

**A MINI PROJECT REPORT  
ON  
BLUETOOTH CONTROLLED ELECTRONIC HOME APPLIANCES**

submitted in partial fulfillment for the award of the degree  
of

**Bachelor of Technology**

In

**Information Technology**

Submitted by

**E.Lahari(13B81A1226)**

**A.Geetanjali(13B81A1227)**

**T.Naveen Kumar(13B81A1252)**

Under the supervision of

**Mr.P.V.M Seravana Kumar**

**Assistant Professor**

**IT DEPARTMENT**



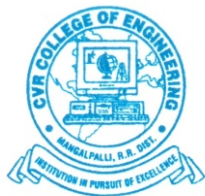
**Department of Information Technology**

**CVR COLLEGE OF ENGINEERING**

**(An Autonomous Institution&Affiliated to JNTUH)**

**Ibrahimpatnam (M), Ranga Reddy (D), Telangana**

**2016-2017**



Cherabuddi Education Society's  
**CVR COLLEGE OF ENGINEERING**

(An Autonomous Institution)

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, AICTE

(Approved by AICTE & Govt. of Telangana and Affiliated to JNT University)

Vastunagar, Mangalpalli (V), Ibrahimpatan (M), R.R. District, PIN - 501 510

Web : <http://cvr.ac.in>, email : [info@cvr.ac.in](mailto:info@cvr.ac.in)

Ph : 08414 - 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

---

## CERTIFICATE

This is to certify that the project titled “**Bluetooth Controlled Electronic Home Appliances**” is a bonafide record of the work done by ***E.Lahari(13B81A1226)***, ***A.Geetanjali(13B81A1227)***, ***T.NaveenKumar(13B81A1252)*** under the guidance and supervision in partial fulfillment of requirement for the award of the degree of **Bachelor of Technology in Information Technology** of JNTU Hyderabad during the academic year 2016-2017.

### Supervisor

Mr. P.V.M Seravana Kumar  
Assistant Professor  
Department of IT.

### Head of the Department

Prof.U.V Ramana Sharma  
Head of the Department  
Professor,IT Department.

### Coordinator

Dr.Bipin Bihari Jaya Singh  
Project Coordinator  
Professor,IT Department.

External Examiner:

## DECLARATION

We hereby declare that the project report entitled “**BLUETOOTH CONTROLLED ELECTRONIC HOME APPLIANCES**” submitted by us to “**CVR College of Engineering**”, in partial fulfillment of the requirements for the award of B.Tech degree. We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Signature of the Student

Lahari(13b81a1226)

Signature of the Student

Geeta(13b81a1227)

Signature of the Student

Naveen(13b81a1252)

## ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

We would like to express our sincere gratitude to our guide, **Mr.P.V.M.Seravana Kumar**, Assistant Professor of **IT Dept, CVR College of Engineering**, whose guidance and valuable suggestions have been indispensable to bring about the successful completion of our project.

It is great pleasure to convey our profound sense of gratitude to our principal **Dr. Nayanathara K.S, Prof. U.V Ramana Sharma**, Head of the IT Department, CVR college of Engineering for having been kind enough for arranging the necessary facilities for executing the project in the college.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

We would also like to express our gratitude to all the staff members and lab faculty, department of Electronics and Communication Engineering, CVR College of Engineering for the constant help and support.

## **ABSTRACT**

Due to the advancement of wireless technology, there are several different of connections are introduced such as GSM, WIFI, ZIGBEE, and Bluetooth. Each of the connection has their own unique specifications and applications. Among the four popular wireless connections, Bluetooth is being chosen with its suitable capability. Bluetooth with globally available frequencies of 2400Hz is able to provide connectivity up to 100 meters at speed of up to 3Mbps depending on the Bluetooth device class. In addition, a Bluetooth master device is able to connect up to 7 devices in a “Piconet”. In terms of cost, this system implemented low cost microcontroller and Bluetooth module as the system main core. With this low budget, this system can still performed with powerful remote functions to make our life in home become easier.

