Web Server Based Home Automation

Submitted in partial fulfillment of the requirements

Of the degree of

Bachelor of Engineering in

ELECTRONICS & TELECOMMUNICATION

By

Mohammed Fahad Kheratkar	12ET99
Ammar Rumane	12ET66
Faisal Rumane	12ET98
Mohd Ajmal Siddiqui	12ET95

Supervisor

Prof. Banda Nawaz Sir

Department of Electronics & Telecommunication

Anjuman-I-Islam's Kalsekar Technical Campus

2014-201

CERTIFICATE

This is to certify that the project entitled "Web Server Based Home Automation System" is the bonafide work carried out by **MR.M.FAHAD KHERATKAR, MR.AMMAR RUMANE,MR.FAISAL RUMANE,MR.M.AJMAL SIDDIQUI** student of ANJUMAN-I-ISLAM'S KALSEKAR TECHNICAL CAMPUS, during the Academic year 2014-2015, in partial fulfilment of the requirements for the award of the Degree in ELECTRONICS & TELECOMMUNICATION and that the project has not formed the basis for the award previously of any degree, diploma, associate ship, fellowship or any other similar title.

--------------------------	--------------------------
Supervisor	**External**
--------------------------	--------------------------
HOD	**Director**

Project Report Approval for B. E

This project entitled "Web Server Based Home Automation" is done by **MR.M.FAHADKHERATKAR, MR.AMMARRUMANE, MR.FAISALRUMANE, MR.M.AJMAL SIDDIQUI** approved for the Degree of Bachelor of Engineering in Electronics & Telecommunication.

--------------------------	-------------------------
Supervisor	**External**

Director

Date Place

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Signature
Mohammed Fahad Kheratkar	___________
Ammar Rumane	___________
Faisal Rumane	___________
Mohd Ajmal Siddiqui	___________

Date:

ACKNOWLEDGEMENT

It is indeed a matter of great pleasure and proud privilege to be able to present this project on "**Web Server Based Home Automation System** ".

The completion of the project work is a millstone in student life and its execution is inevitable in the hands of guide. We are highly indebted the project guide **Prof. Banda Nawaz** for his invaluable guidance and appreciation for giving form and substance to this report. It is due to his enduring efforts; patience and enthusiasm, which has given a sense of direction and purposefulness to this project and ultimately made it a success.

We would like to tender our sincere thanks the staff members for their co-operation.

We would also like to express our deep regards and gratitude to the **Prof. Mujeeb Tamboli.**

We would wish to thank the non - teaching staff and our friends who have helped us all the time in one way or the other.

Really it is highly impossible to repay the debt of all the people who have directly or indirectly helped us for performing the project.

ABSTRACT

Now A Days home and building automation systems are used more and more. On the other hand, they provide increased comfort especially when employed in a private home .On the other hand, automation systems installed in commercial buildings do not only increase comfort ,but also allow centralized control of Security, Alarm Systems, Door lock, CCTV, Heating, Sensing, Air conditioning and lightning. Hence, they contribute to an overall cost reduction and also to energy saving which is certainly a main issue today.

Instead of PC based servers. Arduino based servers are becoming trend of today's market. Cost reduction is achieved using Arduino along with Ethernet module as Embedded Web Server. Idea is utilized for monitoring and controlling maximum no. of either home appliances or industry devices. Without using a computer, Ethernet module can communicate to the owner of the overall system, who is able to manage appliances from any location outside. This server provides a powerful networking solution and enables web access for automation and monitoring of different systems. For industry automation, instrumentation and household devices control, this is an optimized solution. System home page can be accessed using web browser. Operational status of the appliances can be observed and changed in case of necessity. This report proposes development of low cost system for above purpose. Different sensors installed at working place help in sensing real time environmental conditions like temperature, light, humidity etc.

LIST OF FIGURES

INDEX

Chapter 1

INTRODUCTION

Our project is based on Web Server which uses Arduino as a processor to handle various web requests from web browser operated through a PC, mobile or tabs. A web page is created with basic security algorithm and it also contains user name password for authentication process. After getting access to the main page, we can see the various control switches, these control switches help us to switch on or of a particular appliance or device connected to the arduino control board. The web page also gives the status of the status of the appliance. After we send the command "ON" from the web page, the http request travel through the network or internet and reaches the arduino board. The arduino board has an Ethernet shield that has a designated IP and MAC address. The Ethernet shield helps the arduino to connect to a network or internet. The http request received by Ethernet shield and it is then forwarded to arduino mega board. The board process the request and perform the function accordingly. The function involves triggering the relay board to switch the desired appliances ON or OFF.

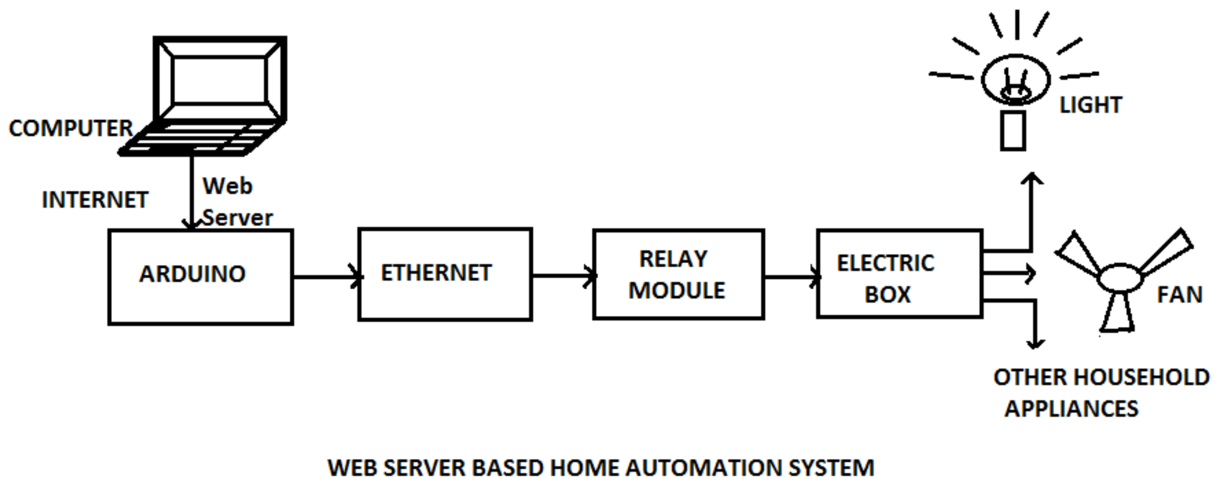


Fig 1.1 Web server based Home Automation

Chapter 2

REVIEW OF LITERATURE SURVEY

2.1 Existing System

2.1.1 Android Based Home Automation

Generally in today's modern world human beings are addicted to using modern equipment. The intention of this project is to make an Android OS based smartphone or tablet workable for controlling each and every appliance of industries or household. There are several Android applications available in the market to turn our Android-based smart phone or tablet into a remote control for our home. If we want to control a system in home or want to get start, we need an INSTEON controller and also INSTEON controllable devices. In addition to this, we need to install different Android applications for home automation. Some of these Android applications are INSTEON Hub, MobiLinc, Conductor, wdISY, Touch Switch,etc.

Fig 2.1.1 Andriond Based Home Automation

In recent years, technology is advancing and houses are becoming smarter by gradually shifting from conventional to centralized switches. Conventional switches are located in different places of the house. These switches are difficult to operate for the users and especially for physically handicapped people as they are difficult for them to approach and operate. Android technology based remote controlled system provides a simple solution to home automation. In this project, a microcontroller is used from the 8051 family, and the loads are interfaced with the 8051 microcontroller by using TRIACs and Opto-Isolators. The remote operation is achieved through any Android smartphone with a Graphical User Interfaced based touch screen operation. GUI is nothing but a type of user interface that allows users to interact with the electronic devices through visual indicators and graphical icons. In order to achieve this, Android smartphone acts as a transmitter and sends on or off commands to the receiver where loads are connected. So, by using wireless technology, we can operate the remote switch on the transmitter and the loads can be turned on or off remotely.

2.1.2 Zigbee Based Voice Controlled Wireless Smart Home System

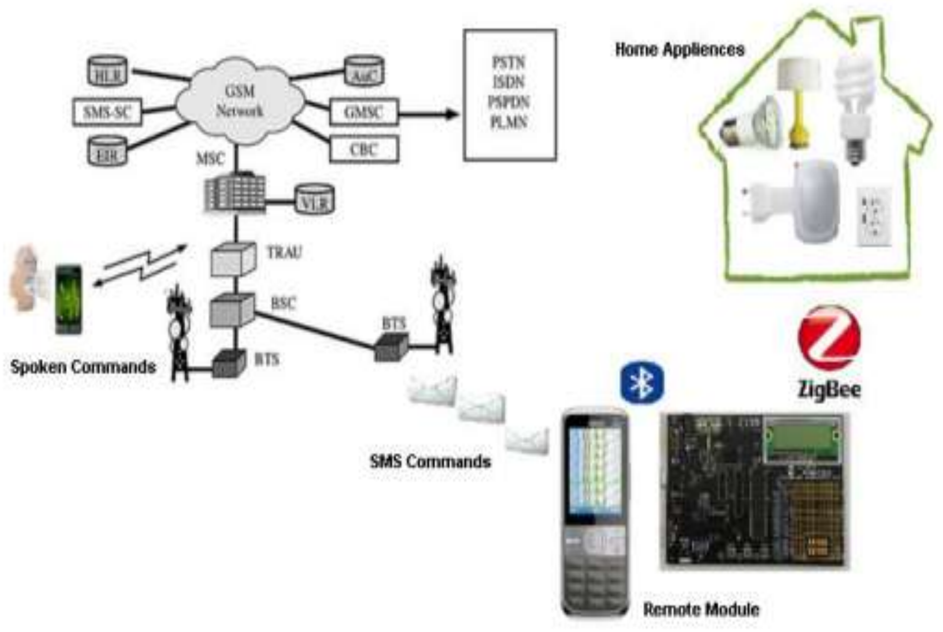


Fig 2.1.2 Zigbee Based Voice Controlled Wireless Smart Home System

In this project,a voice controlled wireless smart home system has been presented for elderly and disabled people. The proposed system has two main components namely (a) voice recognition system, and (b) wireless system. LabView software has been used to implement the voice recognition system. On the other hand, ZigBee wireless modules have been used to implement the wireless system. The main goal of this system is to control home appliances by using voice commands. The proposed system can recognize the voice commands, convert them into the required data format, and send the data through the wireless transmitter. Based on the received data at the wireless receiver associated with the appliances desired switching operations are performed. The proposed system is a low cost and low power system because ZigBee is used. Additionally the proposed system needs to be trained of voice command only once. Then the system can recognize the voice commands independent of vocabulary size, noise, and speaker characteristics.

