronous Double Speed Mode
- Synchronous Mode

Each of these modes are described below.

### ASYNCHRONOUS NORMAL MODE

In this mode of communication, the data is transmitted/received asynchronously, i.e. we do not need (and use) the clock pulses, as well as the XCK pin. The data is transferred at the BAUD rate, we set in the UBBR register. This is similar to the UART operation.

### ASYNCHRONOUS DOUBLE SPEED MODE

This is the higher speed mode for asynchronous communication. In this mode also we set the baud rates and other initializations similar to Normal Mode. The difference is that data is transferred at double the baud we set in the UBBR Register. Setting the U2X bit in UCSRA register can double the transfer rate. Setting this bit has effect only for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however, that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

## **SYNCHRONOUS MODE**

This is the USART operation of AVR. When Synchronous Mode is used (UMSEL = 1 in UCSRC register), the XCK pin will be used as either clock input (Slave) or clock output (Master).

## **6.6 WIRED COMMUNICATION**

Since it is our intention to work with USB to UART Bridge, we will be using Asynchronous Transmission Mode.

Before we transmit the information, we need to set the baud rate, clock frequency, number of data bits and number of parity, start and stop bits to be used same as that of the parameters set in Matlab. Start and/or stop bits are set for every set of data because USART is working on asynchronous mode.

The moment data is sent through the USB COM port, microcontroller stores the information in USART receive buffer. In the case where communication is through wired channel (physical electrical connection), the stored information is retrieved through the program written in microcontroller and robot moves accordingly. Each time transmission is done, the buffer gets overwritten but when no transmission prevails, the previous value is retained. Hence each actuation based data transmission is succeeded by the stop data.

## **6.7 WIRELESS COMMUNICATION**

### **6.7.1 WIRELESS TRANSMISSION**

In case of wireless communication, transmitting pin of USB to UART Bridge is physically connected to USART receiver pin of microcontroller whose transmit pin is connected to RF transmitter which generates RF equivalent signal. This analogue signal is transmitted isotropically all around RF transmitter within the range.



### **6.7.2 WIRELESS RECEPTION**

In case of wireless communication, data pin of receiver is connected to receiver pin of robot's microcontroller. The demodulated data from the RF receiver is stored in the USART buffer of the microcontroller on robot and is used for actuation of the robot through the program in flash memory and robot moves accordingly.

### **6.8 CONCLUSION**

Communication has very important role to play in this project, as the robot works only on the basis of commands communicated by the central system. The wired and wireless processes of communication of central system and robot on the arena have been discussed.

## CHAPTER 7

### HARDWARE DESCRIPTION

#### 7.1 INTRODUCTION

As modern embedded systems grow in complexity component-based development is an increasingly attractive approach utilized to make the development of such systems simpler and less error prone. However, in embedded system domain component-based approach to software development is seldom used in practice, and is mostly explored in component models used in the research context.

Amongst other things, one of the problems that component-based development for embedded systems must address is interaction of a software system with the environment; the physical world that the system is embedded. This interaction is done using hardware devices, such as sensors and many devices depending on the application like camera and actuators. A simple example of an embedded system is a temperature regulation system, which keeps a constant temperature in a room, cooling or heating the air in it. Such a system must have at least one temperature sensor, and two actuators : one for starting the heating process and one for starting the cooling process. So, even if complexity of software in such a system is very low, its behavior is highly dependent on the communication with hardware devices, and behavior of the devices themselves.

Communication between software and hardware devices can be as simple as writing a value to a hardware pin or port of the device that the system is deployed on, or as complex as an invocation of a service on a remote device. In all cases, this interaction with the environment implies dependencies of software components on the hardware, middleware used to communicate with the environment. Same environment and a combination of hardware and middleware also affect the behavior of an embedded system. As reusability and analyzability of software components and component based systems highly rely on such dependencies and effects on behavior, failure to adequately express they can hinder the use of

component-based approach in the embedded system domain.

Now we have hardware components for connecting with the Raspberry Pi. The description of each component used will be given below.

## 7.2 DOT MATRIX

A dot-matrix display is a display device used to display information on machines, clocks, railway departure indicators and many other devices requiring a simple display device of limited resolution.

The display consists of a dot matrix of lights or mechanical indicators arranged in a rectangular configuration such that by switching on or off selected lights, text or graphics can be displayed. A dot matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced.



Fig 7.1 dot matrix

## 7.3 TEMPERATURE SENSOR

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  and is accurate to  $\pm 0.5^{\circ}\text{C}$  over the range of  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ . In addition, the DS18B20 can derive power directly from the data

line (“parasite power”), eliminating the need for an external power supply. Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

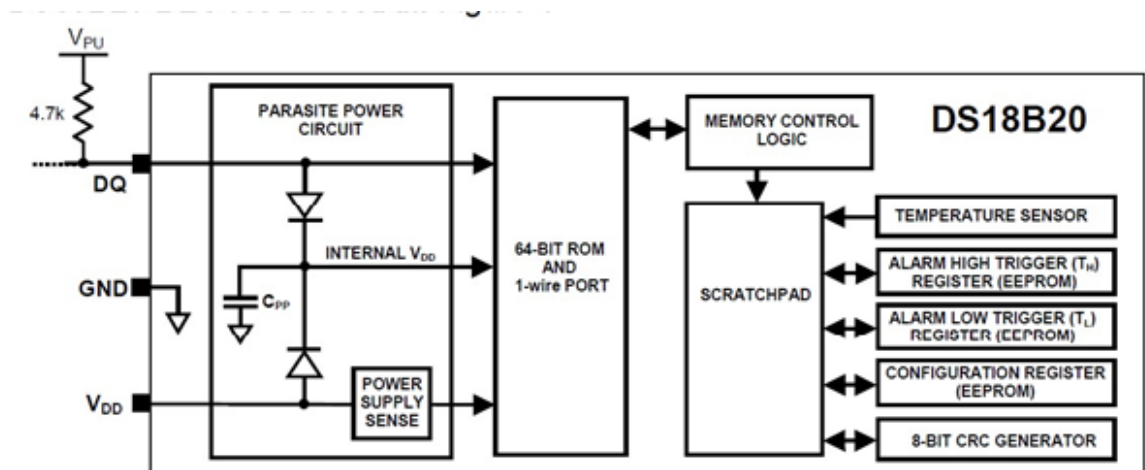


Fig 7.2 Block diagram of temperature sensor

### 7.3.1 OPERATION—MEASURING TEMPERATURE

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue “read time slots” (see the 1-Wire Bus System section) after the Convert T command and the DS18B20 will

respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the Powering the DS18B20 section. The DS18B20 output temperature data is calibrated in degrees Celsius; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 2). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers  $S = 0$  and for negative numbers  $S = 1$ . If the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C *	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FF6Fh
-55°C	1111 1100 1001 0000	FC90h

Table 7.1 Sample digital outputs of temperature sensor

\*The power on reset register value is +85°C.

## 7.4 RASPBERRY PI CAMERA

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge.

The camera module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

The camera works with all models of Raspberry Pi 1 and 2. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi camera Python library.

### 7.4.1 INSTALLATION OF RASPBERRY PI CAMERA

1. Open up your Raspberry Pi Camera module. Be aware that the camera can be damaged by static electricity. Before removing the camera from its grey anti-static bag, make sure you have discharged yourself by touching an earthed object
2. Install the Raspberry Pi Camera module by inserting the cable into the Raspberry Pi. The cable slots into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port.

Boot up your Raspberry Pi.

From the prompt, run "**sudo raspi-config**". If the "camera" option is not listed, you will need to run a few commands to update your Raspberry Pi.

Run "**sudo apt-get update**" and "**sudo apt-get upgrade**"



Fig 7.3 configuration for camera module

### 7.4.2 How to take a photo with your Raspberry Pi camera module

1. "**raspistill**" is a command line application that allows you to capture images with your camera module. Below is an example of this command in use.
2. To capture an image in jpeg format, type "**raspistill -o image.jpg**" at the prompt, where "image" is the name of your image

## 7.5 RF MODULE

The RF module, as the name suggests, operates at Radio Frequency. The corresponding frequency range varies between 30 kHz & 300 GHz. In this RF system, the digital data is represented as variations in the amplitude of carrier wave. This kind of modulation is known as Amplitude Shift Keying (ASK).

Transmission through RF is better than IR (infrared) because of many reasons. Firstly, signals through RF can travel through larger distances making it suitable for long range applications. Also, while IR mostly operates in line-of-sight mode, RF signals can travel even

when there is an obstruction between transmitter & receiver. RF transmission is stronger and reliable than IR transmission. RF communication uses a specific frequency unlike IR signals which are affected by other IR emitting sources.

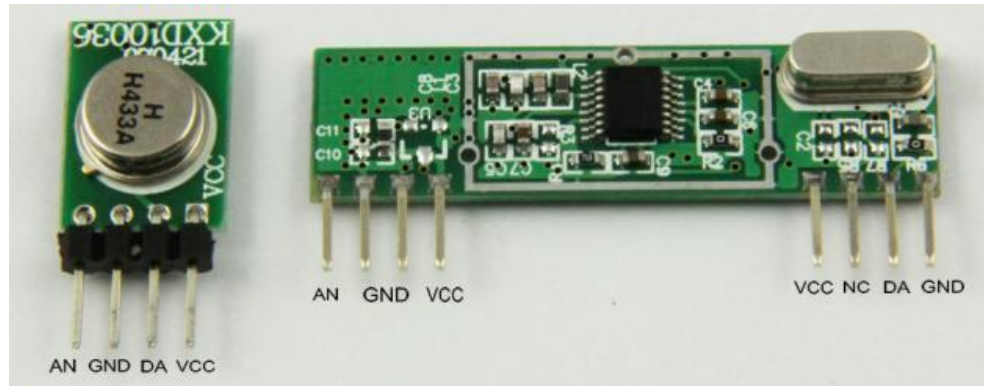


Figure 7.4 433MHz Super heterodyne 3400 RF Transmitter and Receiver

This RF module comprises of an RF Transmitter and an RF Receiver. The transmitter/receiver (Tx/Rx) pair operates at a frequency of 315/434 MHz. An RF transmitter receives serial data and transmits it wirelessly as a radio frequency signal through its antenna connected at pin4. The transmission occurs at the rate of 1Kbps - 10Kbps. The transmitted data is received by an RF receiver operating at the same frequency as that of the transmitter.

### Transmitter Specifications:

- Working voltage: 3V~12V
- Working current: max ≤ 40mA (12V), min ≤ 9mA (3V)
- Modulation mode: ASK/OOK
- Working frequency: 315MHz - 433.92MHz, customized frequency is available.
- Transmission power: 25mW (315MHz at 12V)
- Frequency error: +150kHz (max)
- Baud Rate: ≤ 10Kbps
- Aerial Length: 24cm (315MHz), 18cm (433.92MHz)



### Receiver Specifications:

- Working voltage: 5.0VDC  $\pm 0.5V$
- Working current:  $\leq 2.5mA$  (5.0VDC)
- Bandwidth: 2MHz (315MHz, having result from testing at lowing the sensitivity 3dBm)
- Sensitivity: excel  $-105dBm$  ( $50\Omega$ )
- Output signal: TTL electric level signal entire transmit

## 7.6 MICROCONTROLLER

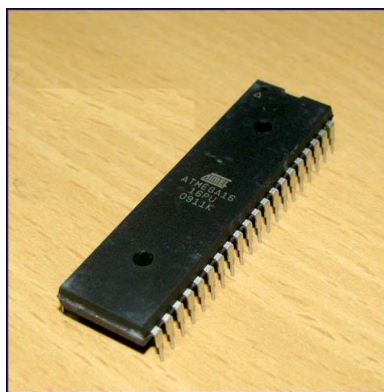


Figure 7.5 Microcontroller – Atmega16

Microcontrollers are "special purpose Computers". Any device that measures, stores, controls, calculates, or displays information is a candidate for putting a microcontroller inside. The microcontroller includes a CPU, RAM, ROM, I/O ports, and timers like a standard computer. Microcontrollers have become common in many areas, and can be found in home appliances, computer equipment, and instrumentation. They are often used in automobiles, and have many industrial uses as well, and have become a central part of industrial robotics. Because they are usually used to control a single process and execute simple instructions, microcontrollers do not require significant processing power. Microcontrollers are hidden inside a surprising number of products these days. If your microwave oven has an LED or LCD screen and a keypad, it contains a microcontroller

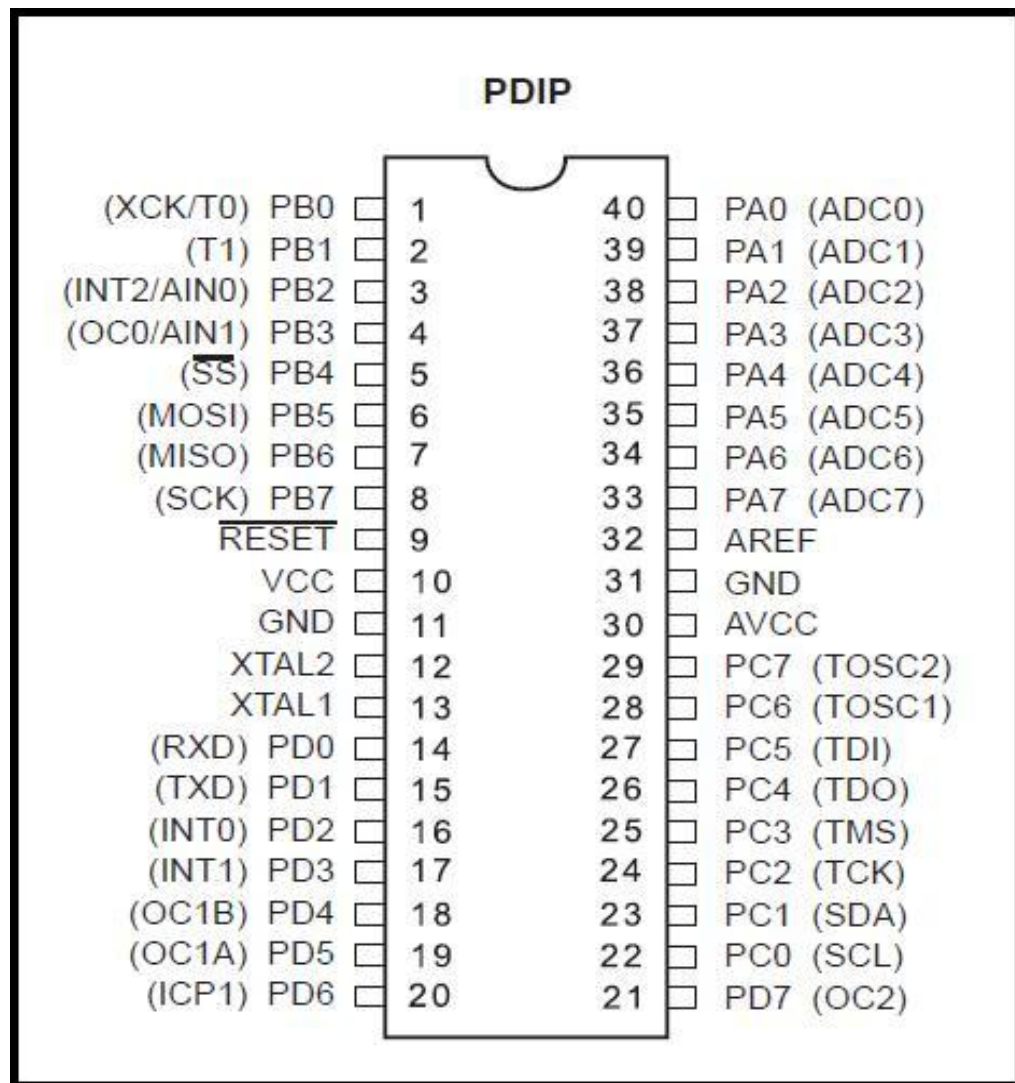


Figure 7.6 Pin description of ATmega16

## Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions—Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier

- Non-volatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    - Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits.In-System Programming by On-chip Boot Program True Read-While-Write Operation
  - 512 Bytes EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
    - Real Time Counter with Separate Oscillator
    - Four PWM Channels
    - 8-channel, 10-bit ADC

8 Single-ended Channels

7 Differential Channels in TQFP Package Only

2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

- 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
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1  $\mu$ A

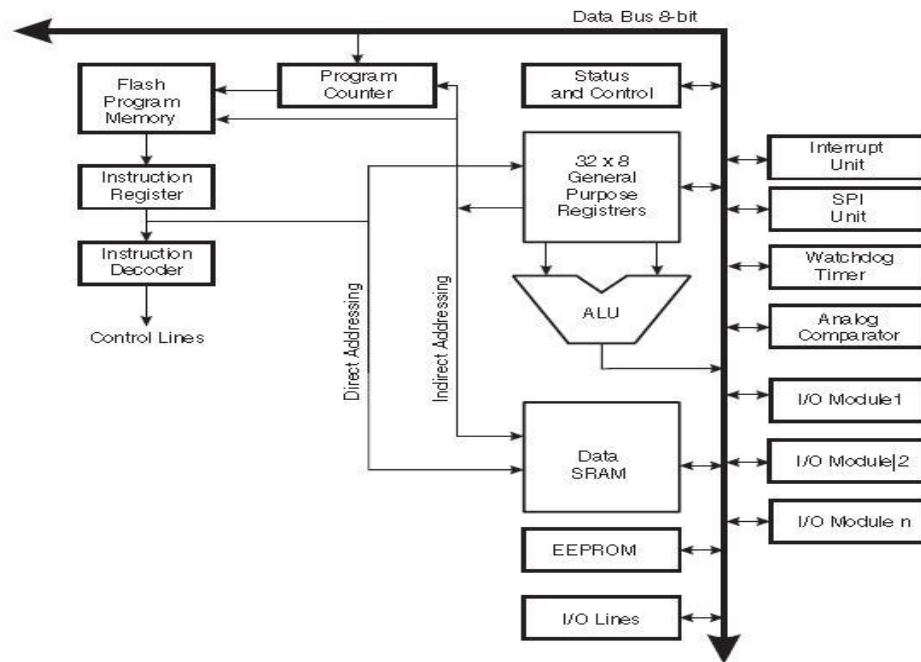


Figure 7.7 AVR CPU architecture

## CISC

Pronounced sisk, and stands for Complex Instruction Set Computer. Most PC's use CPU based on this architecture. For instance Intel and AMD CPU's are based on CISC architectures. Typically CISC chips have a large amount of different and complex instructions. The philosophy behind it is that hardware is always faster than software, therefore one should make a powerful instruction set, which provides programmers with assembly instructions to do a lot with short programs.