In this project proposal Bluetooth module is interfaced to Arduino microcontroller. This Bluetooth module receives the commands from the Android application device using wireless communication. The program which is written to the Arduino microcontroller communicates with Bluetooth module serially to receive the commands. Microcontroller switches the electrical loads automatically based on the commands received from the Bluetooth.

This project is very much useful for physically handicapped people, who can control home appliances by sitting at one place ( up to certain amount of distance).

# CONTENTS

TITLE	1
CERTIFICATE	2
DECLARATION	3
ACKNOWLEDGEMENT	4
ABSTARCT	5
<b>TABLE OF CONTENTS:</b>	
1.INTRODUCTION	8
1.1 Literature survey	
1.2 Objective	
2. REQUIREMENTS	9
2.1 Arduino History	9
2.2 Why Arduino UNO ?	10
2.3 Overview of Arduino	11
2.4 Block diagram	14
2.5 Hardware components and their uses	15
2.6 Software Components	21
3. DESIGN	22
4.. IMPLEMENTATION	25
4.1 Hardware setup	
4.2 Software setup	

5.TESTING	31
CONCLUSION	36
FUTURE ENHANCEMENT	37
REFERENCES	38
APPENDIX A	39
APPENDIX B	44

# 1.INTRODUCTION

## 1.1 Literature survey

The basic idea is to easily control any electronic home appliance via Bluetooth module. This project is very helpful for the physically handicapped. In this project proposal Bluetooth module is interfaced to Arduino microcontroller. This Bluetooth module receives the commands from the Android application device using wireless communication. The program which is written to the Arduino microcontroller communicates with Bluetooth module serially to receive the commands. Microcontroller switches the electrical loads automatically based on the commands received from the Bluetooth.

## 2. Objective

- This project will give an overview of wireless home network and we can also realize the concept of smart phone that is application development.
- Appliances can communicate with each other on network.
- This project provides secure communication between electronic home appliances and the android app that is communicating.
- Entire project will cost very less.
- At the end of this project we are able to control electronic home appliances using android app that is designed .
- One who is unable (i.e; physically handicapped ) switch on/off the devices can use this project and control the devices in a room by sitting at one place in that room.



## 2.REQUIREMENTS

### ARDUINO

#### 2.1Arduino History

**Arduino**, sold as **Genuino**, due to a trademark dispute, outside the U.S. and U.K., is a hardware and software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller based kits for building digital devices and interactive objects that can sense and control physical devices.

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

The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino boards are available commercially in preassembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone.

This is a non-exhaustive list of Arduino boards and compatible systems. It lists boards in these categories:

- Released under the official Arduino name
- Arduino "shield" compatible
- Development-environment compatible
- Based on non-Atmel processors

Where different from the Arduino base feature set, compatibility, features, and licensing details are included.

Out of many different models we had in this project we used ARDUINO UNO .