In this method, the user with any mobile speaks voice commands, the mobile application convert the voice command in to text and payload the command on GSM network via SMS. Below Table shows basic command which is appended in SMS.

Voice Commands	**SMS Commands**
Main console on	‘MCONE’
Main console off	‘MCOFFE’
Zone 1 appliances on	‘Z1ONE’
Zone 1 appliances off	‘Z1OFFE’
Zone 2 appliances on	‘Z2ONE’
Zone 2 appliances off	‘Z2OFFE’
Light zone 1 on	‘LZ1ONE’
Light zone 1 off	‘LZ1OFFE’
Fan zone 2 on	‘FZ2ONE’
Fan zone 2 off	‘FZ2OFFE’

On the receiver side these commands are received and transfer to the controller using Bluetooth medium which is further preceded.

2.2 Problem Description

Referring to the title "Home Automation System", the system that will be developed is because of the several problems that arise in our today's modem society. They are:

1. A system of home automation products can cost more than RM 1000 for end users. Although this technology is available in the local and international market, not all people can afford it.

2. The increasing number of housebreaking caused the anxiety to people to leave their house. We will feel inferiors every time going out for a vacation because of the incomplete security system surrounding the house. We cannot always rely on our neighbour to look after our house while we are away from home unless we are rich enough to hire a house security guard.

2.3 Proposed System

2.3.1 Web Server Based Home Automation System

Instead of PC based servers. Arduino based servers are becoming trend of today's market. Cost reduction is achieved using Arduino along with Ethernet module as Embedded Web Server. Idea is utilized for monitoring and controlling maximum no. of either home appliances or industry devices. Without using a computer, Ethernet module can communicate to the owner of the overall system, who is able to manage appliances from any location outside. This server provides a powerful networking solution and enables web access for automation and monitoring of different systems.For industry automation, instrumentation and household devices control, this is an optimized solution. System home page can be accessed using web browser. Operational status of the appliances can be observed and changed in case of necessity. This report proposes development of low cost system for above purpose. Different sensors installed at working place help in sensing real time environmental conditions like temperature, light, humidity etc.

Web server hosts a web site and provides reliable services for any requesting client. . Such web servers are developed using general purpose computers. They use different kind of operating systems such as NT, Unix, Linux Windows etc. Fig Different Devices Connected For Automation Such systems apply typical client-server architecture where, the client accesses the server through the LAN router and the Internet. Client sends the request to the server. This request is processed by the router to connect to the Internet. The web processes the request made and finally connects to the desired web server. Requested data is sent to the client. An embedded web server is a microcontroller including software and application code to monitor and control the systems. Microcontroller or Arduino is an integral part of an embedded network and create a way for easy controlled activities of any device from any remote location. Such servers designed using very low resource usage, are highly reliable, portable and secure systems.

2.4 Potential Benefits

The potential benefits we can gain from home automation are almost only limited by imagination and as such it would be infeasible to create a comprehensive list of them. The short list below exemplifies potential benefits in four areas of home automation. The examples are meant to spark the imagination.

Energy Savings Through user tracking both in- and outdoors, a home automation system would potentially be able to make sure that, for example, no unnecessary light or heat is turned on in individual rooms.

Convenience Through Web based access to the home automation system a forgetful user will potentially no longer have to worry about if the coffee machine was left on when he left for work. Simply go to a Web page, check it, and turn it off if necessary.

Security Tracking user locations can assist in automatic alarm system arming. Also, security cameras might be accessed from a vacation to check that the house is alright.

Home Entertainment When engaging in movie watching, the lights might be set to an appropriate dimming level. When listening to music, speakers might be changing from room to room for your listening pleasure throughout the house. Digital paintings on the wall might change according to persons currently occupying the room.

Chapter 3
HARDWARE & SOFTWARE SPECIFICATION

3.1 Hardware Specification

The following are the hardware we are proposing to use in our project:

3.1.1 Arduino Mega board (2560):

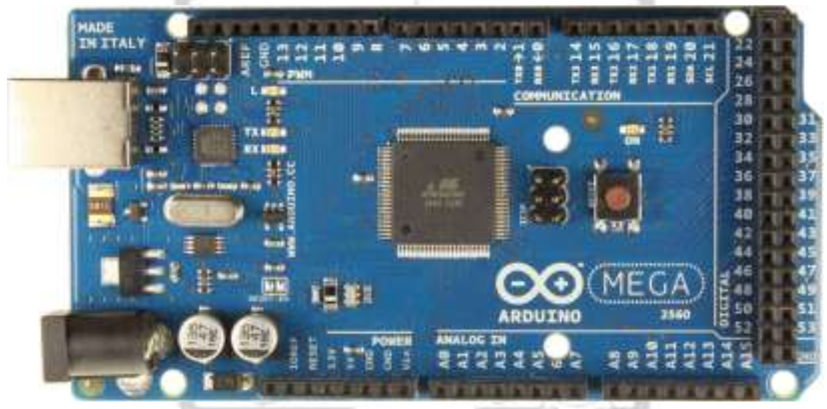


Fig.3.1.1 Arduino Mega Board (2560)

Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

The Mega 2560 is an update to the Arduino Mega, which it replaces.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 boards)programme dasa USB-to-serial converter.

Revision 2 of the Mega2560 board has a resistor pulling the 8U2 HWB line to ground, making it easiertoput into DFU mode.Revision 3 of the board has the following new features:
1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
Stronger RESET circuit.
Atmega 16U2 replace the 8U2.

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt()function for details.

PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analogReference().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2(ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports TWI and SPI communication. The Arduino software includes a Wire library to simplify use of the TWI bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference andtutorials.

The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. TheATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).

3.1.2 W5100 Ethernet Shield

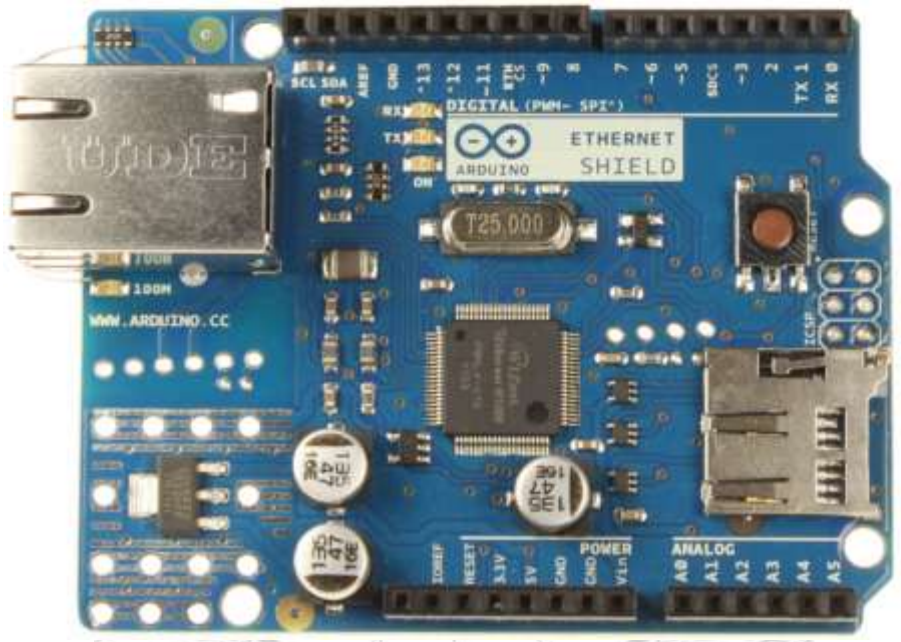


Fig3.1.2 W5100 Ethernet Shield

Overview

The Arduino Ethernet Shield connects your Arduino to the internet in mere minutes. Just plug this module onto your Arduino board, connect it to your network with an RJ45 cable (not included) and follow a few simple instructions to start controlling your world through the internet. As always with Arduino, every element of the platform – hardware, software and documentation – is freely available and open-source. This means you can learn exactly how it's made and use its design as the starting point for your own circuits. Hundreds of thousands of Arduino boards are already fueling people's creativity all over the world, everyday

- Requires an Arduino board (not included)
- Operating voltage 5V (supplied from the Arduino Board)
- Ethernet Controller: W5100 with internal 16K buffer

- Connection speed: 10/100Mb
- Connection with Arduino on SPI port

Description

The Arduino Ethernet Shield allows an Arduino board to connect to the internet. It is based on the Wiznet W5100ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The most recent revision of the board exposes the 1.0 pinout on rev 3 of the Arduino UNO board.