## RISC

Pronounced risk, and stands for Reduced Instruction Set Computer. RISC chips evolved around the mid-1980 as a reaction at CISC chips. The philosophy behind it is that almost no one uses complex assembly language instructions as used by CISC, and people mostly use compilers which never use complex instructions. Apple for instance uses RISC chips. Therefore fewer, simpler and faster instructions would be better, than the large, complex and slower CISC instructions. However, more instructions are needed to accomplish a task.

Another advantage of RISC is that - in theory - because of the more simple instructions, RISC chips require fewer transistors, which makes them easier to design and cheaper to produce. Finally, it's easier to write powerful optimized compilers, since fewer instructions exist.

## **RISC Vs CISC**

There is still considerable controversy among experts about which architecture is better. Some say that RISC is cheaper and faster and therefore the architecture of the future. Others note that by making the hardware simpler, RISC puts a greater burden on the software. Software needs to become more complex. Software developers need to write more lines for the same tasks.

Therefore they argue that RISC is not the architecture of the future, since conventional CISC chips are becoming faster and cheaper anyway. RISC has now existed more than 10 years and hasn't been able to kick CISC out of the market. If we forget about the embedded market and mainly look at the market for PC's, workstations and servers I guess a least 75% of the processors are based on the CISC architecture. Most of them the x86 standard (Intel, AMD, etc.), but even in the mainframe territory CISC is dominant via the IBM/390 chip.

Looks like CISC is here to stay ... Is RISC than really not better? The answer isn't quite that simple. RISC and CISC architectures are becoming more and more alike. Many of today's RISC chips support just as many instructions as yesterday's CISC chips. The PowerPC 601, for example, supports more instructions than the Pentium. Yet the 601 is considered a RISC chip, while the Pentium is definitely CISC

### 7.6.1 POWER SUPPLY CIRCUIT

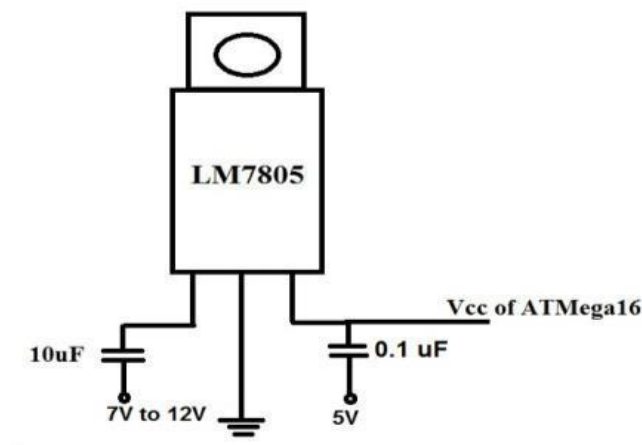


Figure 7.8 Regulated +5V Power supply Circuit

LM 7805 is the heart of regulated power supply circuit.

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, Hi-Fi, and other solid state electronic equipment. Although they are designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents. The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating. Considerable effort was expended to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply. For output voltage other than 5V, 12V and 15V the LM117 series provides an output

voltage range from 1.2V to 57V. Following are the features of LM7805 regulators

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminium TO-3 package

### 7.6.2 RESET CIRCUIT

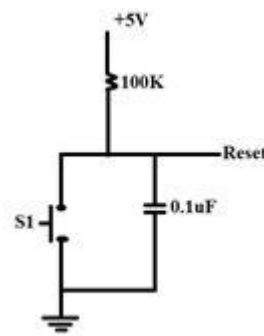


Figure 7.9 Microcontroller reset circuit

A real reset circuit is not necessary in order for a Microcontroller to function in a circuit. In order to set up internal register properly to their initial state after power is applied, there is reset on power-on needed. The only component required to run a Microcontroller, other than those parts that make up the oscillator circuit, is a pull-up resistor connected to the MCLR/ $V_{PP}$  pin. If we omit the pull-up resistor, the Microcontroller will remain in reset (clear) mode on power-up, and will not execute its program. Resistor and pushbutton switch S1 make up an actual reset circuit. When S1 is pressed, it completes a low impedance connection from the MCLR/ $V_{PP}$  pin to ground, forcing the Microcontroller into reset (clear) mode.



### 7.6.3 CRYSTAL OSCILLATOR CIRCUIT

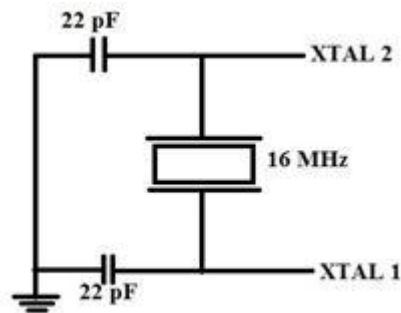


Figure 7.10 Crystal Oscillator Circuit

A crystal oscillator is an electronic circuit that uses the mechanical resonance of a vibrating crystal of piezo electric material to create an electric signal with a very precise frequency. The frequency is commonly used to keep track of time, to provide a stable clock signal for digital integrated circuits. The most common type of piezo electric resonator used is the quartz crystal, so the oscillator circuits designed around them are called Crystal Oscillators.

### 7.7 DC MOTORS



Fig 7.11 DC motors

As a beginner we mostly use DC motors, stepper motor and servo motor will come later. As everybody knows DC motor has two leads. If we apply +ve to one lead and ground to another motor will rotate in one direction, if we reverse the connection the motor will rotate in opposite direction. If we keep both leads open or both leads ground it will not rotate (but some inertia will be there). If we apply +ve voltage to both leads then braking will occur. You can

test this, first without applying any voltage you rotate the shaft of the motor, then apply ground on both lead and try to rotate the shaft. Both will almost remain same, but if we apply both lead +ve voltage(+12V) and try to rotate the shaft, you can feel the difference between the previous one. You have to apply more force to rotate the same rotation in previous connection. So we take this condition as braking, because if we want to stop the motor suddenly then this is the better way which is easily possible. There are methods to brake motor fastly, like shorting two leads, applying negative polarity exists, but we won't use this in robotics. We apply (1,1) condition to break the motor faster (see H-bridge section for more about it). The main things about a DC motor are Voltage rating, current rating, Torque, Speed. Remember Torque is inversely proportional to speed. So we had to get a good speed motor to get good torque because we can operate the good speed motor in slow speed to get good torque. So maximum speed of the motor should be as high as possible.

## 7.8 MOTOR DRIVER: L298N

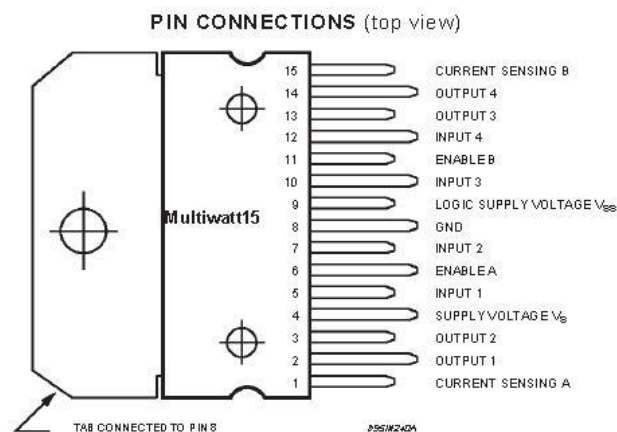


Figure 7.12 Multi-watt Motor driver L298N

### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multi-watt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

## 7.9 LEVEL SHIFTER

- 74HC245 buffer is a high speed buffer from NXP(*philips*). It is used to convert voltage level of different databus in embedded systems.
- Some time we need to connect a 3.3 volt operated device with 5 volt device. For example we have to connect ethernet controller ENC28J60 with P89v51RD2 microcontroller. In this case ethernet controller ENC28J60 is 3.3 volt compatible while P89v51RD2 is 5 volt compatible.
- Here we can't connect data bus(*SPI data bus*) of both directly. We have to convert voltage level.
- For this we can use 74HC245 buffer from NXP.

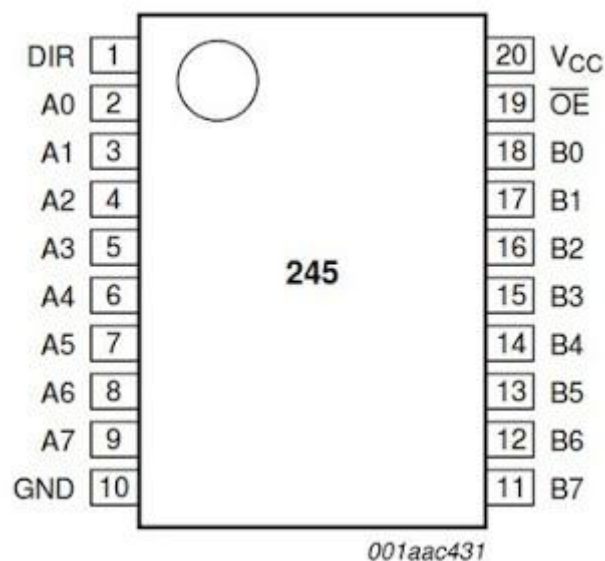


Fig 7.13 Pin diagram of 74HCT245

### 7.9.1 How to use 74HC245 buffer?

- As shown in above figure there are 20 pins in 74HC245 buffer.
- From these 16 pins are for data, one is for VCC, one for ground, one DIR and one OE(bar).
- VCC we can apply -0.5V to +7V. It depends on how much volt we want on data pins.
- DIR and OE(bar) defines direction of data pins. Configuration of DIR and OE pins should be as shown in below figure.

- If you want to connect 3.3V device with 5V controller then you have to apply 3.3V VCC to 74HC245 buffer. So we will get 3.3V output at data pins of both sides.

Input		Input/output	
OE	DIR	An	Bn
L	L	A = B	input
L	H	input	B = A
H	X	Z	Z

H=HIGH, L=LOW, Z= HIGH IMPEDENCE OFF-STATE, X= DON'T CARE

Table 7.2 Functional table of 74HCT245

## 7.10 CONCLUSION

Complete description of hardware used for the project has been discussed and complete description of ATmega 16 micro controller is also studied.

## CHAPTER 8

### SOFTWARES USED

#### 8.1 INTRODUCTION

In this chapter we will get to know about the software we are using in this project. The following is the list of all software that has been used for several purposes in this project.

- RASPBIAN - Operating System in Raspberry Pi.
- PUTTY (Secure Shell), Virtual Network Computing Server and Virtual Network Computing Viewer (For viewing desktop of Raspberry Pi) – Software to access Raspberry Pi remotely.
- Wiring Pi – Software to access GPIO pins on Raspberry Pi.
- Apache - Software to make Raspberry Pi a web server.
- HTML – For client side Programming.
- PHP – Server side Programming.
- Proteus – Simulation software.
- AVR Studio – To Program Atmega16 micro controller.
- AVR Loader – To dump the program in Atmega16 micro controller.

#### 8.2 RASPBIAN

##### 8.2.1 INSTALLATION

Every hardware should be equipped with a software in order to run applications on that. Operating System is a software that is an interface between user and hardware. Raspbian is the OS that we are using in Raspberry Pi which is available in the official website Raspberry Pi.org. Now we need to write this OS into an SD card which is used as an external memory in Raspberry Pi. The OS available is an image file (2014-06-20-wheezy-

raspbian.img) which can be written on to the SD card with the help of a win32 disk imager software.

Insert the SD card into the laptop/pc and run the image writer. Once open, browse and select the downloaded Raspbian image file. Select the correct device that is the drive representing the SD card. If the drive (or device) selected is different from the SD card then the other selected drive will become corrupted so much care should be taken.

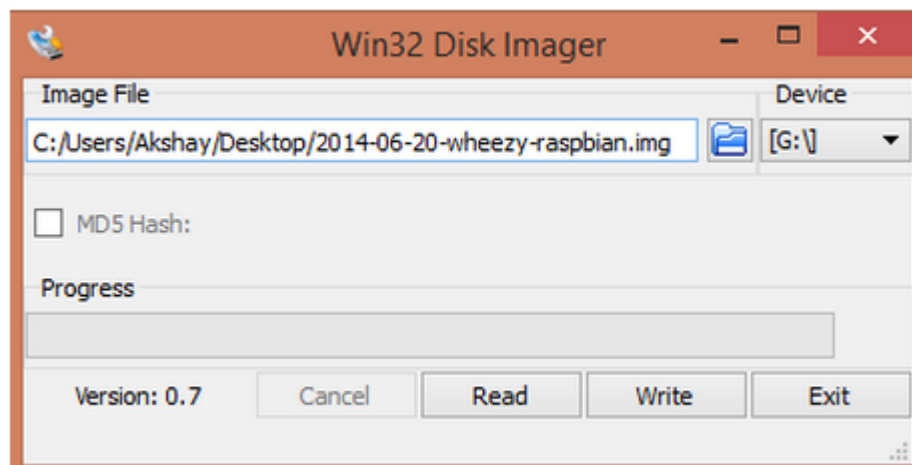


Fig 8.1 Downloading of disk imager

After that, click on the "Write" button in the bottom. In the image shown above SD card (or micro SD) drive is represented by the letter "G:"

## 8.2.2 CONFIGURATION

After booting the Pi, there might be situations when the user credentials like the "username" and password will be asked. Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are

Login: pi (This is username)

Password: raspberry

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the image below.