## 2.2 Why Arduino UNO?

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

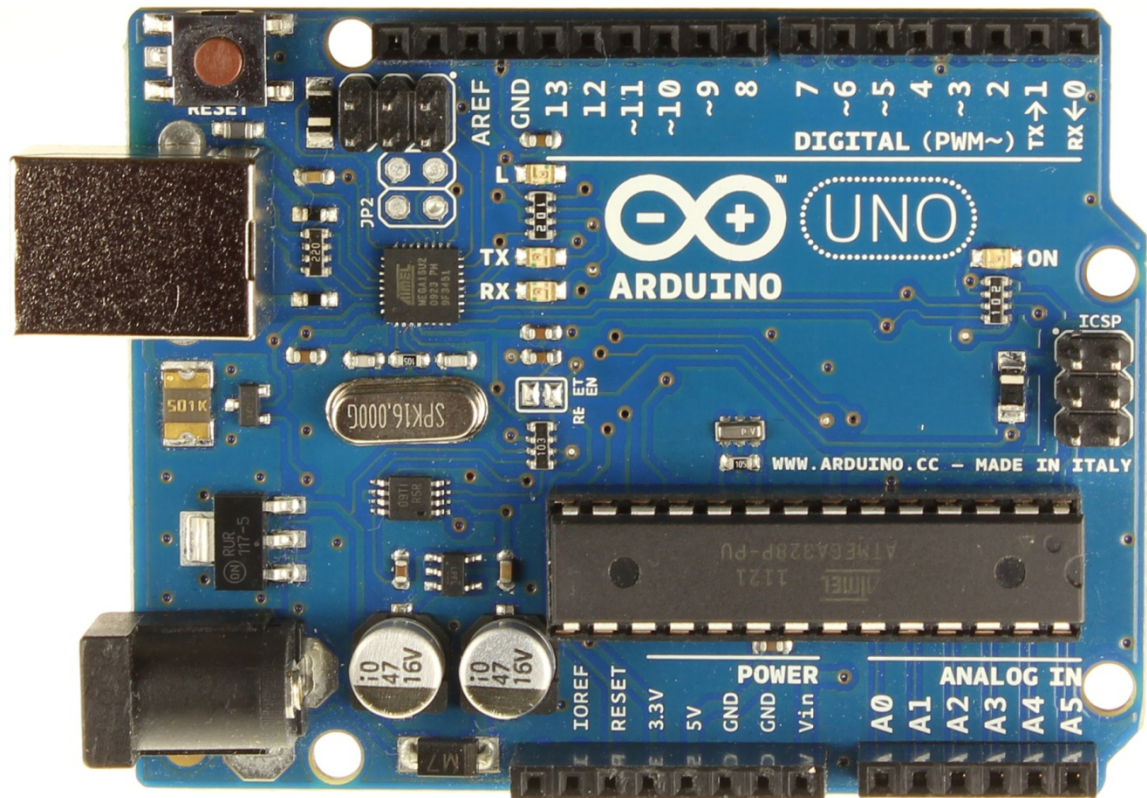
"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model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

When you need more control and are actually thinking on converting your prototype into a real product, then yes, you need to get deep down into the microcontroller and get rid of all the excess fat, trim the circuit to just the bare bones, optimize the code, etc.

For prototyping, the Arduino platform gives you a lot of pre-wiring and free code libraries that will let you concentrate on testing your idea instead of spending your time building supporting circuitry or writing tons of low level code.

The second benefit is that you really understand what's going on. But that's also more work.

## 2.3 Overview of Arduino



Technical Specifications:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

The Arduino/Genuino Uno board 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 from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

## Input and Output

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

In addition, some pins have specialized functions:

**Serial:** pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

**External Interrupts:** pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.

**PWM:** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the `analogWrite()` function.

**SPI:** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

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

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

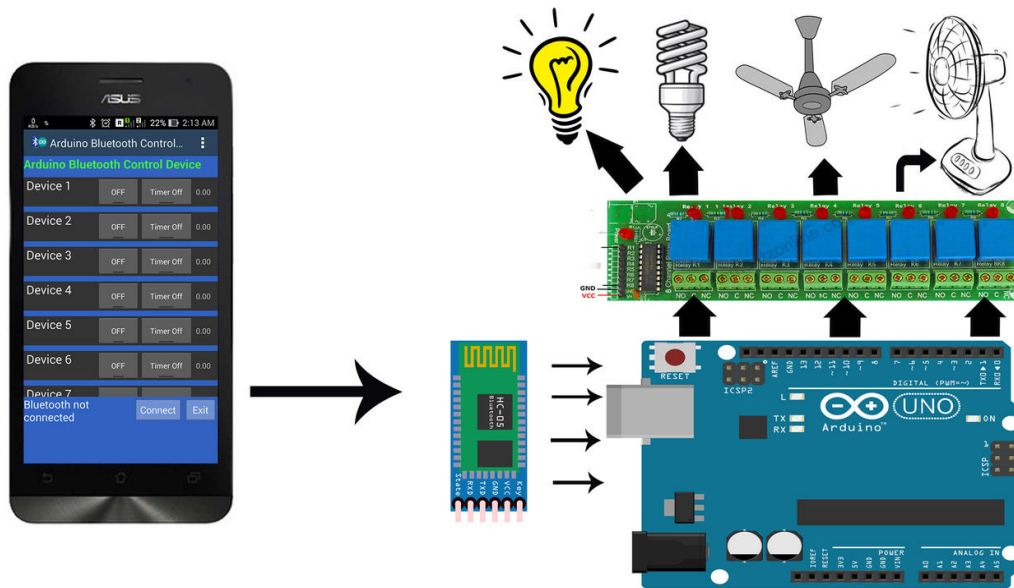
**TWI:** A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