The Ethernet Shield has a standard RJ-45 connection, with an integrated line transformer and Power over Ethernet enabled.

There is an onboard micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Uno and Mega (using the Ethernet library). The onboard microSD card reader is accessible through the SD Library. When working with this library, SS is on Pin 4. The original revision of the shield contained a full-size SD card slot; this is not supported.

The shield also includes a reset controller, to ensure that the W5100 Ethernet module is properly reset on power-up. Previous revisions of the shield were not compatible with the Mega and need to be manually reset after power-up.

The current shield has a Power over Ethernet (PoE) module designed to extract power from a conventional twisted pair Category 5 Ethernet cable:

- IEEE802.3af compliant
- Low output ripple and noise (100mVpp)
- Input voltage range 36V to 57V
- Overload and short-circuit protection
- 9V Output
- High efficiency DC/DC converter: typ 75% @ 50% load
- 1500V isolation (input to output)

The shield does not come with the PoE module built in, it is a separate component that must be added on.

Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 10, 11, 12, and 13 on the Uno and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general I/O. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

Note that because the W5100 and SD card share the SPI bus, only one can be active at a time. If you are using both peripherals in your program, this should be taken care of by the corresponding libraries. If you're not using one of the peripherals in your program, however, you'll need to explicitly deselect it. To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output.

The shield provides a standard RJ45 Ethernet jack.

The reset button on the shield resets both the W5100 and the Arduino board.

The shield contains a number of informational LEDs:

- PWR: indicates that the board and shield are powered
- LINK: indicates the presence of a network link and flashes when the shield transmits or receives data
- FULLD: indicates that the network connection is full duplex
- 100M: indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- RX: flashes when the shield receives data
- TX: flashes when the shield sends data
- COLL: flashes when network collisions are detected

The solder jumper marked "INT" can be connected to allow the Arduino board to receive interrupt-driven notification of events from the W5100, but this is not supported by the Ethernet library. The jumper connects the INT pin of the W5100 to digital pin 2 of the Arduino.

3.1.3 Relay Module

3.1.3.1 Relay Brick

Fig 3.1.3.1(a) Relay Brick

The Relay Brick uses some simple electronics principles that you can apply to other Output Devices for Arduino. Here's the fundamental principle:

"Electronic Devices using small amount of power, can Control a much larger amount of power"

The Relay Brick does this in two steps. Let's look at a diagram of what's on that little Brick:

First, notice the cable connector has a standard pattern of wires: Ground-Voltage-Signal (see the GVS labels on the Sensor Shield above). So the brick has +5 volt power available. Here's what happens:

When the Blink Software Sketch does " digitalWrite(13, HIGH); " the Arduino connects the Signal wire to HIGH, which is +5 Volts.

+5 volts flow through 10,000 ohm resistor R1 to the Base of transistor Q1, and a current of about .0005 amps (500 microamps) flows and turns the transistor ON.

The transistor connects one end of the electromagnet inside the relay to Ground (the other end is already connected to 5 volts), and a current of about .07 amps flows. We say the transistor has a Current Gain of more than 100.

The electromagnet pulls the switch contact inside and moves it to connect the COM terminal to the NO (Normally Open) terminal.

A Lamp, or other load, is connected by the relay contacts. Let's say it might be a 100 watt lamp.

Details: the diode across the electromagnet conducts in the reverse direction when the transistor is turned off to protect against a voltage spike.

Details: An LED and it's current limiting resistor are also connected across the electromagnet and it lights up when the relay is turned on. Just for you...

This is the way high power can be controlled by a very small power from Arduino. Let's quickly think about the numbers:

Arduino outputs 5 Volts at .0005 amps. Multiply and that's .0025 watts (2.5 milliwatts) Not Much!

The transistor controls 5 Volts at .07 amps. That's, um, .35 watts.

The relay controls, say, 220 Volts at .5 amps. That's 110 watts.

So with this little brick, Arduino controls a power 4400 times it's own power. That's what this power control stuff is all about.

How Relay Contacts Work

Look at the photo of a relay down below on the right. Notice the 3 screw-type terminals. They are labelled "NO", "COM", "NC". Those labels mean:

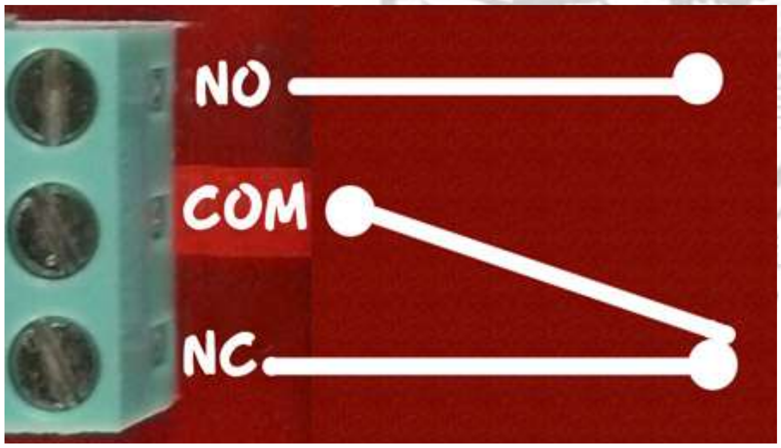


Fig 3.1.3.1(B) Relay Contacts

NO: Normally Open

COM: Common Connection

NC: Normally Closed

Look at the diagram on the right. This shows the switch that is inside the relay. This switch is "thrown" by the electromagnet inside. The diagram shows that COM is connected to the Normally Closed contact. That's the case when the relay is off. When the relay is turned on the electromagnet flips the switch up and COM is then connected to Normally Open. So, if we want a lamp to be on when the relay is on, we connect our circuit from COM to NO.

3.1.3.2 Relay Module Schematic

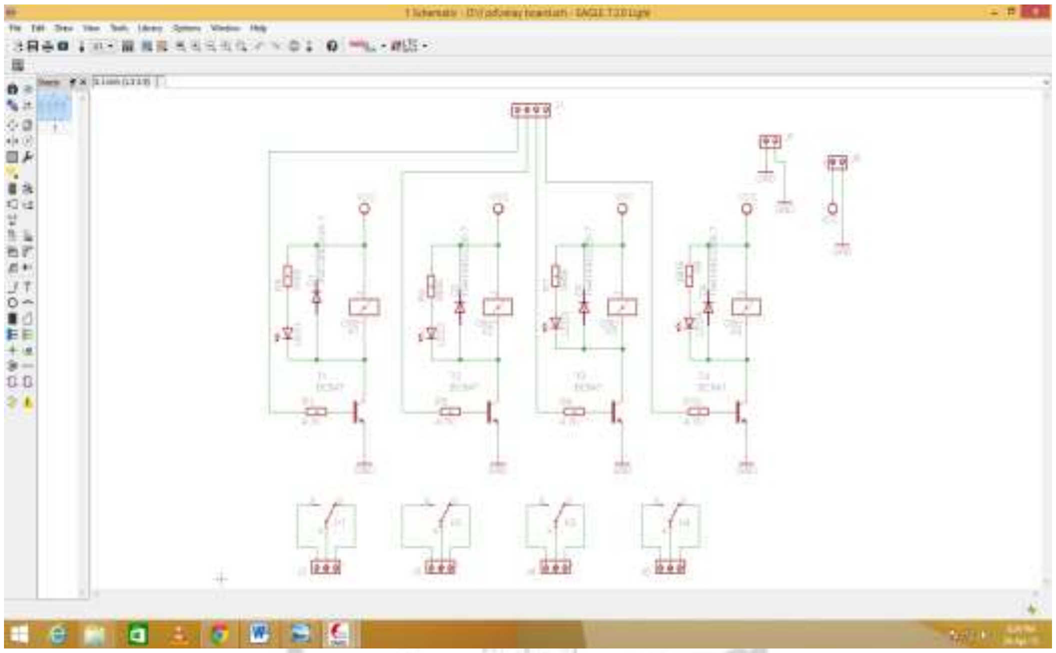


Fig 3.1.3.2 Relay Module Schematic

3.1.3.3 Relay Module Layout

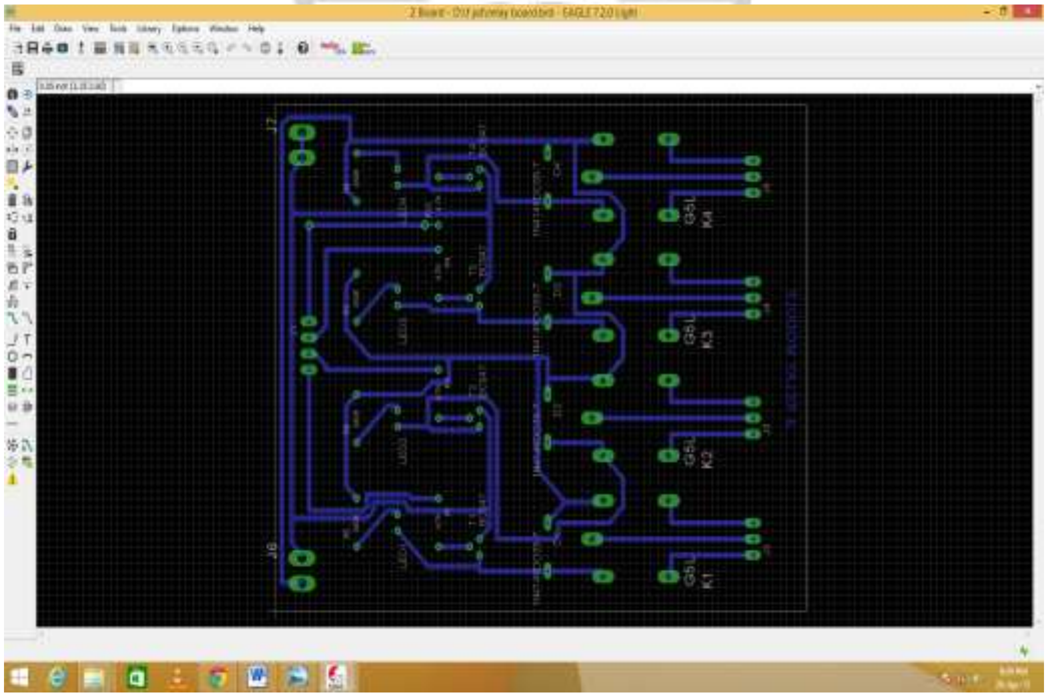


Fig 3.1.3.3 Relay Module Layout

Take a print of layout on photo paper or transparent sheet then put that sheet on copper clad and start ironig it.Ironing the printed layout transfers the ink from the paper going to the PCB board. You need to set your iron's temperature to the highest setting if your paper is thick but if not, set it to the medium setting.