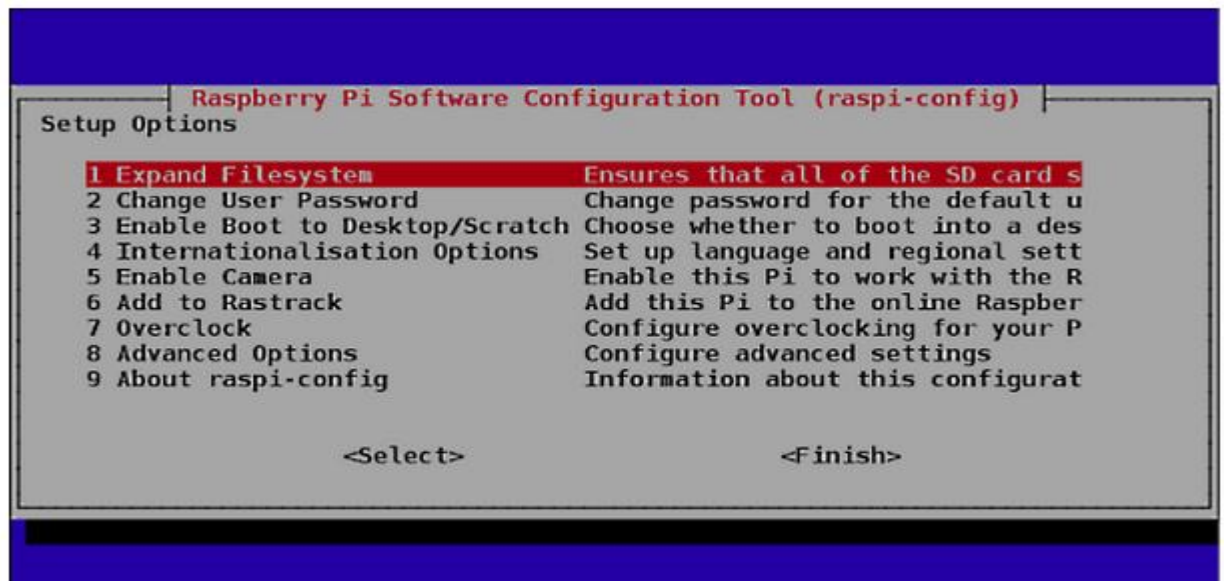


Fig 8.2 Configuration in the Raspberry Pi

If we miss the setup Options screen we can always get it by typing the following command in the terminal.

**sudo raspi-config**

Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, we need to do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

### **The first thing to do:**

Select the first option in the list of the setup options window, that is select the "Expand File system" option and hit the enter key. We do this to make use of all the space present on the SD card as a full partition. All this does is, expand the OS to fit the whole space on the SD card which can then be used as the storage memory for the Pi.

## The Second thing to do:

Select the third option in the list of the setup options window, that is select the "Enable Boot To Desktop/Scratch" option and hit the enter key. It will take you to another window called the "choose boot option" window that looks like the image below.

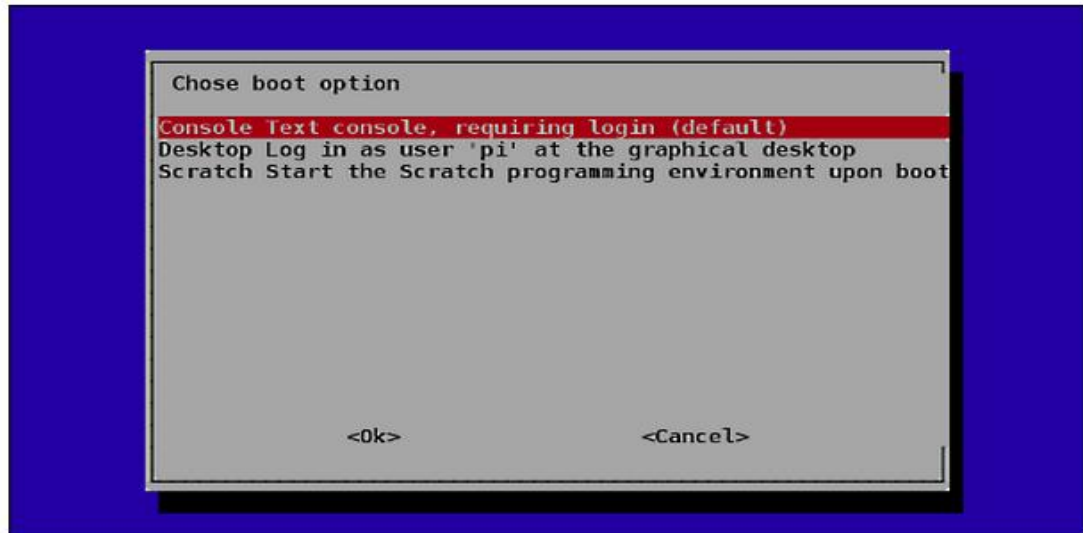


Fig 8.3 Booting operation after the configuration

In the "choose boot option window", select the second option, that is, "Desktop Log in as user 'pi' at the graphical desktop" and hit the enter button. Once done you will be taken back to the "Setup Options" page, if not select the "OK" button at the bottom of this window and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step then the Raspberry Pi boots into a terminal each time with no GUI options.

Once, both the steps are done, select the "finish" button at the bottom of the page and it should reboot automatically. If it doesn't, then use the following command in the terminal to reboot.0

**sudo reboot**



## Updating the firmware

After the reboot from the previous step, if everything went right, then you will end up on the desktop which looks like the image below. Once you are on the desktop, open a terminal and enter the following command to update the firmware of the Pi.

### **`sudo raspi-update`**

This method works on all the different models of Raspberry Pi ( model A, B, B+ and also RPi 2) as Raspbian was made to be supported on all models. However, while installing other software or libraries , the procedure might change a bit while installing depending on the model of the Pi or the version of Raspbian itself.

## 8.3 PUTTY, VNC

### 8.3.1 INTRODUCTION

As said earlier there are two ways of accessing Raspberry Pi. First method is connecting desktop to Raspberry Pi through HDMI port, connecting keyboard, mouse through USB ports. Second method is accessing Pi remotely with another computer or a laptop using some software that enable us to do this.

### 8.3.2 PUTTY

We can remotely gain access to the command line of a Raspberry Pi from another computer on the same network using ssh. **Secure Shell**, or **SSH**, is a cryptographic (encrypted) network protocol for initiating text-based shell session on remote machines in a secure way. This allows a user to run commands on a machine's command prompt without them being physically present near the machine. It also allows a user to establish a secure channel over an insecure network in a client-server architecture, connecting an SSH client application with an SSH server Enter **sudo raspi-config** in the terminal, then

navigate to SSH hit enter and select Enable or disable SSH server. SSH is built into Linux distributions and Mac OS, and a third-party SSH client is available for Windows.

On windows we will need to download an SSH client. The most commonly used one is Putty. Putty is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection.

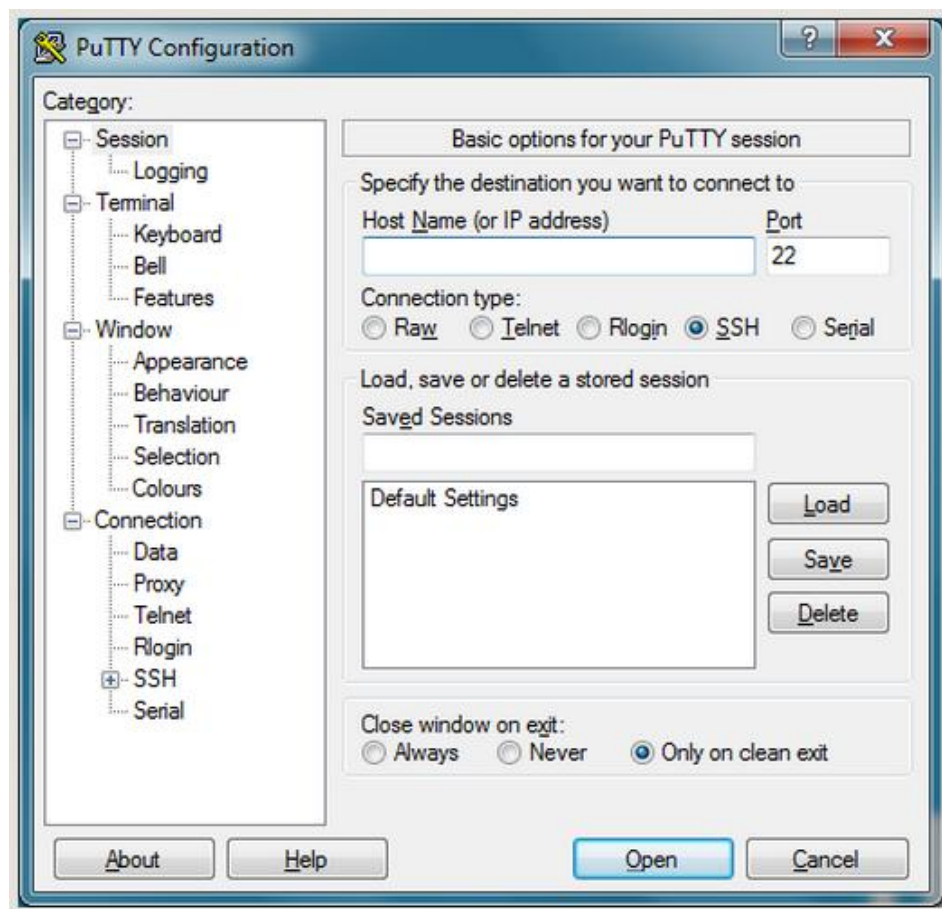


Fig 8.4 Putty Login page

Type the IP address of the Pi into the Host Name field and click the open button. If nothing happens for a while when you click the open button and eventually see a message saying Network Error: connection Timed out it's likely that you've entered the wrong IP address for the Pi. If you don't know the IP address just type Host Name-I in the Raspberry Pi command line. See more methods of finding your IP address. When the connection works

you'll see this security warning (below), you can safely ignore it and click the Yes button. You'll only see this warning the first time when Putty connects to a Pi that it has never Type the IP address of the Pi into the Host Name field and click the open button. If nothing happens for a while when you click the open button and eventually see a message saying Network Error: connection Timed out it's likely that you've entered the wrong IP address for the Pi. If you don't know the IP address just type Host Name-I in the Raspberry Pi command line. See more methods of finding your IP address. When the connection works you'll see this security warning (below), you can safely ignore it and click the Yes button. You'll only see this warning the first time when Putty connects to a Pi that it has never seen before.

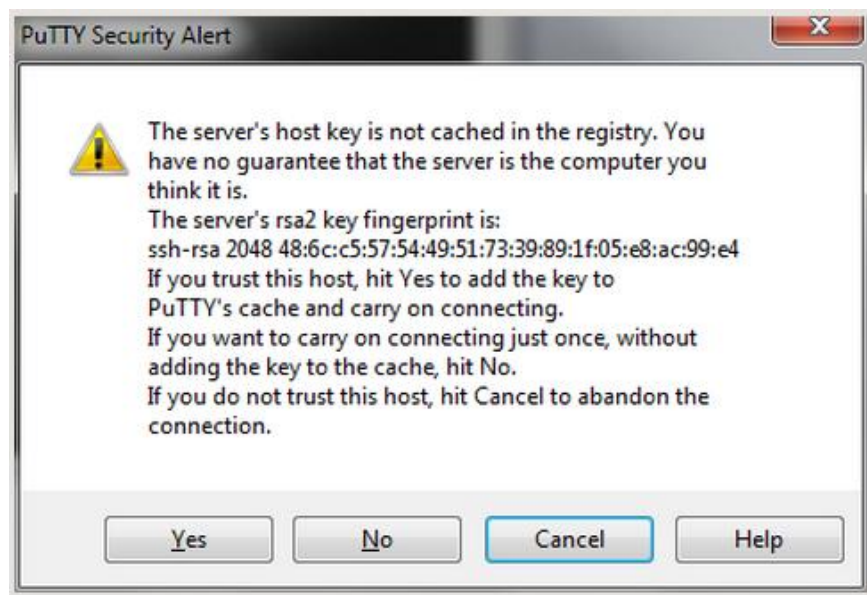
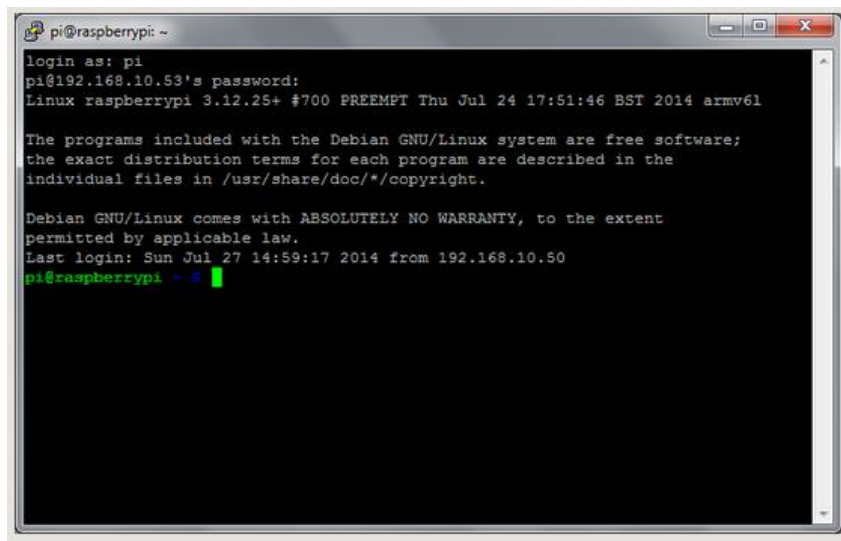


Fig 8.5 Putty security alert

we'll now have the usual login prompt, login with the same username and password as we would use on the Pi itself. The default login for Raspbian is Pi with the password raspberry.

You should now have the Raspberry Pi prompt which will be identical to the one found on the Raspberry Pi itself.

```
pi@raspberrypi ~ $
```



```
pi@raspberrypi: ~
login as: pi
pi@192.168.10.53's password:
Linux raspberrypi 3.12.26+ #700 PREEMPT Thu Jul 24 17:51:46 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 27 14:59:17 2014 from 192.168.10.50
pi@raspberrypi ~$
```

Fig 8.6 Putty page for programming

You can type exit to close the Putty window.

The next time you use Putty look for the Saved Sessions section on the bottom half of the configuration screen. If you use this I recommend switching to the Connection page in the left hand tree and setting the Seconds between keep alives value to 30. Then switch back to the Session page in the tree before you click Save. Using this setting allows you to leave a Putty window open for long periods of time with no activity and the Pi will not time out and disconnect us.

### 8.3.3 VNC

Sometimes it is not convenient to work directly on the Raspberry Pi. Maybe we would like to work on it from another computer by remote control. VNC is a graphical desktop sharing system that allows you to remotely control the desktop interface of one computer from another. It transmits the keyboard and mouse events from the controller, and receives updates to the screen over the network from the remote host.

We will see the desktop of the Raspberry Pi inside a window on your computer. We'll be able to control it as though you were working on the Raspberry Pi itself.

On your Pi (using a monitor or via SSH), install the TightVNC package:

```
sudo apt-get install tightvncserver
```

Next, run TightVNC Server which will prompt us to enter a password and an optional view-only password:

```
tightvncserver
```

Start a VNC server from the terminal. This example starts a session on VNC display zero (:0) with full HD resolution:

```
vncserver :0 -geometry 1920x1080 -depth 24
```

Now, on our computer, we need to install and run the VNC client:

On a Linux machine install the package tightvncviewer:

```
sudo apt-get install tightvncviewer
```

When we install tightvnc on our computer we will find file like tvnviewer.exe .when we double click on it we get a popup like below and we need to enter the IP address of the Raspberry Pi. After entering the IP address of the Raspberry Pi we need to enter the password to view the desktop of Pi on our Laptop or Computer.

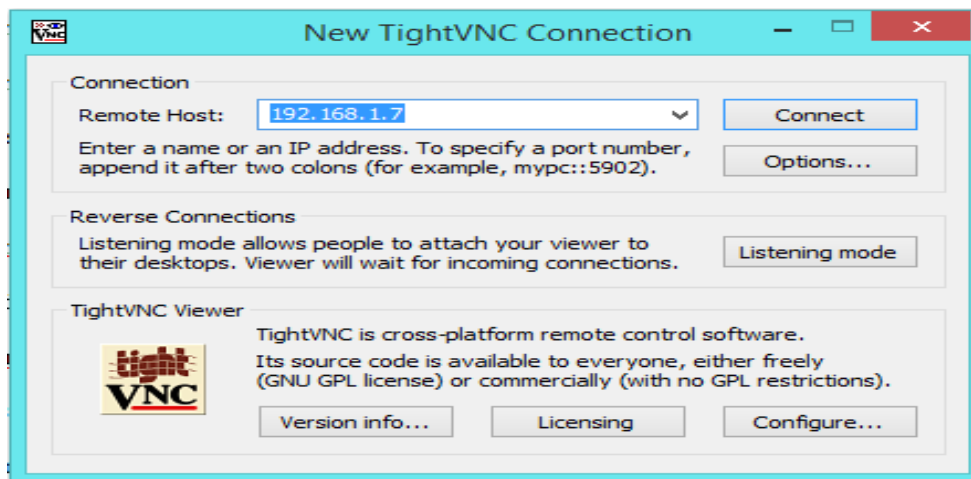


Fig 8.7 VNC login page

## 8.4 WiringPi

### 8.4.1 INTRODUCTION

WiringPi is a GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It's released under the GNU LGPLv3 license and is usable from C and C++ and many other languages with suitable wrappers (See below) It's designed to be familiar to people who have used the Arduino "*wiring*" system<sup>1</sup>

The Raspberry Pi has a 26-pin General Purpose Input/Output (GPIO) connector and this carries a set of signals and buses. There are 8 general purpose digital I/O pins – these can be programmed as either digital outputs or inputs. One of these pins can be designated for PWM output too. Additionally there is a 2-wire I2C interface and a 4-wire SPI interface (with a 2nd select line, making it 5 pins in total) and the serial UART with a further 2 pins.