3.1.3.4 PCB Etching Process

All PCB's are made by bonding a layer of copper over the entire substrate, sometimes on both sides. Etching process has to be done to remove unnecessary copper after applying a temporary mask, leaving only the desired copper traces.

Though there are many methods available for etching, the most common method used by electronics hobbyists is etching using ferric chloride ir hydrochloric acid. Both are abundant and cheap. Dip the PCB inside the solution and keep it moving inside. Take it out at times and stop the process as soon as the copper layer has gone. After etching, rub the PCB with a little acetone to remove the black colour, thus giving the PCB a shining attractive look. The PCB layout is now complete.

3.1.3.5 Soldering

- For soldering of any joints first the terminal to be soldered are cleaned to remove oxide film or dirt on it. If required flux is applied on the points to be soldered.
- Now the joint to be soldered is heated with the help of soldering iron. Heat applied should be such that when solder wire is touched to joint, it must melt quickly.
- The joint and the soldering iron are held such that molten solder should flow smoothly over the joint.
- When joint is completely covered with molten solder, the soldering iron is removed.
- The joint is allowed to cool, without any movement.
- The bright shining solder indicates good soldering.
- In case of dry solder joint, an air gap remains in between the solder maternal and the joint. It means that soldering is improper. This is removed and again soldering is done.
- Thus is this way all the components are soldered on P. C. B.

Relay Module Soldering

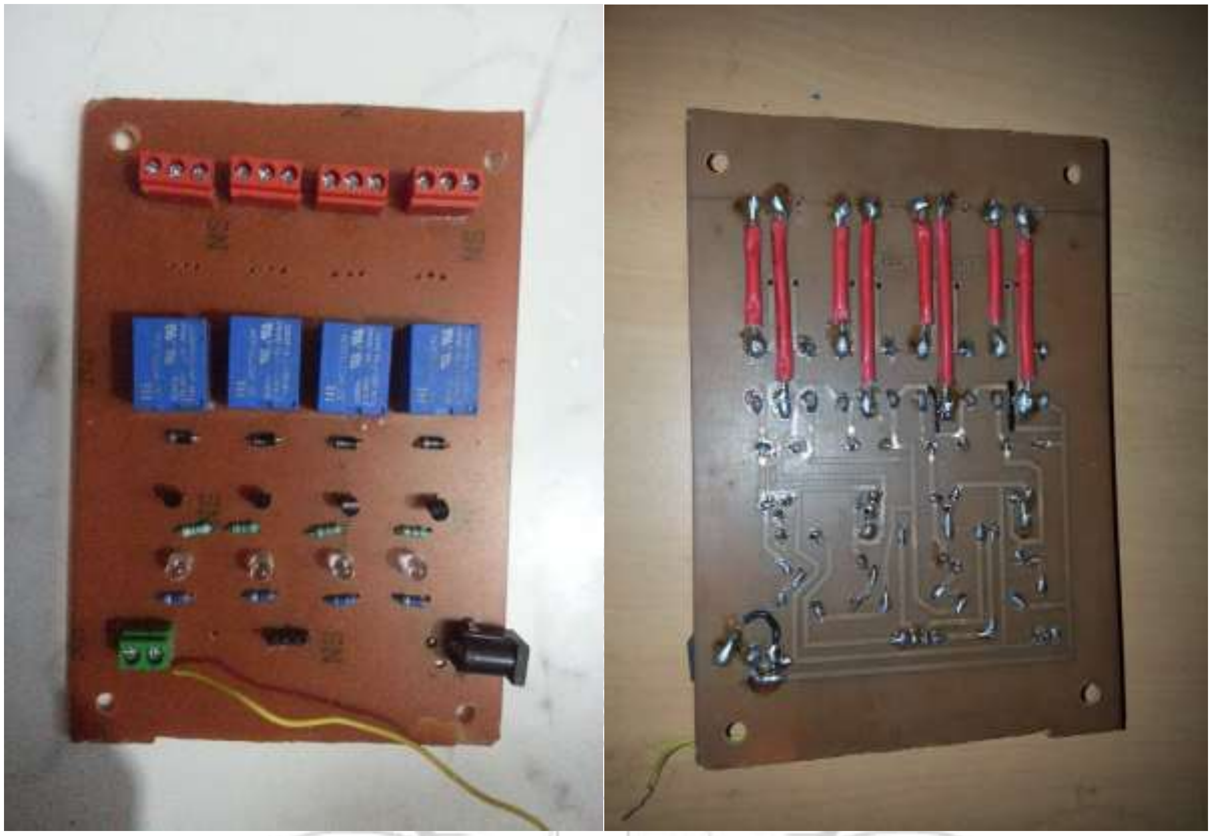


Fig 3.1.3.5 Soldering

3.2 Software specification

3.2.1 Arduino Development Environment

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

3.2.1.1 Writing Sketches

Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the IDE prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify

Checks your code for errors.

Upload

Compiles your code and uploads it to the Arduino I/O board. See uploading below for details.

Note: If you are using an external programmer, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save

Saves your sketch.

Serial

Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive which means only those items relevant to the work currently being carried out are available.

Edit

Copy for Forum

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

Copy as HTML

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

Sketch

Verify/Compile

Checks your sketch for errors.

Show Sketch Folder

Opens the current sketch folder.

Add File...

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu.

Import Library

Adds a library to your sketch by inserting #include statements at the code of your code.

Tools

Auto Format

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements instead curly braces are indented more.

Archive Sketch

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Board

Select the board that you're using.

Serial Port

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Programmer

For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

Burn Bootloader

The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader.

Sketchbook

The Arduino environment uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino

software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.
'"Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the Tools >
Board and Tools > Serial Portmenus. The boards are described below. On the Mac, the serial port is probably something like/dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board),
or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board)
or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyUSB0, /dev/ttyUSB1 or similar.
Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino environment will display a message when the upload is complete, or show an error.
When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it

starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #includestatements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its#include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources. To install these third-party libraries, create a directory called libraries within your sketchbook directory. Then unzip the library there. For example, to install the DateTime library, its files should be in the /libraries/DateTime sub-folder of your sketchbook folder.

3.2.2 Web Server

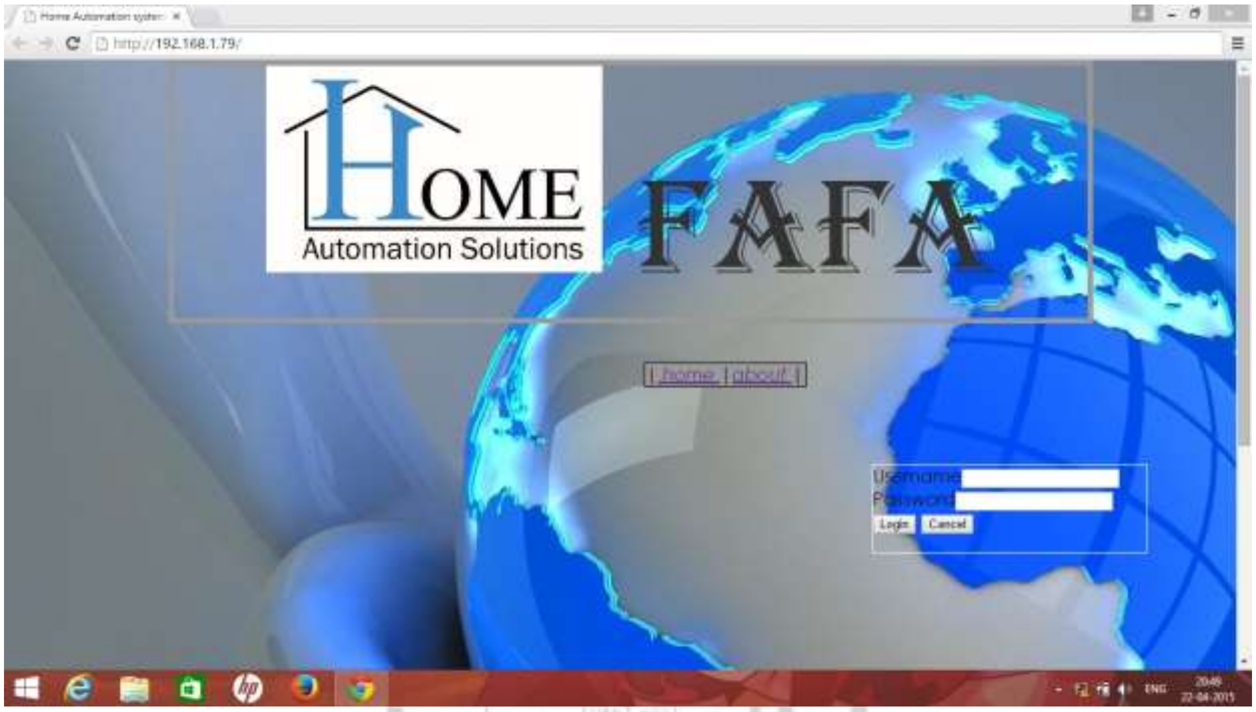
The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.

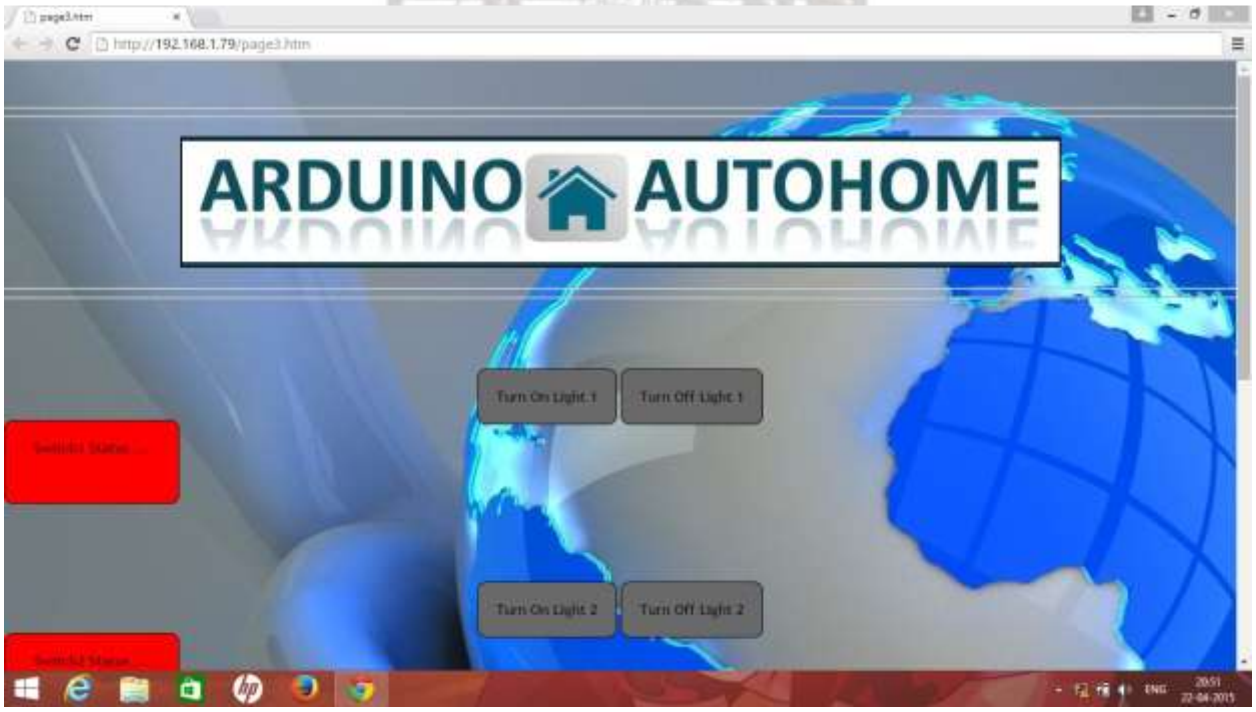
A user agent, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary storage, but this is not necessarily the case and depends on how the web server is implemented.While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Many generic web servers also support server-side scripting using Active Server Pages (ASP), PHP, or other scripting languages. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to create HTML documents dynamically ("on-the-fly") as opposed to returning static documents. The former is primarily used for retrieving and/or modifying information from databases. The latter is typically much faster and more easily cached but cannot deliver dynamic content.

Web servers are not always used for serving the World Wide Web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administering the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

3.2.2.1 Website Page Design





3.2.2.1 Website Page Design

3.2.3 IP Addressing

An IP address is an identifier for a computer or device on a TCP/IP network. Networks using the TCP/IP protocol route messages based on the IP address of the destination.

How to find IP Address?

To view your IP address you can use the ipconfig (IPCONFIG) command line tool. Ipconfig displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.

To launch the command prompt from a Windows-based computer click: Start > All Programs > Accessories > Command Prompt. Type ipconfig and hit enter.

You can also use Google search to find your IP address. Type what is my IP address as a search query and Google will show the IP address of the computer from which the query was received as the top search result.

The Format of an IP Address

The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Each number can be zero to 255. For example, 1.160.10.240 could be an IP address.

Within an isolated network, you can assign IP addresses at random as long as each one is unique. However, connecting a private network to the Internet requires using registered IP addresses (called Internet addresses) to avoid duplicates.

An IP address can be static or dynamic. A static IP address will never change and it is a permanent Internet address. A dynamic IP address is a temporary address that is assigned each time a computer or device accesses the Internet.

Editor's recommendation: Understanding IP Addressing.

The four numbers in an IP address are used in different ways to identify a particular network and a host on that network. Four regional Internet registries -- ARIN, RIPE NCC, LACNIC and APNIC-- assign Internet addresses from the following three classes:

Class A - supports 16 million hosts on each of 126 networks

Class B - supports 65,000 hosts on each of 16,000 networks

Class C - supports 254 hosts on each of 2 million networks

The number of unassigned Internet addresses is running out, so a new classless scheme called CIDR is gradually replacing the system based on classes A, B, and C and is tied to adoption of IPv6. In IPv6 the IP address size is increased from 32 bits to 128 bits.

3.2.4 HTTP

Short for HyperText Transfer Protocol, HTTP is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.

HTTP: A Stateless Protocol

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies.

Introduction to Web Analytics for e-Commerce

Download Now

HTTP Status Codes

Errors on the Internet can be quite frustrating — especially if you do not know the difference between a 404 error and a 502 error. These error messages, also called HTTP status codes are response codes given by Web servers and help identify the cause of the problem.

For example, "404 File Not Found" is a common HTTP status code. It means the Web server cannot find the file you requested. The file -- the webpage or other document you try to load in your Web browser -- has either been moved or deleted, or you entered the wrong URL or document name.

Knowing the meaning of the HTTP status code can help you figure out what went wrong. On a 404 error, for example, you could look at the URL to see if a word looks misspelled, then correct it and try it again. If that doesn't work backtrack by deleting information between each backslash, until you come to a page on that site that isn't a 404. From there you may be able to find the page you're looking for.

3.2.5 HTML

A web browser can read HTML files and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses them to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML.

HTML5 is the newest hyper textmarkup language for websites from the World Wide Web Consortium (W3C). The first draft was made public in 2008, but not much happened until 2011. In 2011, HTML5 was released and people started writing about it and using it, but the support in different browsers was still poor. Today all major browsers (Chrome, Safari, Firefox, Opera, IE) offer HTML5 support, therefore the newest HTML technology can be used at its best today.

HTML5 works with CSS3 and is still in development. W3C plans to release a stable version next year, but it still looks like this is a long shot. Since its release, HTML5 has been in continuous development, with the W3C adding more and more impressive features, therefore it seems quite unlikely that HTML5's development will end soon, which is not necessarily a bad thing.

HTML5 is the successor of HTML 4.01, released for the first time in 1999. The internet has changed significantly since 1999 and it seemed like the creation of HTML5 was necessary. The new markup language was developed based on pre-set standards:

New features should be based on HTML, CSS, DOM, and JavaScript.

- The need for external plugins (like Flash) needs to be reduced.
- Error handling should be easier than in previous versions.
- Scripting has to be replaced by more markup.
- HTML5 should be device-independent.
- The development process should be visible to the public.

3.2.6 CSS

HTML was originally designed as a simple way of presenting information, with the aesthetics of a web page being far less important than the content (and largely being left up to the web browser). Of course, now that the web has become as popular as it has, the presentation of your content has become almost critical to a site’s success. CSS is the key presentational technology that is used to design websites.
What are Stylesheets?

In the late ‘90s, HTML coders noticed that they were retyping the same old tags again and again on the same page, leading to bigger HTML files and above all, time consumption and frustration. You may have found yourself in the same situation, adding in mountains of <font> tags, despite wanting them all the same; or using tricks like invisible gifs for spacing.

Then, someone had a great idea: have one file that defines all the values that those piles of tags would have done, and then have all your pages checking this file and formatting your pages accordingly. You can therefore leave out most of the formatting tags in HTML and use only nice structural elements (like headings, paragraphs and links) — separating structure and presentation.

In late 1996 CSS (Cascading StyleSheets) became a reality, forged by our good friends the » World Wide Web Consortium (W3C). Your stylesheet acts as a partner to your HTML code; taking care of all the layout, fonts, colours and overall look of your site.

If you ever decide to change the look of your site, you modify that one CSS file (your style sheet) and all the HTML pages reading from that file will display differently. This makes maintenance of your design much easier.

3.2.7 Internet Time

Internet time was a common catchphrase that originated during the late1990s Internet boom.[1] In this period, people who worked with the Internet had come to believe that "everything moved faster on the 'net", because the Internet made the dissemination of information far easier and cheaper. In layman's terms, "Internet Time" involves efficiencies inherent to digital transactions that are produced by the virtual reality of one product, one product type, or one service provided to consumers from one virtual cash register residing on one server. The amount of time required to conduct simultaneous transactions is reduced to irrelevance, as everything is occurring at one compressed—albeit virtual—location.