The Revision 2 Raspberry Pi has an additional 4 GPIO lines on a separate connector which you have to solder onto the board.

The model B+ Raspberry Pi represents 2 years of research, development and testing and now features a single 40-pin GPIO connector with 28 usable GPIO pins.

The I2C, SPI and UART interfaces can also be used as general purpose I/O pins when not being used in their bus modes, giving a grand total of  $8 + 2 + 5 + 2 = 17$  I/O pins on the P1 connector (plus 4 more on the P5 connector on a Revision 2 Pi)

WiringPi includes a command-line utility `gpio` which can be used to program and setup the GPIO pins. You can use this to read and write the pins and even use it to control them from shell scripts.

WiringPi is extendable and modules are provided to extend WiringPi to use analog interface devices on the Gertboard, and to use the popular MCP23x17/MCP23x08 (I2C 7 SPI) GPIO expansion chips, as well as a module that will allow blocks of up to 4 74x595 shift registers to be daisy-chained together for an additional 32-bits worth of output as a single unit. (You can have several blocks of 4 74x595s if needed) One of the extension modules allows you to use an ATmega (e.g. Arduino, or the Gertboard) as more GPIO expansion too – via the Pi's serial port.

Additionally, you can easily write your own expansion modules to integrate your own peripheral devices with Wiring Pi as required.

Wiring Pi supports analog reading and writing, and while there is no native analog hardware on a Pi by default, modules are provided to support the Gert boards analog chips and other A/D and D/A devices can be implemented relatively easily.

## 8.5 INSTALLING WIRINGPI ON RASPBERRY PI

WiringPi is maintained under GIT for ease of change tracking.

If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:

```
sudo apt-get install git-core
```

If you get any errors here, make sure your Pi is up to date with the latest versions of Raspbian:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

To obtain WiringPi using GIT:

```
git clone git://git.drogon.net/wiringPi
```

If you have already used the clone operation for the first time, then

```
cdwiringPi
```

```
git pull origin
```

Will fetch an updated version then you can re-run the build script below.

To build/install there is a new simplified script:

```
cdwiringPi
```

```
./build
```

The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.

Test wiringPi's installation:

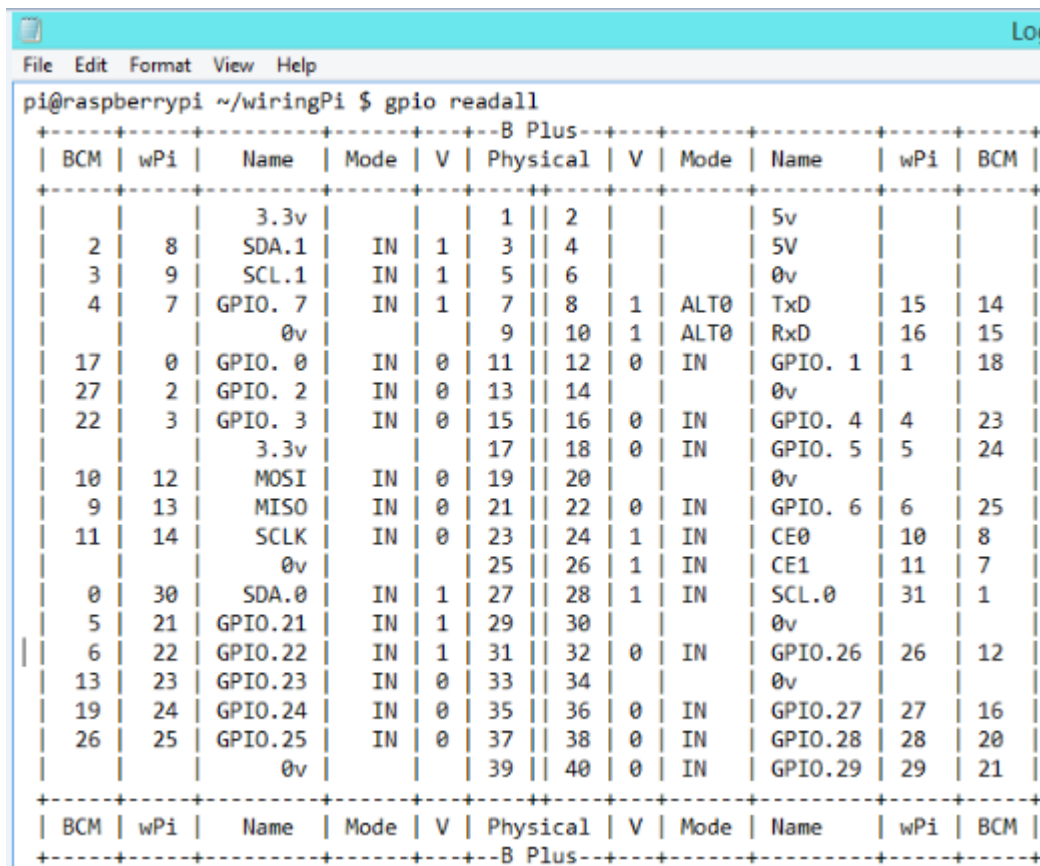
run the gpio command to check the installation:

```
gpio -v
```

```
gpioreadall
```

The above command shows the status of pins whether they are enabled/disabled or low/high.





```

pi@raspberrypi ~/wiringPi $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  2  |  8  |  3.3v    |      |   |  1  |  2  |      |  5v      |     |     | |
|  3  |  9  |  SDA.1   | IN    | 1 |  3  |  4  |      |  5V      |     |     |
|  4  |  7  |  SCL.1   | IN    | 1 |  5  |  6  |      |  0v      |     |     |
|  4  |  7  | GPIO. 7   | IN    | 1 |  7  |  8  |  1  | ALT0     | Tx D | 15  | 14  |
|      |      |  0v      |      |   |  9  | 10  |  1  | ALT0     | Rx D | 16  | 15  |
| 17  |  0  | GPIO. 0   | IN    | 0 | 11  | 12  |  0  | IN       | GPIO. 1 | 1  | 18  |
| 27  |  2  | GPIO. 2   | IN    | 0 | 13  | 14  |      |  0v      |     |     |
| 22  |  3  | GPIO. 3   | IN    | 0 | 15  | 16  |  0  | IN       | GPIO. 4 | 4  | 23  |
|      |      |  3.3v    |      |   | 17  | 18  |  0  | IN       | GPIO. 5 | 5  | 24  |
| 10  | 12  |  MOSI    | IN    | 0 | 19  | 20  |      |  0v      |     |     |
|  9  | 13  |  MISO    | IN    | 0 | 21  | 22  |  0  | IN       | GPIO. 6 | 6  | 25  |
| 11  | 14  |  SCLK    | IN    | 0 | 23  | 24  |  1  | IN       | CE0    | 10  |  8  |
|      |      |  0v      |      |   | 25  | 26  |  1  | IN       | CE1    | 11  |  7  |
|  0  | 30  |  SDA.0   | IN    | 1 | 27  | 28  |  1  | IN       | SCL.0   | 31  |  1  |
|  5  | 21  | GPIO.21   | IN    | 1 | 29  | 30  |      |  0v      |     |     |
|  6  | 22  | GPIO.22   | IN    | 1 | 31  | 32  |  0  | IN       | GPIO.26 | 26  | 12  |
| 13  | 23  | GPIO.23   | IN    | 0 | 33  | 34  |      |  0v      |     |     |
| 19  | 24  | GPIO.24   | IN    | 0 | 35  | 36  |  0  | IN       | GPIO.27 | 27  | 16  |
| 26  | 25  | GPIO.25   | IN    | 0 | 37  | 38  |  0  | IN       | GPIO.28 | 28  | 20  |
|      |      |  0v      |      |   | 39  | 40  |  0  | IN       | GPIO.29 | 29  | 21  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Table 8.1 Status of GPIO pins

## 8.6 APACHE

### 8.6.1 INTRODUCTION

Apache is a Web server, It mean it's a program that listens for server access requests from internet browsers and grants them if permitted. So if you want anyone to be able to access a website on your Raspberry Pi, including yourself, you need to install a Web server.

Apache is the world's most widely used web server software. Originally based on the NCSA HTTP server. The software is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, OSX, Microsoft, Windows, OS/2, TPF, OpenVMS and eComStation. Released under the Apache License, Apache is free and open-source software.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP. Popular authentication modules include `mod_access`, `mod_auth`, `mod_digest`, and `mod_auth_digest`, the successor to `mod_digest`.

Apache is a free, open-source HTTP (Hypertext Transfer Protocol) Web server application. When you type a URL into your Web browser, a Web server somewhere replies by serving up a Web page. Apache is popular for these purposes: Roughly 50 percent of sites are hosted by servers running Apache.

## 8.6.2 INSTALLATION OF APACHE

This is a one-step process. Go to the command line and type:

```
sudo apt-get install apache2 php5 libapache2-mod-php5
```

This prompt accomplishes several things all at once. It installs the latest version of Apache, the server we need to use. It also installs two other packages: PHP and a library that helps Apache work together with PHP.

For a basic HTML site that remains static and doesn't have many features aside from text, you do not need PHP. But if you ever want your site to connect to a database, you'll need a web framework. PHP is a Web framework that adds more functionality to basic HTML websites.

For example, if you wanted to install Word Press on your Raspberry Pi hosted site, you'd need to make sure you could install at least one database.

When Apache is finished installing, restart it with this command to activate the program:

```
sudo service apache2 restart
```

As soon as the Raspberry Pi finishes processing the above command, it instantly generates a basic, working website.

Go to our Web browser and type in our Pi's local address. This will look something like 192.168.X.X. A very basic site should appear, headlined with the phrase, "It works!" This simple index.html page came preinstalled along with Apache.