Fast-moving developments were therefore said to run "on Internet time." For example:

Companies released new (usually unstable and buggy) revisions of their software as free downloads, counting on feedback from customers to provide quality assurance. This development strategy, called "release early, release often", was perhaps epitomized in the development of the Netscape Navigator Web browser. The resulting pressure to release new features quickly and grab "mindshare" before one's competitors had disastrous effects on software quality, but resulted in an unprecedented rapid pace of innovation.[1]

A meme could travel the world, in the form of forwarded email, in a week or frequently less. Early instances of such memes included the infamous make money fast spam.

Worms, viruses, and other malware could infect large portions of the Internet in a matter of days or hours, crippling systems worldwide with speed that was shocking to system administrators accustomed to a less networked era.

The meaning (and historical origin) of the phrase "Internet time" strongly parallels that of "New York minute".

3.2.8 Log file

In computing, a **logfile** is a file that records either events that occur in an operating system or other software runs, or messages between different users of a communication software. **Logging** is the act of keeping a log. In the simplest case, messages are written to a single logfile.

Many operating systems, software frameworks, and programs include a logging system. A widely used logging standard is syslog, defined in Internet Engineering Task Force (IETF) RFC 5424). The syslog standard enables a dedicated, standardized subsystem to generate, filter, record, and analyze log messages. This relieves software developers of having to design and code their own ad hoc logging systems.

Event logs

Event logs record events taking place in the execution of a system in order to provide an audit trail that can be used to understand the activity of the system and to diagnose problems. They are essential to understand the activities of complex systems, particularly in the case of applications with little user interaction (such as server applications).

It can also be useful to combine log file entries from multiple sources. This approach, in combination with statistical analysis, may yield correlations between seemingly unrelated events on different servers. Other solutions employ network-wide querying and reporting.

Transaction logs

Most database systems maintain some kind of **transaction log**, which are not mainly intended as an audit trail for later analysis, and are not intended to be human-readable. These logs record changes to the stored data to allow the database to recover from crashes or other data errors and maintain the stored data in a consistent state. Thus, database systems usually have both general event logs and transaction logs.

Message logs

Internet Relay Chat (IRC), instant messaging (IM) programs, peer-to-peer file sharing clients with chat functions, and multiplayer games (especially MMORPGs) commonly have the ability to automatically log (i.e. save) textual communication, both public (IRC channel/IM conference/MMO public/party chat messages) and private chat messages between users.

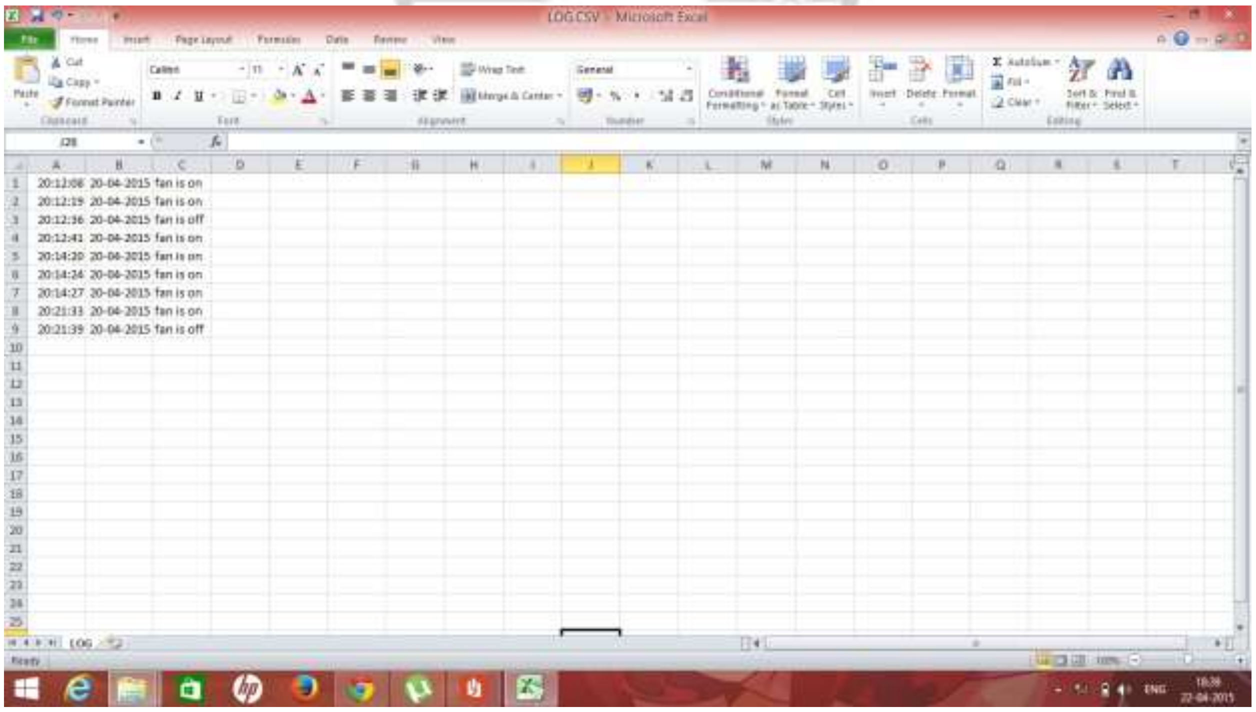
Message logs are almost universally plain text files, but IM and VoIP clients (which supports textual chat, e.g. Skype) might save them in HTML files or in a custom format to ease reading and encryption.

Internet Relay Chat (IRC)

In the case of IRC software, message logs often include system/server messages and entries related to channel and user changes (e.g. topic change, user joins/exits/kicks/bans, nickname changes, user status changes), making them more like a combined message/event log of the channel in question, but such a log isn't comparable to a true IRC server event log, because it only records user-visible events for the time frame the user spent being connected to a certain channel.

Instant Messaging (IM)

Instant messaging and VoIP clients often offer the chance to store encrypted logs to enhance the user's privacy. These logs require a password to be decrypted and viewed, and they are often handled by their respective writing application.



	A	B	C
1	20:12:06	20-04-2015	fan is on
2	20:12:19	20-04-2015	fan is on
3	20:12:36	20-04-2015	fan is off
4	20:12:41	20-04-2015	fan is on
5	20:14:20	20-04-2015	fan is on
6	20:14:24	20-04-2015	fan is on
7	20:14:27	20-04-2015	fan is on
8	20:21:33	20-04-2015	fan is on
9	20:21:39	20-04-2015	fan is off

Fig 3.2.8 Log File

Chapter 4

ADVANTAGES, DRAWBACKS AND APPLICATIONS

4.1 Advantages

- Energy saving.
- Security management.
- Home entertainment.
- Convenience Access through Internet.
- Adds Safety Through Appliance and Lighting Control.
- Secures Home Through Automated Door Locks.
- Increases Awareness Through Security Cameras.
- Increases Convenience Through Temperature Adjustment.

4.2 Drawbacks

- Internet is necessary.
- User must be acquainted with internet services.

4.3 Applications

- Quickly and easily open and close curtains.
- Cameras to monitor your home to See who is at the front gate or front door.
- Heating and cooling.
- Adjust the lighting scenes yourself without calling the integrator for help.
- We Can turn ON & OFF any appliance from anywere using internet.

Chapter 5

FUTURE SCOPES

- Doors & Gates can be controlled using proper mechanism.
- Using infrared light we can control Air Conditioner.
- Using CCTV camera we can monitor indoor & outdoor areas surround house.
- Using internet timing we can turn ON & OFF any appliance during any selected hours.
- Using image recognition,finger print,ratina scan etc methods only authorized person is permitted to enter the house.

Chapter 6
CONCLUSION

Home automation is not a new industry anymore, but it is still an emerging industry in developing countries. The huge potential market leads many electronics corporation into home automation. Sony, Siemens and even Apple company are trying to share a market of home automation. But the home automation industry scale hasn't formed yet. The good and bad mixed products are together in the market. There are also no unified industry standards. It is the challenge of home automation.

When compared with todays internet world, we can find that home automation has very big chance in the future. As, now a days every one uses internet it became very easy to work on product or device which can control by internet.

If the price of home automation is decreased or the practical applicability increased, there will be more people willing to buy it. So in our project we tried to achieve this purpose. As we use open source software the cost of product is low. Using open source software also means you are not locked in to using a particular vendor's system that only work with their other systems. It continually evolving in real time as developers add to it and modify it, which means it can be better quality and more secure and less prone to bugs than proprietary systems.

Home Automation is undeniably a resource which can make a home environment automated. People can control their electrical devices via this Home Automation product and set up the controlling actions in the computer or smart phones with simple internet facility. We think this product have high potential for marketing in the future.

At last, we have to say home automation is like a rising sun. One day it will be like a burning sun bringing more decent, enjoyable and efficient life to people.

Chapter 7
APPENDIX

7.1 Programs

7.1.1 HTML Page

```
<!DOCTYPE HTML  "web server home automation">

<html lang="en">
<head>
<title>Home Automation system</title>
<style type= "text/css">
body{
                font-family: century gothic, arial;
                margin: 0 auto;
               background-
image:url('http://cdn.wonderfulengineering.com/wp-
content/uploads/2014/07/background-wallpapers-27-798x350.jpg')
                }
.header{
       margin: 0 100px 0 180px;
       text-align: center;
       width: 1000px;
       border: 6px solid #999999;
       background-colour: #7BF96D;
       align:left;
       font-size: 150px;
       font-family: "algerian";
       color:#333333;
       font-style:bold;
}
.s{
       margin: 0 300px 0 700px;
       width 40px;
       border: 2px solid #444444;
       background-color: #6fffff;
       font-size: 20px;
       font-style:italic;
       align: left
}
.s1{
       margin: 0 150px 0 950px;
       width: 300px;
       background color: #555555;
       font-size: 20px;
       border: 1px solid #ffffff;
       align: center
}
```

```
</style>
<meta name='viewport' content='width-devicewidth, initial-scale=1.0'>
<meta charset='UTF-8'>

</head>
<body>
<div class='header'><img
src='http://homeautomationsolutions.com.au/images/HAS%20Logo.jpg'>  FAFA
</div>
<br>
<br>
<div class='s'>     |<a href='http://www.facebook.com'> home </a> | menu |
about | switch |
</div>
<br>
<br>
<br>
<br>
<div class='s1'>
<form name="login">
            Username<input type="text" name="userid"/>
            Password<input type="password" name="pswrd"/>
            <input type="button" onclick="check(this.form)"
value="Login"/>
            <input type="reset" value="Cancel"/>
        </form>
</div>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br><br>
<br>
<br>
<br><br>
<br>
<br>
<br><br>
<br>
<br>
<br><br>
<br>
<br>
<br>
<script language="javascript">
            function check(form) { /*function to check userid & password*/
                /*the following code checkes whether the entered userid
and password are matching*/
                if(form.userid.value == "u1" && form.pswrd.value == "v1")
{
```

```
                    self.location.href = 'page3.htm';/*opens the target
page while Id & password matches*/
                }
                else {
                    alert("Error Password or Username")/*displays error
message*/
                }
            }
        </script>
</body>
</html>
</form>
```