```
cd /var/www/
```

```
sudo nano index.html
```

Try changing the words around, saving the file, and navigating back to the Pi's local address again to watch your changes take form.

## **8.7 Win AVR**

### **8.7.1 ADVANTAGES OF EMBEDDED-C OVER ASSEMBLY**

Programs written in assembly can execute faster, while programs written in C are easier to develop and maintain. In traditional applications, such as programs run on personal computers and mainframes, C is almost always the first choice. If assembly is used at all, it is restricted to short subroutines that must run with the utmost speed. For every traditional programmer that works in assembly, there are approximately ten that use C.

A key advantage of using a high-level language (such as C, Fortran, or Basic) is that the programmer does not need to understand the architecture of the microprocessor being used; knowledge of the architecture is left to the compiler. For instance, a short C program uses several variables: n, s, result, plus the arrays: x[ ] and y[ ]. All of

these variables must be assigned a "home" in hardware to keep track of their value. Depending on the microprocessor, these storage locations can be the general purpose data registers, locations in the main memory, or special registers dedicated to particular functions. However, the person writing a high-level program knows little or nothing about this memory management; this task has been delegated to the software engineer who wrote the compiler. The problem is, these two people have never met; they only communicate through a set of predefined rules. High-level languages are easier than assembly because you give half the work to someone else. However, they are less efficient because you aren't quite sure how the delegated work is being carried out.

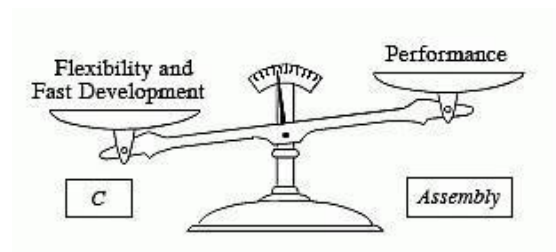


Fig 8.8 Trade-off between Embedded-C and Assembly

### 8.7.2 INTRODUCTION TO WIN AVR

Win AVR is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. It includes the GNU GCC compiler for C and C++. Win AVR is a collection of executable software development tools for the Atmel AVR processor hosted on Windows.

These software development tools include:

- Compilers
- Assembler
- Linker
- Librarian
- File converter

- Other file utilities
- C Library
- Programmer software
- Debugger
- In-Circuit Emulator software
- Editor / IDE
- Many support utilities

## 8.8 PROGRAMMER'S NOTEPAD

Win AVR comes with an editor / IDE called Programmers Notepad. This is an Open Source editor with some IDE capabilities. Because the compiler and associated utilities are all command-line driven, we are free to use whatever editor / IDE you want to provide it can call command-line programs.

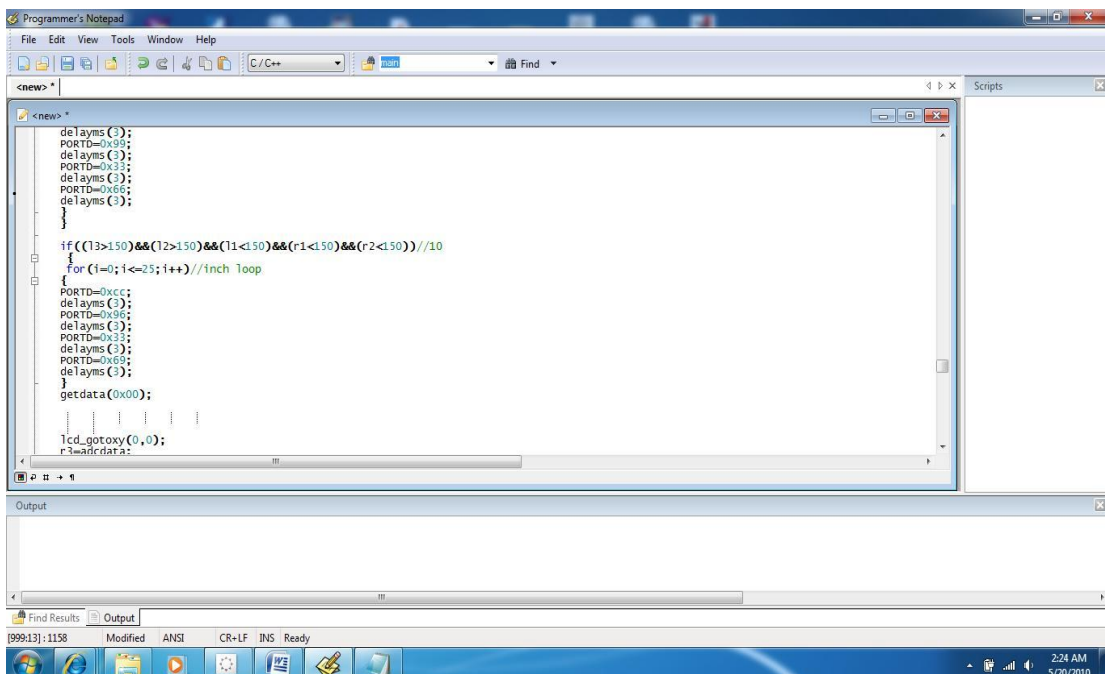


Fig 8.9 Programmer's Notepad

## 8.9 IN-SYSTEM PROGRAMMING

In-system programming (abbreviated ISP) is the ability of some PLDs, micro controllers, and other programmable electronic chips to be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system.

The primary advantage of this feature is that it allows manufacturers of electronic devices to integrate programming and testing into a single production phase, rather than requiring a separate programming stage prior to assembling the system. This may allow manufacturers to program the chips in their own system's production line instead of buying pre-programmed chips from a manufacturer or distributor, making it feasible to apply code or design changes in the middle of a production run.

Typically, chips supporting ISP have internal circuitry to generate any necessary programming voltage from the system's normal supply voltage, and communicate with the programmer via a serial protocol. Other devices usually use proprietary protocols or protocols defined by older standards. In systems complex enough to require moderately large glue logic, designers may implement a JTAG-controlled programming subsystem for non-JTAG devices such as flash memory and microcontrollers, allowing the entire programming and test procedure to be accomplished under the control of a single protocol.

## **8.10 AVR studio 4**

AVR Studio is the platform used to run debug the code written for AVR MCU's, it supports all the AVR microcontrollers and the software can be used in the following way.

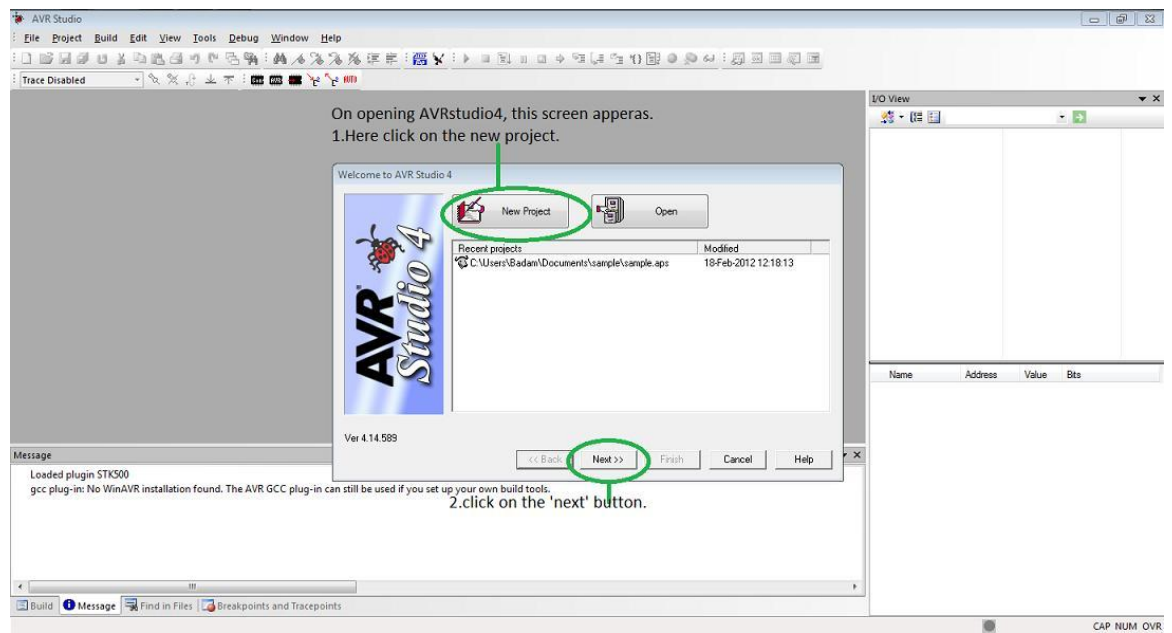


Figure 8.10 AVR studio 4 Creating a new project.

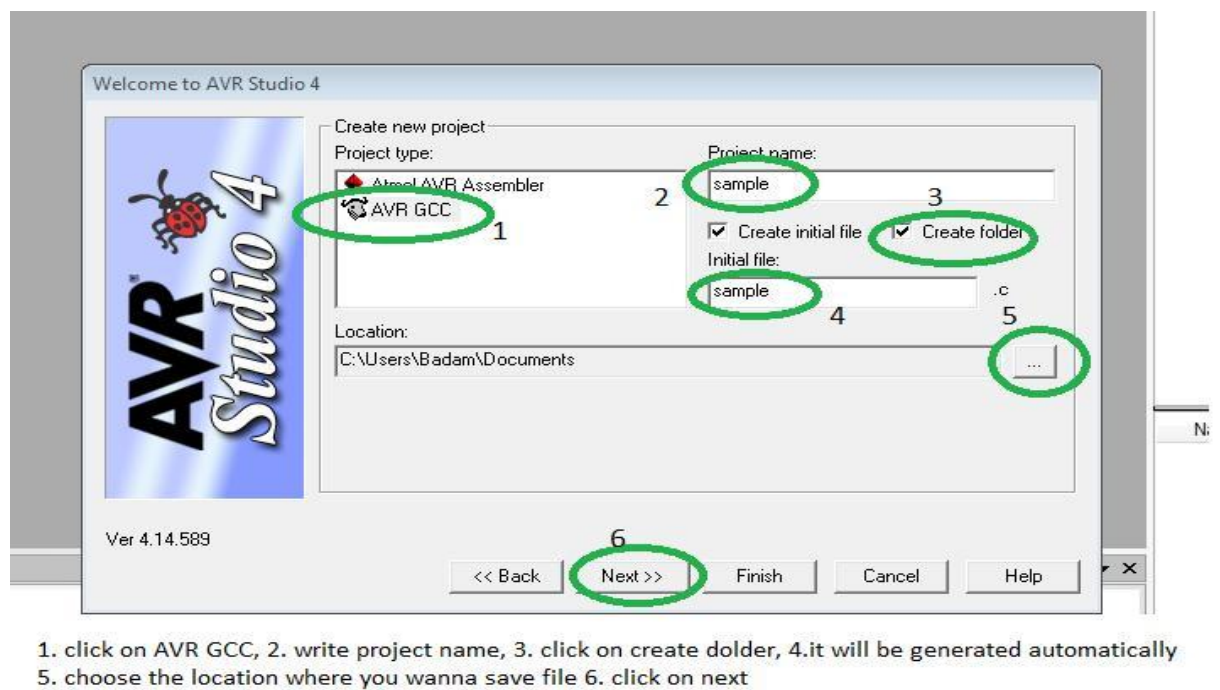
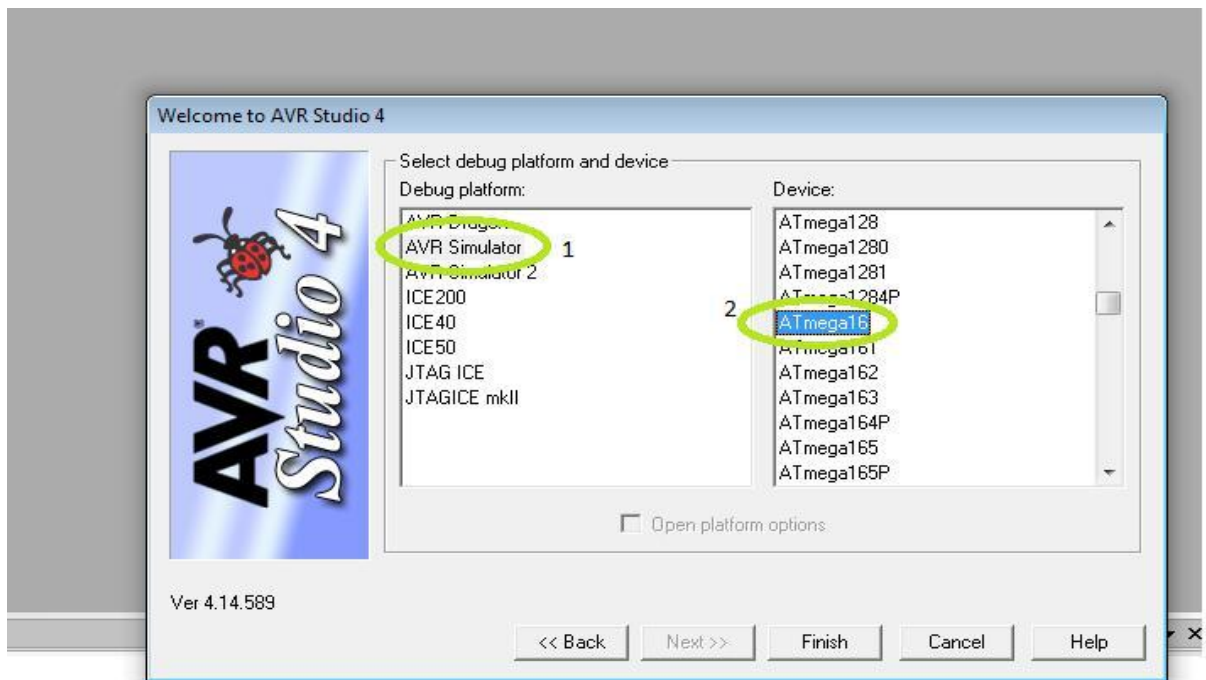


Figure 8.11 selection of compiler



1. select AVR simulator, 2. scroll down and click on ATmega16

Figure 8.12 selection of simulator and device.

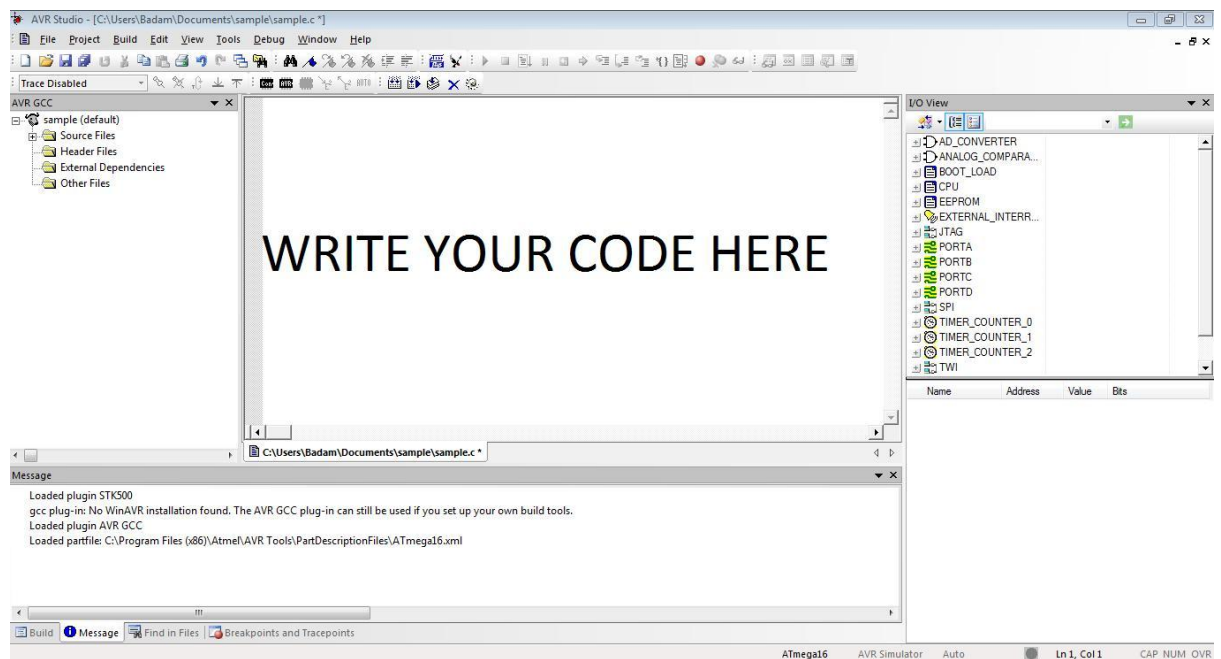


Figure 8.13 Console window to write the code for selected device

To build the code and generate hex file, click on build of build menu on top. The hex code is found in 'default' folder of the project folder.



## 8.11 AVR Loader

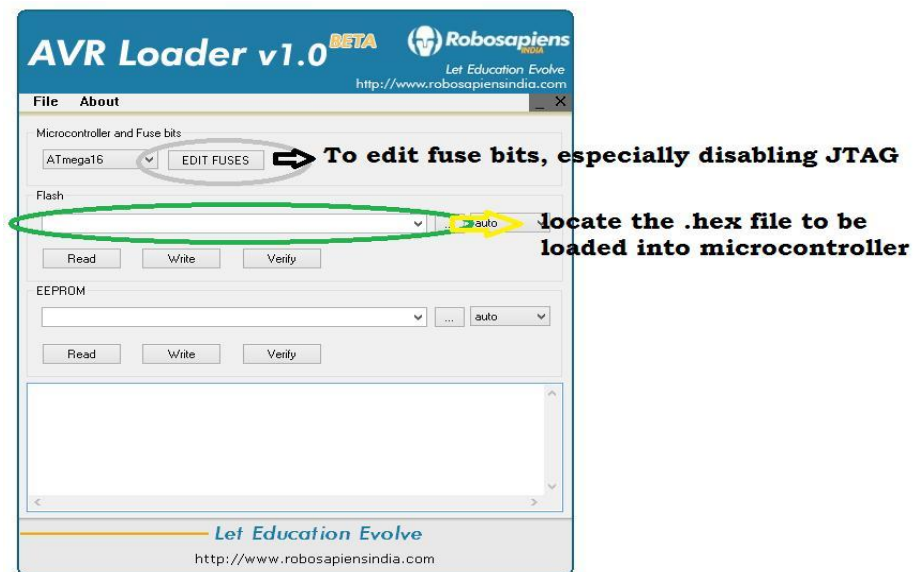


Figure 8.14 AVR Loader for transferring program hex file into microcontroller.

## 8.12 PROTEUS 8

Proteus 8 is a single application with many service modules offering different functionality (schematic capture, PCB layout, etc.). The wrapper that enables all of the various tools to communicate with each other consists of three main parts.

The common database contains information about parts used in the project. A part can contain both a schematic component and a PCB footprint as well both user and system properties. Shared access to this database by all application modules makes possible a huge number of new features, many of which will evolve over the course of the Version 8 lifecycle.

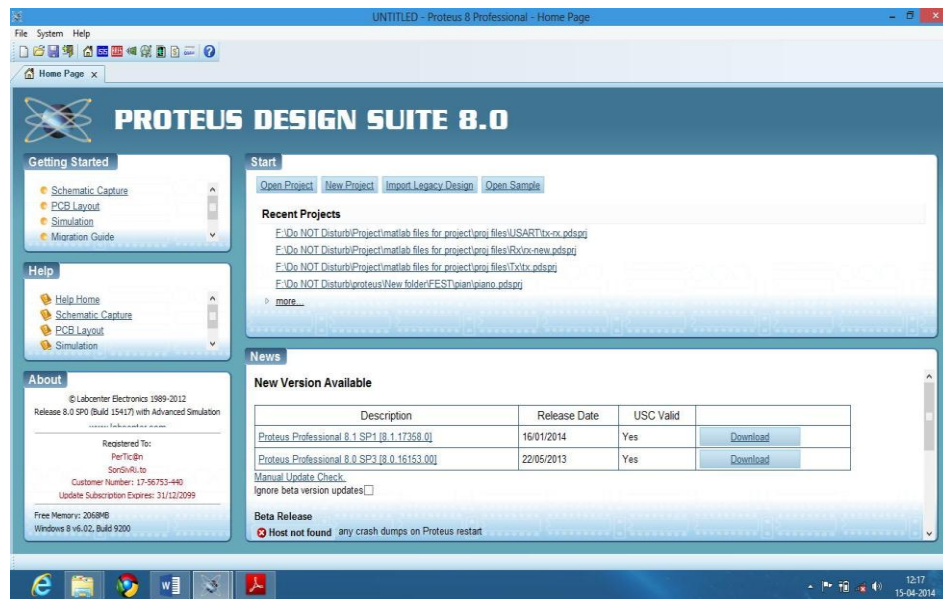


Figure 8.15 (a) PCB design and Circuit simulation software

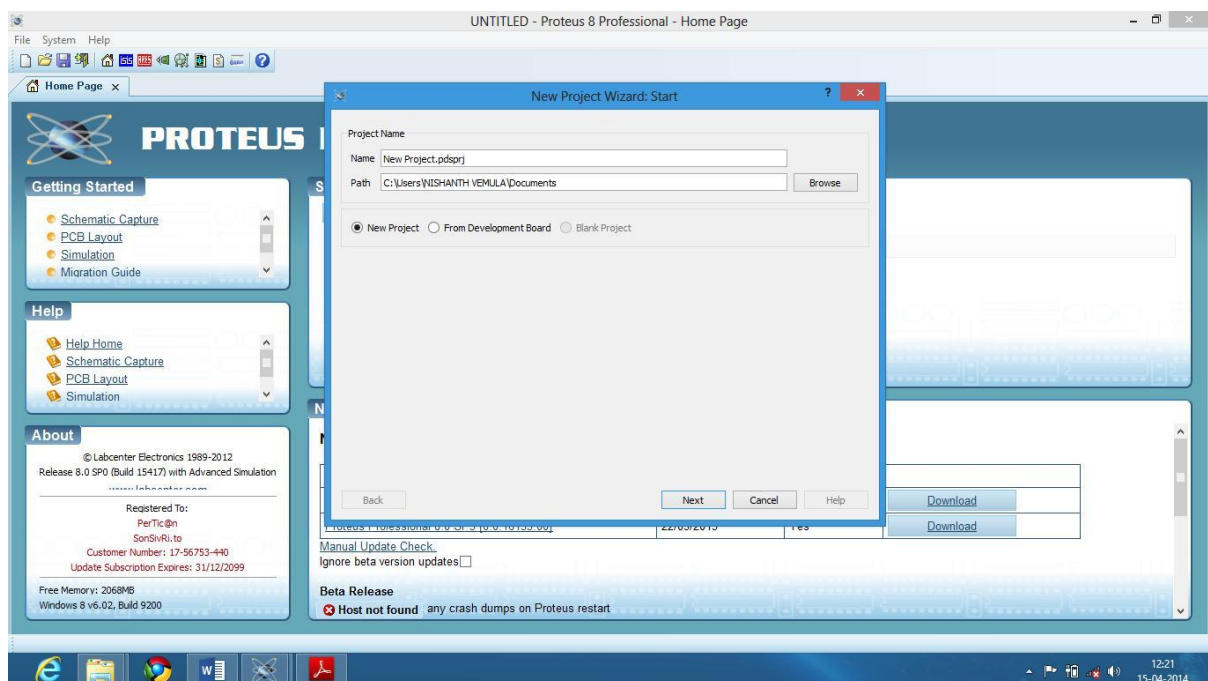


Figure 8.15(b) PCB design and Circuit simulation software creating new project

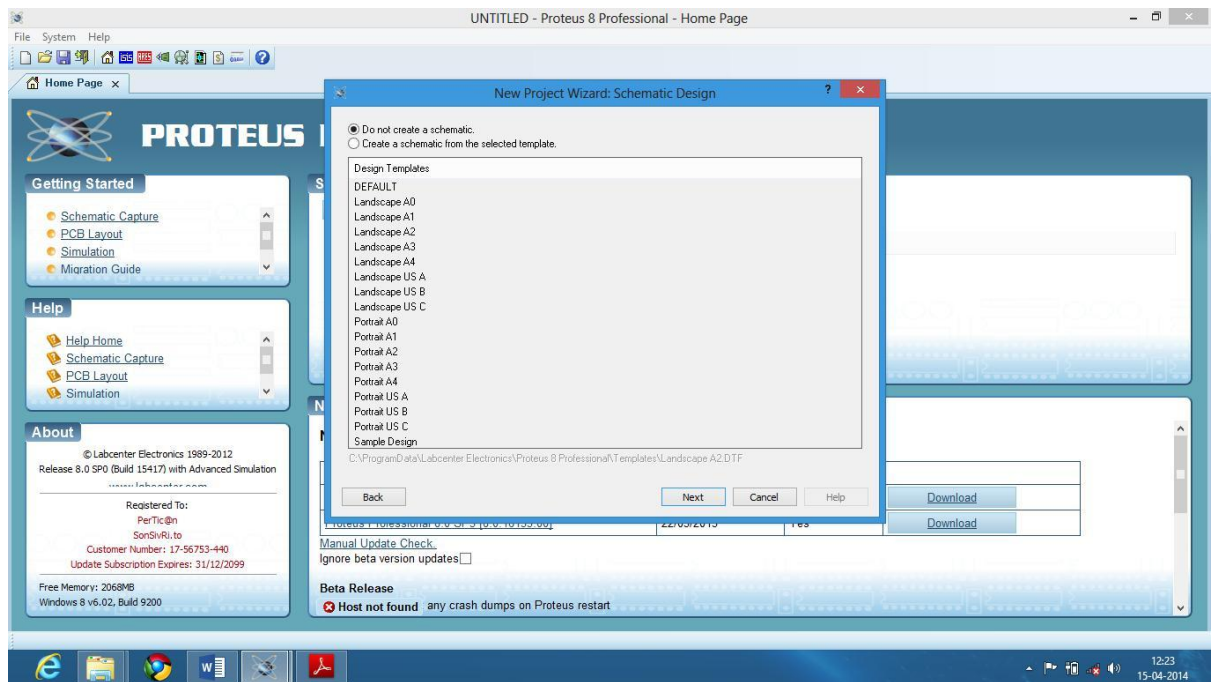


Figure 8.15 (c) PCB design and Circuit simulation software schematic layout panel creation

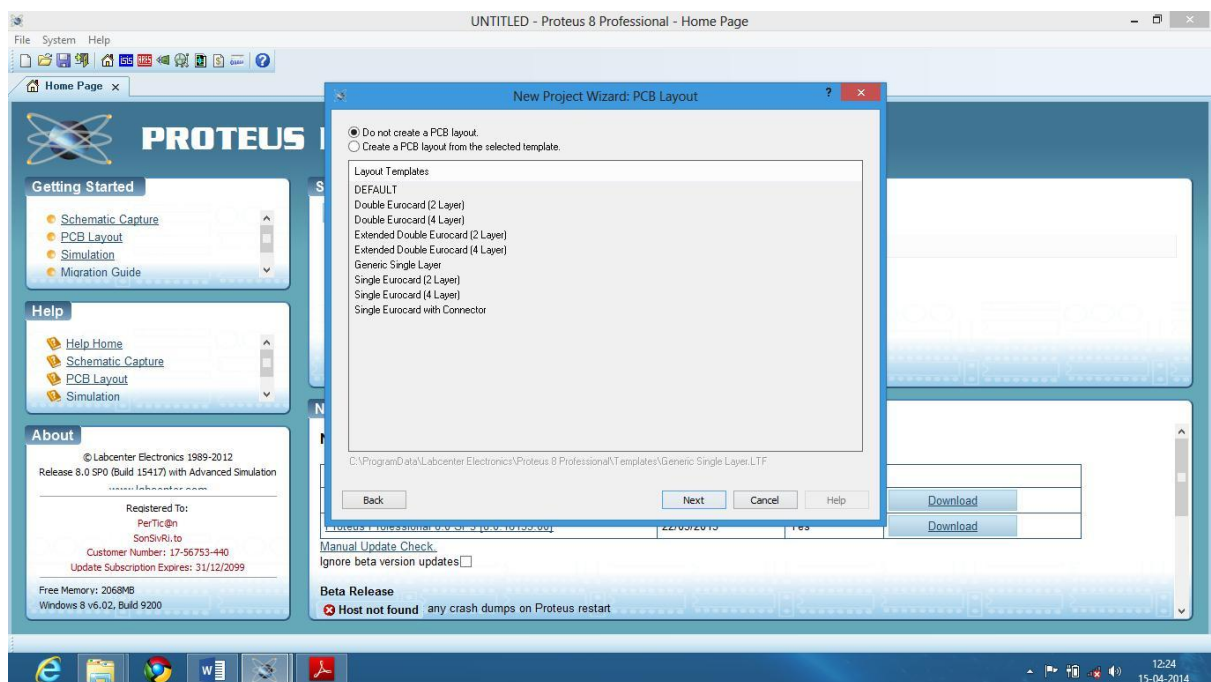


Figure 8.15 (d) PCB design and Circuit simulation software creating PCB layout file

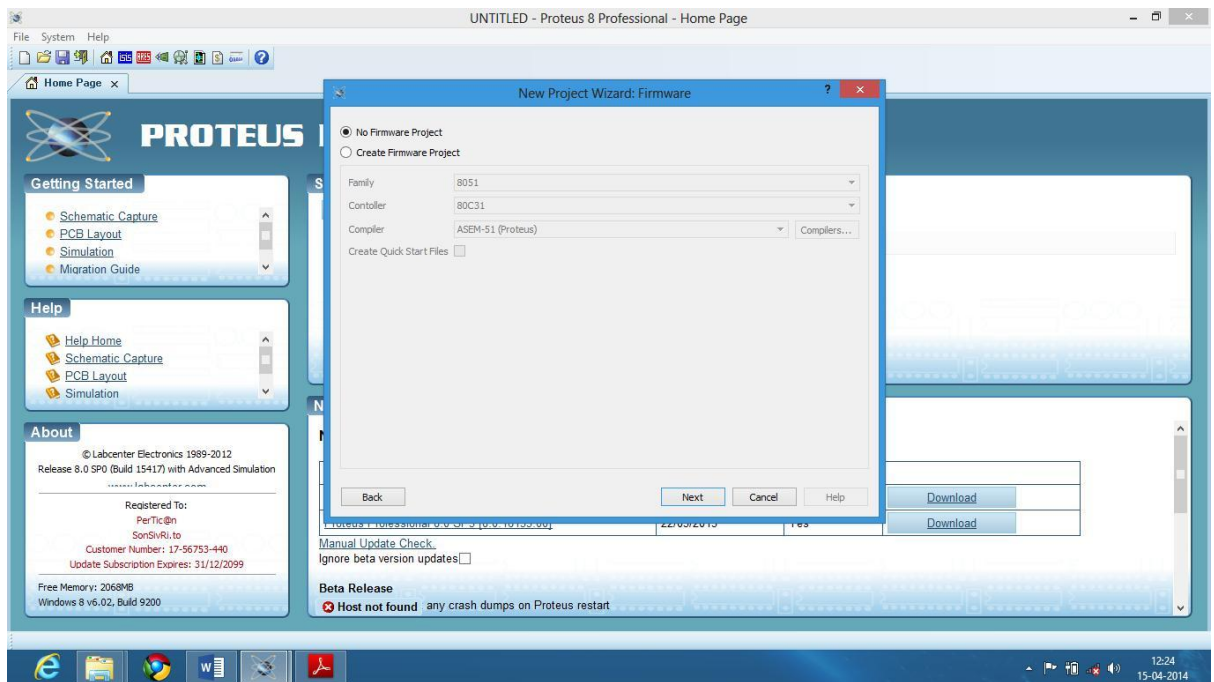


Figure 8.15 (e) PCB design and Circuit simulation software