7.1.2 Main Program

```
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <EthernetUdp.h>
#include <Time.h>
// size of buffer used to capture HTTP requests
#define REQ_BUF_SZ   20
// MAC address from Ethernet shield sticker under board
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };


/* ******** NTP Server Settings ******** */
/* us.pool.ntp.org NTP server
   (Set to your time server of choice) */
IPAddress timeServer(123,108,200,163);
/* Set this to the offset (in seconds) to your local time
   This example is GMT - 4 */
const long timeZoneOffset = +19800L;
/* Syncs to NTP server every 15 seconds for testing,
   set to 1 hour or more to be reasonable */
unsigned int ntpSyncTime = 86400;
/* ALTER THESE VARIABLES AT YOUR OWN RISK */
// local port to listen for UDP packets
unsigned int localPort = 8888;
// NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE= 48;
// Buffer to hold incoming and outgoing packets
byte packetBuffer[NTP_PACKET_SIZE];
// A UDP instance to let us send and receive packets over UDP
EthernetUDP Udp;
// Keeps track of how long ago we updated the NTP server
unsigned long ntpLastUpdate = 0;
// Check last time clock displayed (Not in Production)
time_t prevDisplay = 0;
```

```
String timedateString = "";
String timedate1String = "";
String dataString = "";
EthernetServer server(80);      // create a server at port 80
File webFile;                   // handle to files on SD card
char HTTP_req[REQ_BUF_SZ] = {0}; // buffered HTTP request stored as null terminated string
char req_index = 0;             // index into HTTP_req buffer
String readString;
void setup()
{
   // disable Ethernet chip
   pinMode(10, OUTPUT);
   digitalWrite(10, HIGH);
   Serial.begin(9600);       // for debugging
//Write Log File Header
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
   logFile.println(", , "); //Just a leading blank line, incase there was previous data
   String header = "time,date, status ";
   logFile.println(header);
   logFile.close();
   Serial.println(header);
  }
  else
  {
   Serial.println("Couldn't open log file");
  }
   // initialize SD card
  Serial.println("Initializing SD card...");
  if (!SD.begin(4)) {
      Serial.println("ERROR - SD card initialization failed!");
     return;    // init failed
   }
   Serial.println("SUCCESS - SD card initialized.");
   // check for index.htm file
  if (!SD.exists("fafa22.htm")) {
      Serial.println("ERROR - Can't find fafa22.htm file!");
      return;  // can't find index file
  }
   Serial.println("SUCCESS - Found fafa22.htm file.");
   EthernetClient client;
  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
   Serial.println("Failed to configure Ethernet using DHCP");
   // no point in carrying on, so do nothing forevermore:
   for(;;)
     ;
```

```
 }
 // print your local IP address:
 Serial.print("My IP address: ");
 for (byte thisByte = 0; thisByte < 4; thisByte++) {
  // print the value of each byte of the IP address:
  Serial.print(Ethernet.localIP()[thisByte], DEC);
  Serial.print(".");
 }
 Serial.println();
  //Try to get the date and time
  int trys=0;
  while(!getTimeAndDate() && trys<10) {
    trys++;
  }
   Ethernet.begin(mac);  // initialize Ethernet device
   server.begin();          // start to listen for clients
 pinMode(40, OUTPUT); // Pin Assignment through which relay will be controlled
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 }
  void loop()
 {
//time
 if(now()-ntpLastUpdate > ntpSyncTime) {
    int trys=0;
    while(!getTimeAndDate() && trys<10){
      trys++;
    }
    if(trys<10){
      Serial.println("ntp server update success");
    }
    else{
      Serial.println("ntp server update failed");
    }
  }
  //time
 // make a string for assembling the data to log:
if (timeStatus() != timeNotSet) {
if (now() != prevDisplay) {//update the display only if time has changed
prevDisplay = now();
Get_And_Convert_TimeStamp();
dataString = timedateString;
 dataString += ",";
dataString += timedate1String;
 dataString += ",";
}
}
```

```
EthernetClient client = server.available();  // try to get client
if (client) {  // got client?
    boolean currentLineIsBlank = true;
    while (client.connected()) {
        if (client.available()) {   // client data available to read
            char c = client.read(); // read 1 byte (character) from client
            // buffer first part of HTTP request in HTTP_req array (string)
            // leave last element in array as 0 to null terminate string (REQ_BUF_SZ - 1)
            if (req_index < (REQ_BUF_SZ - 1)) {
                HTTP_req[req_index] = c;          // save HTTP request character
                req_index++;
            }
                  if (readString.length() < 100) {
      //store characters to string
      readString += c;
      //Serial.print(c);
    }
            Serial.print(c);    // print HTTP request character to serial monitor
            // last line of client request is blank and ends with \n
            // respond to client only after last line received
           if (c == '\n' && currentLineIsBlank) {
                // send a standard http response header
                 client.println("HTTP/1.1 200 OK");
                 // remainder of header follows below, depending on if
                // web page or XML page is requested
                // Ajax request - send XML file
                if (StrContains(HTTP_req, "ajax_switch")) {
                    // send rest of HTTP header
                     client.println("Content-Type: text/xml");
                  client.println("Connection: keep-alive");
                    client.println()
                // send XML file containing input states
                    GetSwitchState(client);
                }
                else {  // web page request
                    // send rest of HTTP header
            if (StrContains(HTTP_req, "GET / ") || StrContains(HTTP_req, "GET /index.htm"))
             {
          client.println("Content-Type: text/html");
          client.println("Connection: keep-alive");
          client.println();
                           webFile = SD.open("fafa22.htm");       // open web page file
                }
                   else if (StrContains(HTTP_req, "GET /page3.htm")) {
                              webFile = SD.open("page3.htm");       // open web page file
                         }
                    // send web page to client
                      if (webFile) {
```

```
                while(webFile.available()) {
                  client.write(webFile.read());
                }
                webFile.close();
              }
              }
        if(readString.indexOf("?relay1on") >0)//checks for on
              {
       digitalWrite(40, HIGH); // set pin 4 high
       Serial.println("Led On");
       getLog();
              }
     if(readString.indexOf("?relay1off") >0)//checks for off
              {
    digitalWrite(40, LOW); // set pin 4 low
   Serial.println("Led Off");
   getLog();
     }
   if(readString.indexOf("?relay2on") >0)//checks for on
    {
   digitalWrite(7, HIGH); // set pin 4 high
   Serial.println("Led On");
   getLog();
     }
    if(readString.indexOf("?relay2off") >0)//checks for off
     {
    digitalWrite(7, LOW); // set pin 4 low
     Serial.println("Led Off");
     getLog();
       }
    if(readString.indexOf("?relay3on") >0)//checks for on
      {
    digitalWrite(8, HIGH); // set pin 4 high
    Serial.println("Led On");
    getLog();
       }
    if(readString.indexOf("?relay3off") >0)//checks for off
       {
    digitalWrite(8, LOW); // set pin 4 low
    Serial.println("Led Off");
    getLog();
       }
     if(readString.indexOf("?relay4on") >0)//checks for on
       {
     digitalWrite(9, HIGH); // set pin 4 high
     Serial.println("Led On");
     getLog();
       }
```

```
        if(readString.indexOf("?relay4off") >0)//checks for off
          {
        digitalWrite(9, LOW); // set pin 4 low
        Serial.println("Led Off");
        getLog();
           }
        readString="";
        // reset buffer index and all buffer elements to 0
         req_index = 0;
          StrClear(HTTP_req, REQ_BUF_SZ);
          break;
                }
                // every line of text received from the client ends with \r\n
                if (c == '\n') {
                    // last character on line of received text
                    // starting new line with next character read
                    currentLineIsBlank = true;
                }
                else if (c != '\r') {
                    // a text character was received from client
                    currentLineIsBlank = false;
                }
            } // end if (client.available())
         } // end while (client.connected())
         delay(1);      // give the web browser time to receive the data
         client.stop(); // close the connection
    } // end if (client)
}
// send the state of the switch to the web browser
 void GetSwitchState(EthernetClient cl)
 {
    cl.print("<?xml version = \"1.0\" ?>");
    cl.print("<inputs>");
    cl.print("<button1>");
    if (digitalRead(40)) {
        cl.print("ON");
    }
    else {
        cl.print("OFF");
    }
    cl.print("</button1>");
    cl.print("<button2>");
    if (digitalRead(7)) {
        cl.print("ON");
    }
    else {
        cl.print("OFF");
    }
    cl.print("</button2>");
    cl.print("<button3>");
    if (digitalRead(8)) {
      cl.print("ON");
    }
    else {
      cl.print("OFF");
    }
    cl.print("</button3>");
    cl.print("<button4>");
    if (digitalRead(9)) {
      cl.print("ON");
    }
    else {
      cl.print("OFF");
    }
    cl.print("</button4>");
     cl.print("</inputs>");
     }
  // sets every element of str to 0 (clears array)
  // sets every element of str to 0 (clears array)
  void StrClear(char *str, char length)
  {
    for (int i = 0; i < length; i++) {
      str[i] = 0;
    }
  }
  // searches for the string sfind in the string str
  // returns 1 if string found
  // returns 0 if string not found
  char StrContains(char *str, char *sfind)
  {
    char found = 0;
    char index = 0;
    char len;
    len = strlen(str);
      if (strlen(sfind) > len) {
      return 0;
    }
    while (index < len) {
      if (str[index] == sfind[found]) {
        found++;
        if (strlen(sfind) == found) {
          return 1;
        }
      }
      else {
        found = 0;
        }
        index++;
    }
return 0;
    }
// Do not alter this function, it is used by the system
int getTimeAndDate() {
   int flag=0;
   Udp.begin(localPort);
   sendNTPpacket(timeServer);
   delay(1000);
   if (Udp.parsePacket()){
     Udp.read(packetBuffer,NTP_PACKET_SIZE);  // read the packet into the buffer
     unsigned long highWord, lowWord, epoch;
     highWord = word(packetBuffer[40], packetBuffer[41]);
     lowWord = word(packetBuffer[42], packetBuffer[43]);
     epoch = highWord << 16 | lowWord;
     epoch = epoch - 2208988800  + timeZoneOffset;
     flag=1;
     setTime(epoch);
     ntpLastUpdate = now();
      }
   return flag;
      }
// Do not alter this function, it is used by the system
unsigned long sendNTPpacket(IPAddress& address)
{
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;
  Udp.beginPacket(address, 123);
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();
      }
void Get_And_Convert_TimeStamp() {
// digital clock display of the time
timedateString = "";
timedateString += hour();
printDigits(minute());
printDigits(second());
timedateString += " ";
timedate1String = "";
```

```
timedate1String += day();
timedate1String += "/";
timedate1String += month();
timedate1String += "/";
timedate1String += year();
timedate1String += " ";
}
void printDigits(int digits){
// utility for digital clock display: prints preceding colon and leading 0
timedateString += ':';
if(digits < 10)
timedateString += '0';
timedateString += digits;
}
void getLog()
{
  if(readString.indexOf("?relay1on") >0)
     {
  if (digitalRead(40))
  {
   File logFile = SD.open("LOG.csv", FILE_WRITE);
   if (logFile)
   {
     dataString += "fan is on";
     logFile.println(dataString);
     logFile.close();
     Serial.println(dataString);
   }
   else
   {
     Serial.println("Couldn't open log file");
   }
   }
  else
  {
   File logFile = SD.open("LOG.csv", FILE_WRITE);
   if (logFile)
   {
      dataString += "---";
     logFile.println(dataString);
     logFile.close();
     Serial.println(dataString);
   }
   else
   {
   Serial.println("Couldn't open log file");
   }
   }
```

```
    }
    if(readString.indexOf("?relay1off") >0)
    {
   if (digitalRead(40))
    {
    File logFile = SD.open("LOG.csv", FILE_WRITE);
    if (logFile)
    {
    dataString += "---";
      logFile.println(dataString);
      logFile.close();
      Serial.println(dataString);
      }
    else
     {
      Serial.println("Couldn't open log file");
     }
     }
  else
    {
    File logFile = SD.open("LOG.csv", FILE_WRITE);
    if (logFile)
    {
       dataString +="fan is off";
      logFile.println(dataString);
      logFile.close();
      Serial.println(dataString);
    }
    else
    {
      Serial.println("Couldn't open log file");
    }
    }
    }
if(readString.indexOf("?relay2on") >0)
     {
  if (digitalRead(7))
     {
    File logFile = SD.open("LOG.csv", FILE_WRITE);
    if (logFile)
     {
       dataString += "tube light is on";
      logFile.println(dataString);
      logFile.close();
      Serial.println(dataString);
      }
    else
      {
```

```
     Serial.println("Couldn't open log file");
     }
     }
 else
   {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
     dataString += "---";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
   }
  else
  {
   Serial.println("Couldn't open log file");
  }
  }
  }
   if(readString.indexOf("?relay2off") >0)
  {
  if (digitalRead(7))
  {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
    dataString += "---";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
  }
  else
  {
    Serial.println("Couldn't open log file");
  }
  }
 else
 {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
    dataString +=  "tube light is off";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
  }
  else
  {
```

```
  Serial.println("Couldn't open log file");
 }
 }
 }
  if(readString.indexOf("?relay3on") >0)
 {
  if (digitalRead(8))
 {
 File logFile = SD.open("LOG.csv", FILE_WRITE);
 if (logFile)
 {
  dataString += "switch is on";
  logFile.println(dataString);
  logFile.close();
  Serial.println(dataString);
  }
 else
 {
 Serial.println("Couldn't open log file");
 }
 }
else
 {
 File logFile = SD.open("LOG.csv", FILE_WRITE);
 if (logFile)
 {
   dataString += "---";
  logFile.println(dataString);
  logFile.close();
  Serial.println(dataString);
  }
 else
  {
  Serial.println("Couldn't open log file");
  }
  }
  }
  if(readString.indexOf("?relay3off") >0)
  {
  if (digitalRead(8))
  {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
  dataString += "---";
  logFile.println(dataString);
  logFile.close();
  Serial.println(dataString);
  }
  else
  {
    Serial.println("Couldn't open log file");
  }
  }
  else
  {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
    dataString += "switch is off";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
  }
  else
  {
   Serial.println("Couldn't open log file");
  }
  }
 }
 if(readString.indexOf("?relay4on") >0)
 {
if (digitalRead(9))
 {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
     dataString += "switch2 is on";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
  }
  else
  {
    Serial.println("Couldn't open log file");
  }
  }
else
 {
  File logFile = SD.open("LOG.csv", FILE_WRITE);
  if (logFile)
  {
    dataString += "---";
    logFile.println(dataString);
    logFile.close();
    Serial.println(dataString);
```

```
}
else
{
  Serial.println("Couldn't open log file");
}
}
}
  if(readString.indexOf("?relay4off") >0)
{
  if (digitalRead(9))
{
File logFile = SD.open("LOG.csv", FILE_WRITE);
if (logFile)
{
  dataString += "---";
  logFile.println(dataString);
  logFile.close();
  Serial.println(dataString);
}
else
{
  Serial.println("Couldn't open log file");
}
}
 else
{
File logFile = SD.open("LOG.csv", FILE_WRITE);
if (logFile)
{
  dataString += "switch2 is off";
  logFile.println(dataString);
  logFile.close();
  Serial.println(dataString);
}
else
{
  Serial.println("Couldn't open log file");
}
}
}
}
```

Chapter 8

REFERENCES

(1) Jyotsna A Nanajkar, Vismita D Nagrale "**Embedded Web Server Based Automation**" International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 9, March 2013

.

(2) S.A.N.Sandeep, P.Malyadri "**Embedded Web Server Based on DAC System Using ARM**" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 , Vol. 2, Issue 4, July-August 2012.

(3) Rajeev Piyare "**Internet of Things: Ubiquitous Home Control and Monitoring System using Android based Smart Phone**" Department of Information Electronics Engineering, Mokpo National University, Mokpo, 534-729, Korea South. International Journal of Internet of Things 2013, 2(1): 5-11 DOI: 10.5923/j.ijit.20130201.02

(4) Ahmed ElShafee, Karim Alaa Hamed "**Design and Implementation of a WiFi Based Home Automation System**" World Academy of Science, Engineering and Technology 68 2012.

(5) Malik Sikandar Hayat Khiyal, Aihab Khan, and Erum Shehzadi, "**SMS Based Wireless Home Appliance Control System (HACS) for Automating Appliances and Security**", Issues in Informing Science and Information Technology Volume 6, 2009

(6) Wikipedia. (2012, 12th December). **Home automation. Available**: http://en.wikipedia.org/wiki/Home_automation

(7) Faisal Baig, Saira Beg and Muhammad Fahad Khan "**Zigbee Based Home Appliances Controlling Through Spoken Commands Using Handheld Devices**" International Journal of Smart Home Vol. 7, No. 1, January, 2013

(8) Thoraya Obaid, Haliemah Rashed, Ali Abu El Nour, Muhammad Rehan, Mussab Muhammad Saleh, and Mohammed Tarique "**Zigbee Based Voice Controlled Wireless Smart Home System**" International Journal of Wireless & Mobile Networks (IJWMN) Vol. 6, No. 1, February 2014.