There is a provision to even write programs for microcontrollers like AVR Studio.

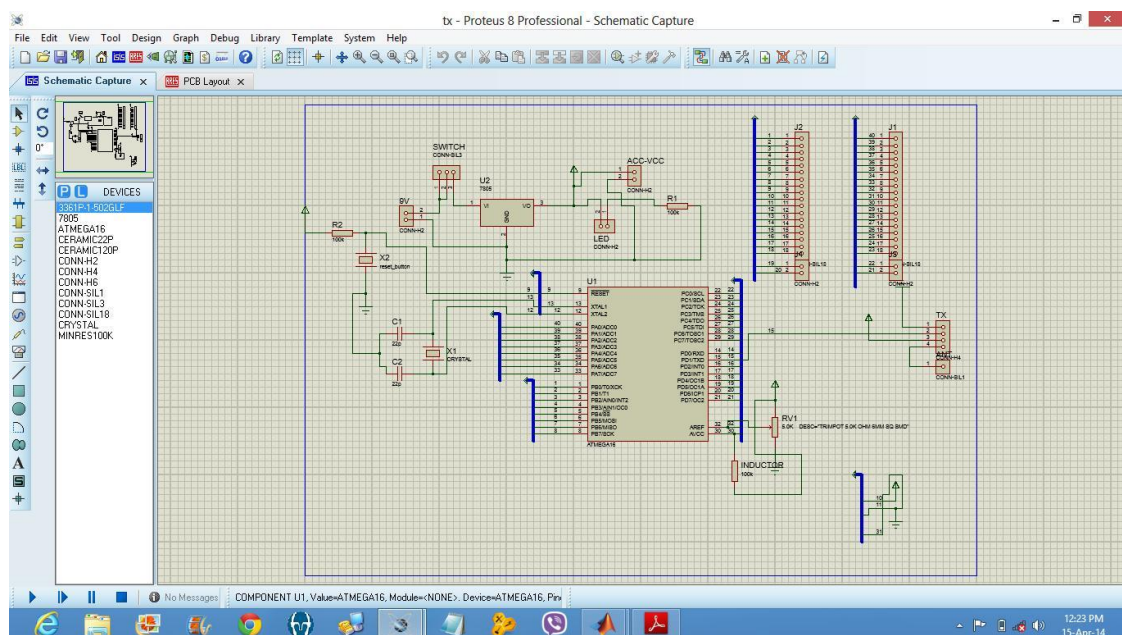


Figure 8.15 (f) Circuit simulation



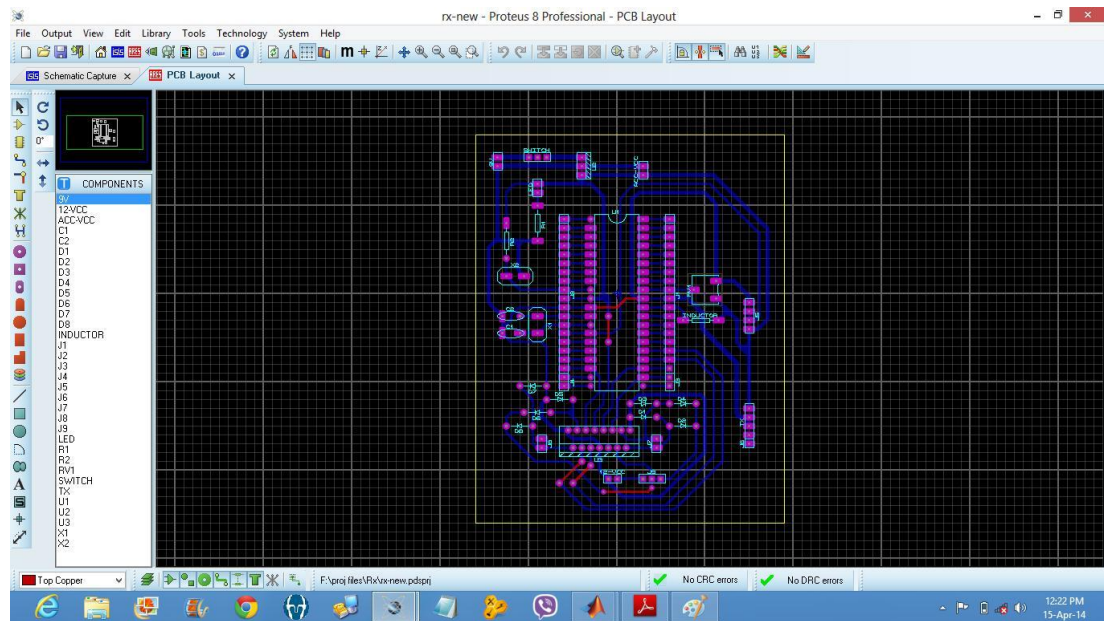


Figure 8.15 (g) PCB design

### 8.13 CONCLUSION:

The software used in this project are listed and briefly explained involving purpose, installation, setup, configuration, and the process to use each. Almost all the software are readily available in the internet and can be downloaded without any difficulty.

## **CHAPTER 9**

### **RESULTS AND CONCLUSION**

#### **9.1 INTRODUCTION**

Internet of things is intended to human to machine, machine to machine communication. Anything i.e. a sensor or a streaming system can be connected to each other over internet and can be accessed from anywhere in the world.

#### **9.2 RESULTS**

We are successfully able to make our raspberry as a web server and it can be accessed globally through internet from anywhere in the world. Exploiting the advantage of this result it was made possible to blink an LED, display a pattern on dot matrix, forward the measured temperature of the surroundings of the central system(Raspberry Pi) on to a webpage which is accessible to the clients of it.

By interfacing camera module to the Raspberry Pi we captured the surroundings of the central system at regular intervals which is useful to spy or monitor the places when required.

#### **9.3 CONCLUSION**

A better communication system was established between client and server which involves giving inputs to the central system by the client through the internet with the help of a well-designed webpage and they were processed in the central system successfully and the results were forward to the client accordingly.

## **CHAPTER 10**

### **SCOPE FOR FUTURE WORK**

1. In the near future the Internet and wireless technologies will connect different sources of information such as sensors, mobile phones and cars in an ever tighter manner. The number of devices which connect to the Internet is seemingly exponentially increasing. These billions of components produce, consume and process information in different environments such as logistic applications, factories and airports as well as in the work and everyday lives of people. The society need new, scalable, and compatible and secure solutions for both the management of the ever more broad, complexly-networked Internet of Things, and also for the support of various business models.
2. In future when load on the central system increases we need to switch over to Raspberry Pi with more amount of RAM.
3. In our vision, the future will witness remarkable progress in intelligent world with every element in the nature is connected to each other to a server and that can be accessed from anywhere on the world and can be corrected or the natural calamities destructions can be eradicated to maximum extent.

## BIBLIOGRAPHY

Davies, E. R. [2005]. *Machine Vision: Theory, Algorithms, Practicalities*, Morgan Kaufmann, San Francisco, CA.

Bezdek, J. C. et al. [2005]. *Fuzzy Models and Algorithms for internet of things*, Springer, New York.

Umbaugh, S. E. [2005]. *Internet of things*, CRC Press, Boca Raton, FL.

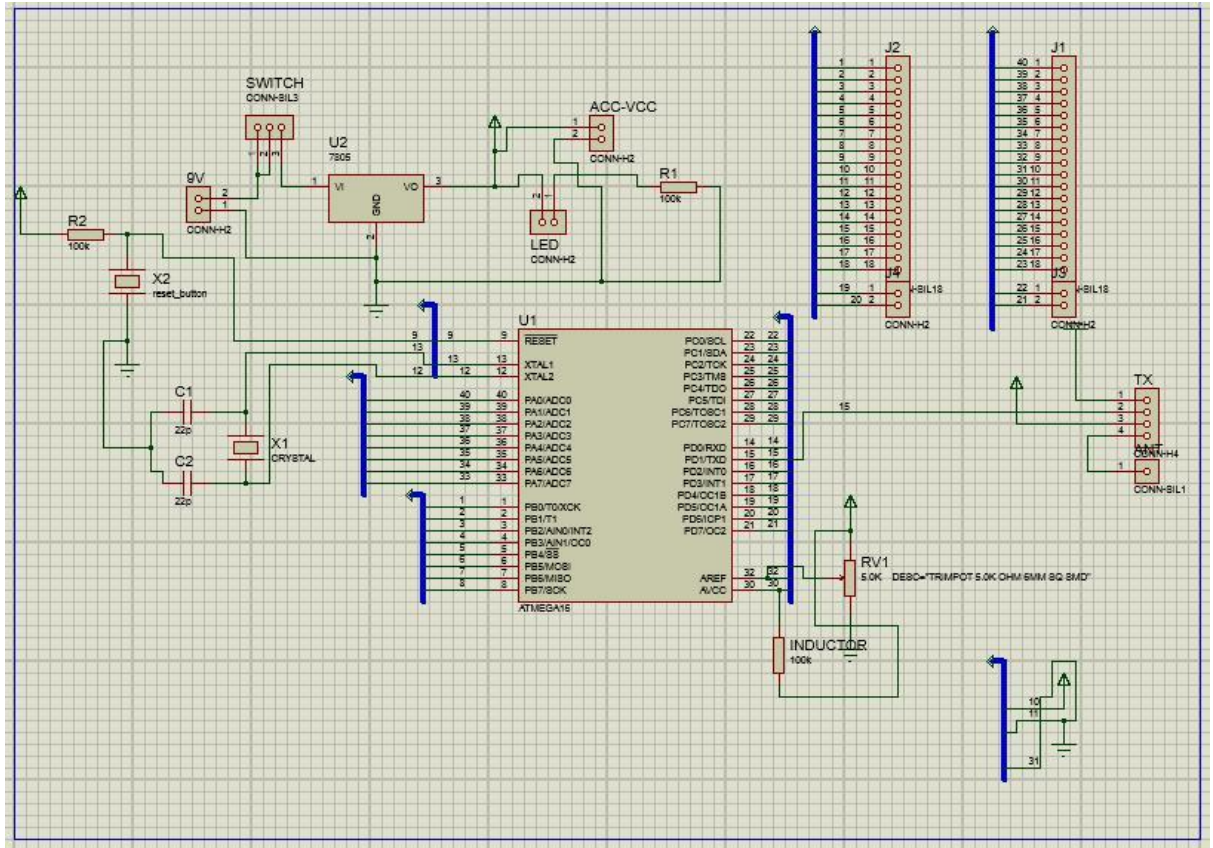
## WEBSITES

- [www.raspberrypi.org.in](http://www.raspberrypi.org.in)
- [www.atmel.com](http://www.atmel.com)
- [www.extremeelectronics.co.in](http://www.extremeelectronics.co.in)
- [www.engineersgarage.com](http://www.engineersgarage.com)
- [www.nex-robotics.com](http://www.nex-robotics.com)
- [www.avrfreaks.com](http://www.avrfreaks.com)
- [www.instrucyables.com](http://www.instrucyables.com)

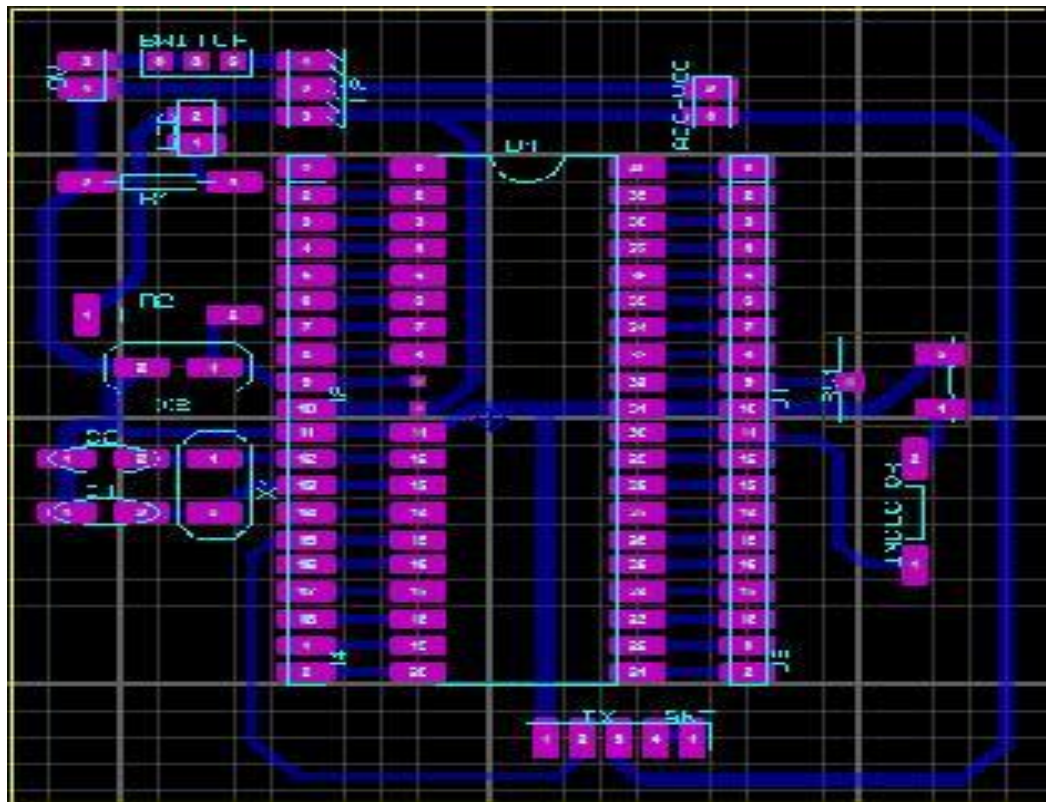


## APPENDIX

## 11.1 ATmega16 BASED RF TRANSMITTER CIRCUIT

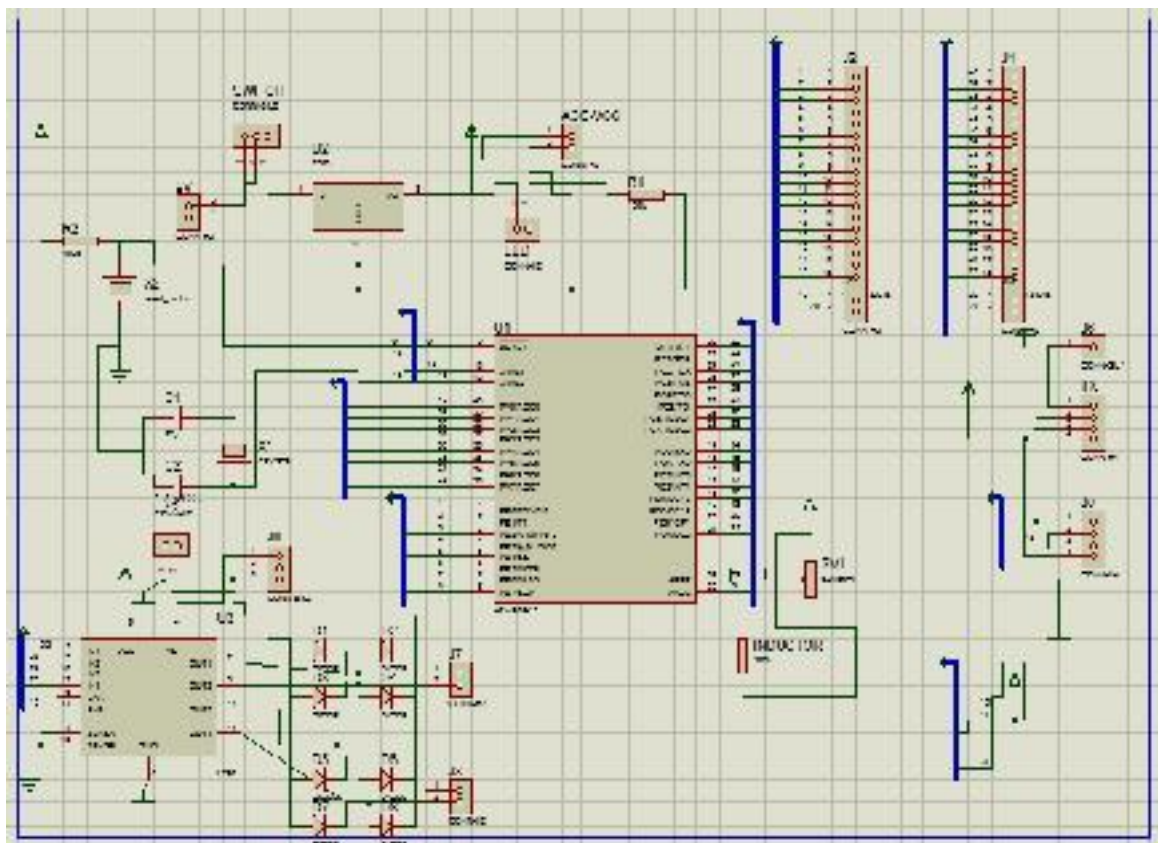


### Fig 11.1 Transmitter circuit

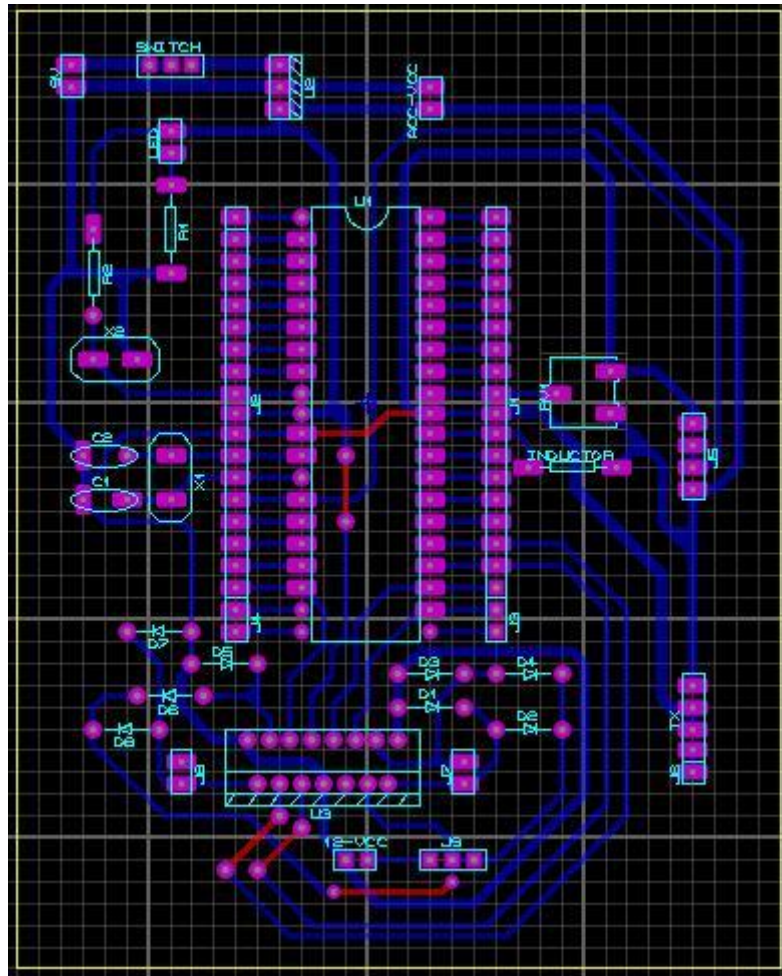


**Fig 11.2 Transmitter layout**

## 11.2 ATmega16 BASED RF RECEIVER CIRCUIT



**Fig 11.3 Receiver circuit**



**Fig 11.4 Receiver layout**

### 11.3 USB Programmer

This device is specially designed to work with Laptops/Notebooks which doesn't have Parallel or serial port. At full clock speed of 16 MHz of the microcontroller it can program the flash at very high speeds in STK500 mode. This programmer is supported in STK500 as well as Human Interface Device (HID) mode. It is supported on all versions of Windows, including Windows Vista and Longhorn, as well as on Linux.

#### Features:

- Compatible to Atmel's STK500V2.
- Compatible with AVR Studio, AVRDUDE and compilers having support for STK500V2 Protocol.
- Supports 2 modes, STK500 and USB-HID for compatibility.



- Adjustable ISP clock allows flashing of devices clocked at very low rate, e.g. 32 kHz.
- High Speed Programming : Programs 32 KB flash in just 15 seconds at full speed of Microcontroller.
- ISP clock can be lowered with a jumper (if the programmer software does not support setting the ISP clock)
- Second USB to Serial converter for processing debug output from the target.
- Uses USB power supply, no external supply required.

Supported on Windows 98, XP, Vista and Linux.

## 1.4 SNAPSHOTS



Fig 11.5 Complete system

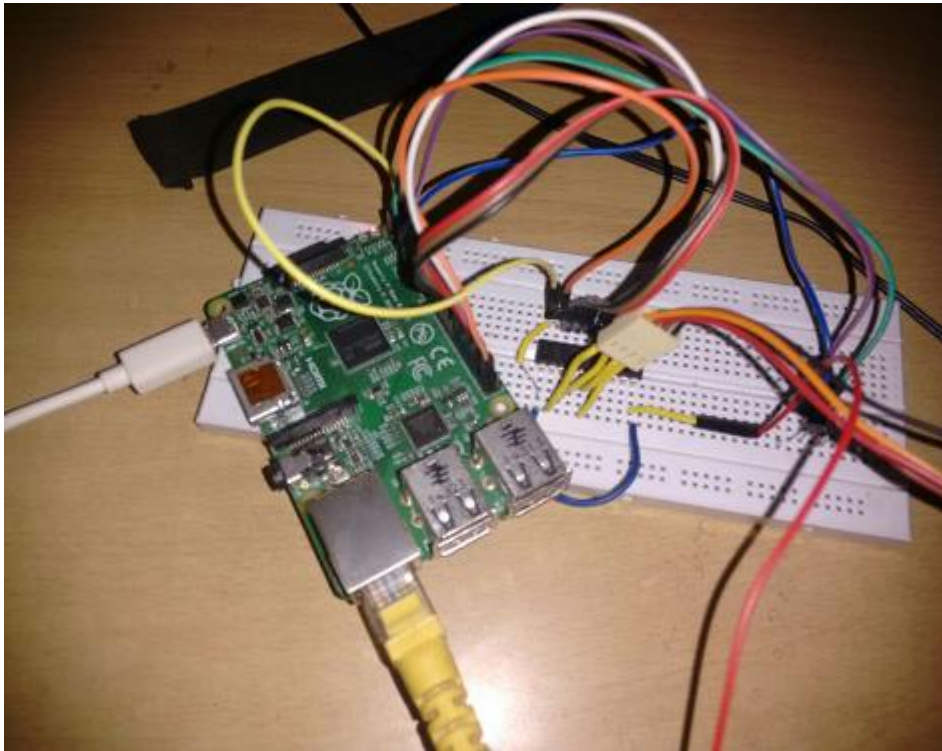


Fig 11.6 Central system

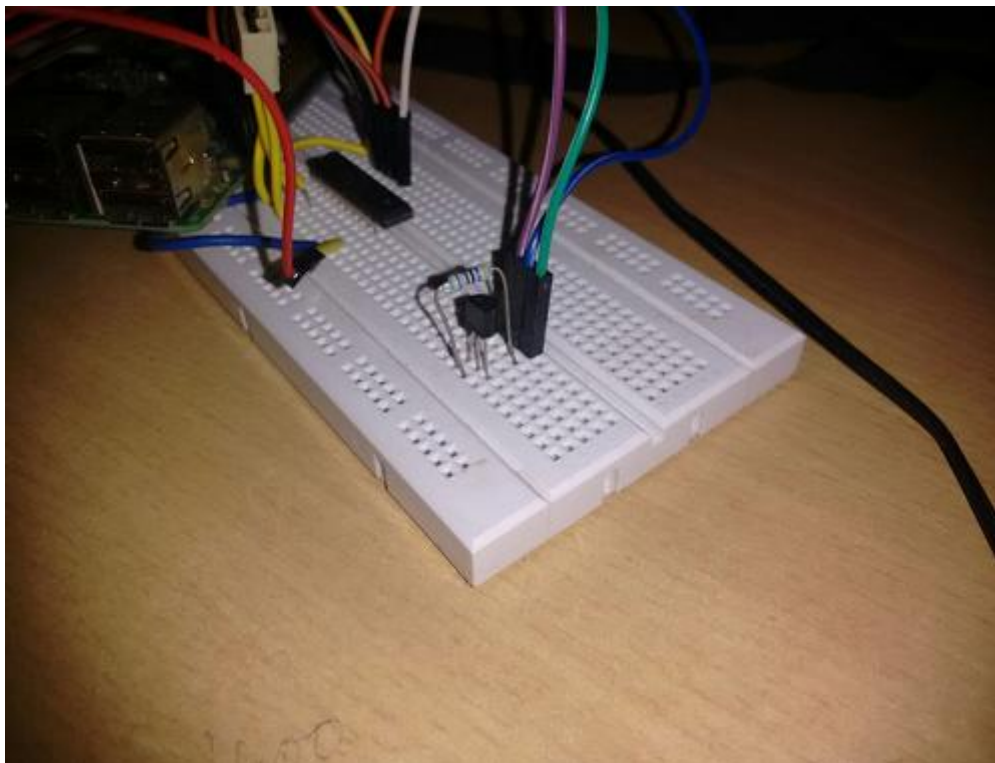


Fig 11.7 Temperature sensor

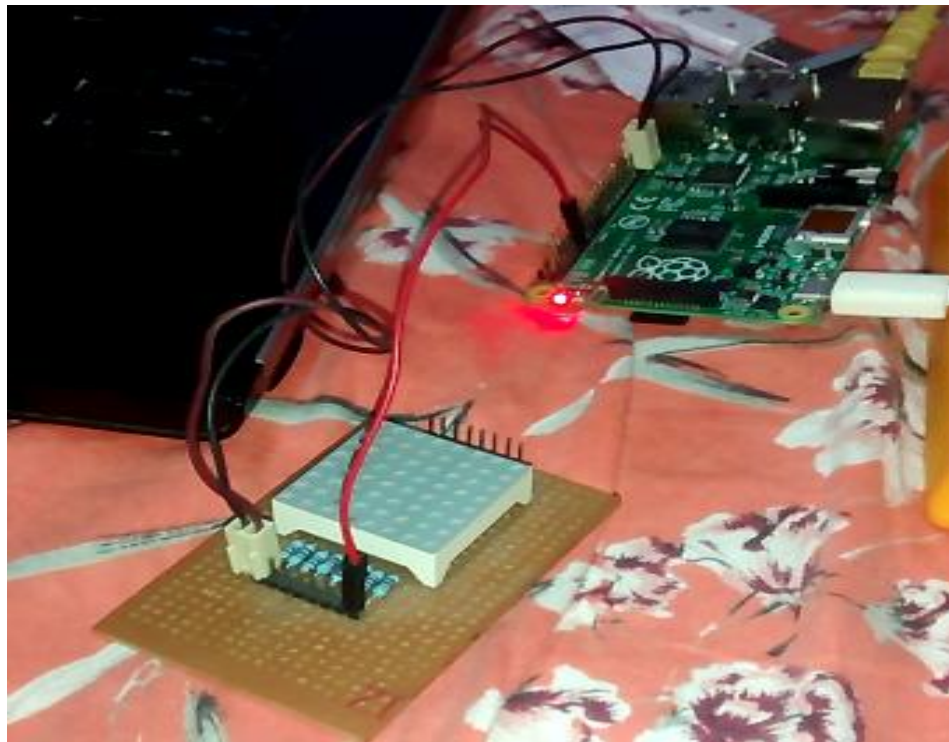


Fig 11.8 Dot matrix



Fig 11.8.1 Dot matrix on state

## 11.5 SOURCE CODE

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.o$

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>IOT</title>

<link rel="stylesheet" type="text/css" href="demo.css" />

<link rel="stylesheet" href="menu.css" type="text/css" media="screen" />

<?php

system ( "gpio mode 0 out" ); // making gpio pins ouput

system ( "gpio mode 2 out" );

system ( "gpio mode 21 out" );

system ( "gpio mode 22 out" );

system ( "gpio mode 23 out" );

system ( "gpio mode 24 out" );

system ( "gpio write 21 1" );

system ( "gpio write 22 1" );

system ( "gpio write 23 1" );

system ( "gpio write 24 1" );

if(isset($_POST["l1_on"])) {

    system ( "gpio mode 0 out" );

    system ( "gpio write 0 1" );

    $p1=system("gpio read 0");

}

```

```
if(isset($_POST["l1_off"])) {  
    system ( "gpio mode 0 out" );  
    system ( "gpio write 0 0" );  
    $p1=system("gpio read 0");  
}  
  
if($p1 == 0)  
    $l1 = "OFF";  
  
else  
    $l1 = "ON";  
  
if(isset($_POST["l2_on"])) {  
    system ( "gpio mode 0 out" );  
    system ( "gpio write 2 1" );  
    $p2=system("gpio read 2");  
}  
  
if(isset($_POST["l2_off"])) {  
    system ( "gpio mode 0 out" );  
    system ( "gpio write 2 0" );  
    $p3=system("gpio read 2");  
}  
  
if($p2 == 0)  
    $l2 = "OFF";  
  
else  
    $l2 = "ON";  
  
if(isset($_POST["capture"])) {  
    system ( "raspistill -o cam.jpg" );  
}
```



```
}

if(isset($_POST["forward"])) {

    system ( "gpio mode 21 out" );

    system ( "gpio write 21 0" );

    system ( "gpio write 22 1" ); // making other gpio pins zero

    system ( "gpio write 23 1" );

    system ( "gpio write 24 1" );

    //$p=system("gpio read 21");

}

if(isset($_POST["backward"])) {

    system ( "gpio mode 22 out" );

    system ( "gpio write 22 0" );

    system ( "gpio write 21 1" ); // making other gpio pins zero

    system ( "gpio write 23 1" );

    system ( "gpio write 24 1" );

    //$p=system("gpio read 22");

}

}

if(isset($_POST["right"])) {

    system ( "gpio mode 23 out" );

    system ( "gpio write 23 0" );

    system ( "gpio write 22 1" ); // making other gpio pins zero

    system ( "gpio write 21 1" );

    system ( "gpio write 24 1" );

    //$p=system("gpio read 23");

}
```

```
if(isset($_POST["left"])) {  
    system ( "gpio mode 24 out" );  
    system ( "gpio write 24 0" );  
    system ( "gpio write 22 1" ); // making other gpio pins zero  
    system ( "gpio write 23 1" );  
    system ( "gpio write 21 1" );  
    //$p=system("gpio read 24");  
}  
  
if(isset($_POST["stop"])) {  
    //system ( "gpio mode 25 out" );  
    system ( "gpio write 21 1" );  
    system ( "gpio write 22 1" );  
    system ( "gpio write 23 1" );  
    system ( "gpio write 24 1" );  
    //system ( "gpio write 25 0" );  
    //$p=system("gpio read 25");  
}  
  
if(isset($_POST["temp"]))  
{  
    $file = "/sys/bus/w1/devices/28-01146407a3ff/w1_slave";  
    $lines = file($file);  
    $temp = explode('=', $lines[1]);  
    $temp = number_format($temp[1] / 1000, 1, '.', '');  
    //echo $temp;  
}
```

?>

<style>

<!--body { behavior: url("csshover3.htc");}

#menu li .drop { background:url("img/drop.gif") no-repeat right 8px;-->

</style>

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/lib\$

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<div id="title">

<div id="title1">Internet Of Things</div>

</div>

<div id="main1">

<ul id="menu2">

<li><a href="#" class="drop">Home</a>

<!-- Begin Home Item -->

<div class="dropdown\_2columns">

<!-- Begin 2 columns container -->

<div class="col\_2">

<h2>Welcome !</h2>

</div>

<div class="col\_2">

<p>Hi and welcome here ! This is a web page where you can control all\$

<p></p>

</div>

<div class="col\_2">

```

    <h2>Cross Browser Support</h2>

</div>

<div class="col_1"> 

    <p>This web page works on all major browsers.</p>
<div class="col_1">

    <p>This web page works on all major browsers.</p>

</div>

</div>

<!-- End 2 columns container -->

</li>

<!-- End Home Item -->

<li></li><li></li><li></li><li></li><li></li>

<li class="menu_right">

    <form action="?php $_PHP_SELF?" method="POST">

        <input type="submit" name="capture" value="capture">

    </form></li>

</ul>

<div id="gallery">

    <div id="slides">

</div>

<div id="cont">

    <div id="cont3">

```

```
<div id="con33"><b>Control a LED</div>

<form action="?php $_PHP_SELF?" method="POST">

<div id="con33"><b>LED 1</b>


        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit" name="l1_on" value="on"> &nb$

    <b>LED 2</b>

        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="submit" name="l2_on" value="on"> &nb$

</form>

</div>

</div>

<div id="cont2">

    <div id="con22">Control a ROBOT</div>

    <div id="con22"><br /><br />

        <form action="?php $_PHP_SELF?" method="POST"

<div id="con22"><br /><br />

        <form action="?php $_PHP_SELF?" method="POST">

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~$

    <input type="submit" name="left" value="left">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n$

    <input type="submit" name="stop" value="stop">&nbsp;&nbsp;&nbsp;&nbsp;&~$

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~$

    </form>

    </div>

</div>
```

```
</div>
```

```
<div id="cont1">
```

```
<div id="con11">Temperature</div>
```

```
<div id="con111">Room Temperature:<br />
```

Click on Temp to read temperature

```
<br />
```

```
<form action="?php $_PHP_SELF?" method="POST">
```

```
<input type="submit" name="temp" value="temp">
```

```
</form> </br>
```

```
<?php echo $temp;?>
```

```
<br />
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div id="fot">
```

```
<div id="fotlinks1">
```

```
<div id="fotlinks4">
```

```
<h2></h2>
```

```
<br /></br>
```

```
</div>
```

```
</div>
```

```
<div id="fotlinks3">

    <div id="fotlinks5">

        <h2>Team</h2>

        <br />

        &nbsp&nbs$

            &$

        </div>

    </div>

    <div id="fotlinks2">

        <h2>About college</h2></br>

        <br />

    </div>

</div>

<div align="center">Copyright &#x000A9;&nbsp&nbsp&nbsp;2015 - All Rights Reserved<br />

<!--Thanks to <a href="http://www.makarska-rivijeraapartment.com/">Croatia Maka$

</div>

</body>

</html>
```

**Transmitter source code:**

```
#include<avr/io.h>

#include<util/delay.h>

#include<string.h>

long int data1;

#define BAUD 9600

#define BAUDRATE((F_CPU)/(BAUD*16UL)-1)

#ifndef F_CPU

#define F_CPU 1000000UL

#endif

void uart_init(void)

{

UBRRH=(BAUDRATE>>8);

UBRRL=BAUDRATE;

UCSRB|=(1<<TXEN)|(1<<RXEN);

UCSRC|=(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);

}

void uart_transmit(unsigned char data)

{

while(!(UCSRA&(1<<UDRE)));

UDR=data;

}

unsigned char USART_receive(void)

{

while ((UCSRA&0x80)==0x00)
```



```

{
;
}

data1=UDR;

return data 1;

}

void main()

{

uart_init();

long int x;

while(1)

{

    x=USART_receive();

    uart_transmit(x);

}

}

```

### **Receiver code:**

```

#include<avr/io.h>

#include<util/delay.h>

#define sbi(x,y) x|=_BV(y)

#define cbi(x,y) x&=~(_BV(y))

void initports(void);

void leftmf(void);

void leftmb(void);

void leftms(void);

void rightmb(void);

```

```

void rightms(void);

void rightmf(void);

void initpwm(void);

#define BAUD 9600

#define BAUDRATE((F_CPU)/(BAUD*16UL)-1)

#ifndef F_CPU

#define F_CPU 1000000UL

#endif

long int data;

void uart_init(void)

{

UBRRH=(BAUDRATE>>8);

UBRRL=BAUDRATE;

UCSRB|=(1<<TXEN)|(1<<RXEN);

UCSRC|=(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);

}

unsigned char USART_receive(void)

{

While((UCSRA&0x80)==0x00)

{

;

}

data=UDR;

return data;

}

Void main()

```

```
{  
  
uart-init();  
  
initports();  
  
initpwm();  
  
char p;  
  
while(1)  
{  
  
p=USART_receive()  
  
if(p=='f')  
{  
  
OCR1A=220;  
  
OCR1B=220;  
  
leftmf();  
  
rightmf();  
  
_delay_ms(200);  
  
p='a';  
  
}  
  
else if(p=='b')  
{  
  
OCR1A=220;  
  
OCR1B=220;  
  
leftmb();  
  
rightmb();  
  
_delay_ms(200);  
  
p='a';  
  
}  
  
}
```

```
else if(p=='1')
{
OCR1A=220;

OCR1B=220;

leftmb();

rightmf();

_delay_ms(200);

p='a';

}

else if(p=='r')
{

OCR1A=220;

OCR1B=220;

leftmf();

rightmb();

_delay_ms(200);

p='a';

}

else
{

OCR1A=0;

OCR1B=0;

leftms();

rightms();

p='a';

}
```

```
}  
  
}  
  
void initports(void)  
{  
  
sbi(DDRD,PD5);  
  
  
sbi(DDRC,PC2);  
  
  
sbi(DDRC,PC3);  
  
  
  
sbi(DDRD,PD4);  
  
  
sbi(DDRC,PC0);  
  
  
  
sbi(DDRC,PC1);  
  
  
  
cbi(PORTC,PC2);  
cbi(PORTC,PC3);  
cbi(PORTC,PC0);  
cbi(PORTC,PC1);  
sbi(PORTD,PD4);  
sbi(PORTD,PD5);  
  
}  
  
void leftmb(void)  
{  
  
sbi(PORTD,PD4);
```

```
cbi(PORTC,PC0);  
sbi(PORTC,PC1);  
}
```

```
void leftmb(void)  
{  
sbi(PORTD,PD4);  
cbi(PORTC,PC0);  
sbi(PORTC,PC1);  
}
```

```
void rightmf(void)  
{  
sbi(PORTD,PD5);  
cbi(PORTC,PC2);  
sbi(PORTC,PC3);  
}
```

```
void leftmf(void)  
{  
sbi(PORTD,PD4);  
sbi(PORTC,PC0);  
cbi(PORTC,PC1);  
}
```

```
void rightmb(void)  
{  
sbi(PORTD,PD5);  
sbi(PORTC,PC2);
```

```
cbi(PORTC,PC3);

}

void leftms(void)

{

cbi(PORTD,PD4);

cbi(PORTC,PC0);

cbi(PORTC,PC1);

}

void rightms(void)

{

cbi(PORTD,PD5);

cbi(PORTC,PC2);

cbi(PORTC,PC3);

}

void initpwm(void)

{

TCCR1A=_BV(WGM10)

|_BV(COM1A1)

|_BV(COM1B1);

TCCR1B=_BV(CS11)

|_BV(WGM12);

}
```