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

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

## **2.4 Block diagram:**



This project is a fine combination of Android mobile technology and embedded system. User can control Home appliances using Android mobile. An application should be installed on his/her Android mobile handset to control various home appliances. User can send commands using that application. Wireless controlling technique used in this project is Bluetooth technology. This project consists of a Bluetooth receiver. This Bluetooth device is connected to the circuit which has a decoder. This decoder sends code for respective command sent by user. Then the respective device connected to the circuit will be turned on or off depending on the command given.

## 2.5 Hardware componens and their uses:

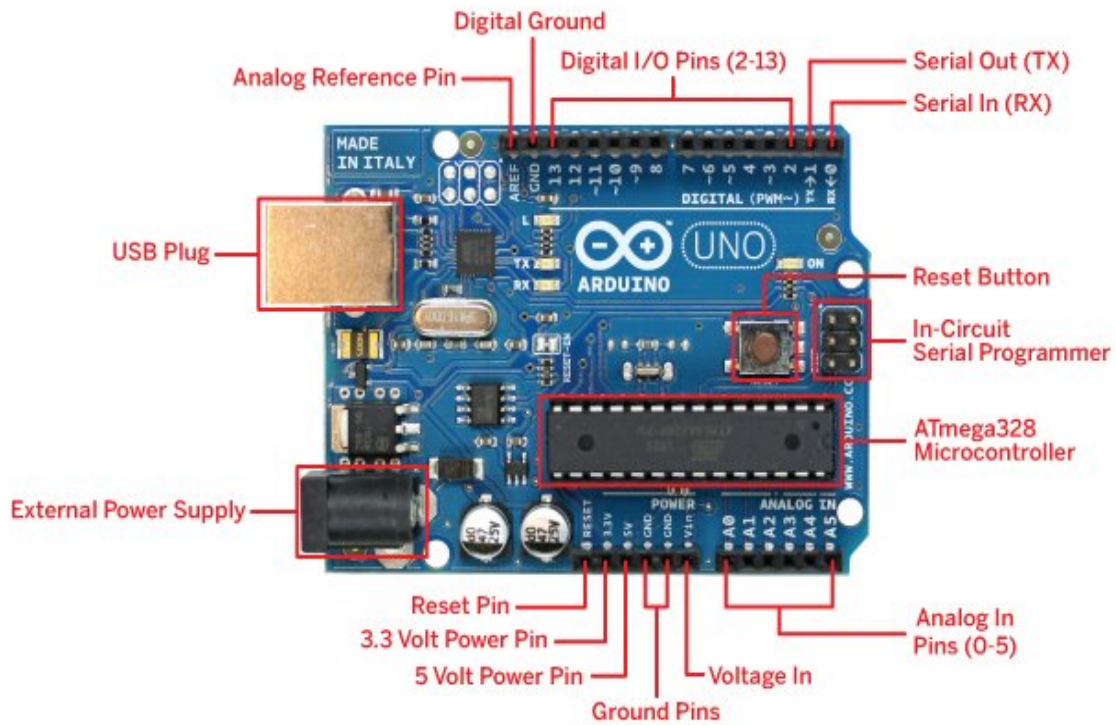
The hardware components used in this project are as follows:

- Arduino UNO
- Transformer, 5V Relays.
- Connecting Wires
- PCB board
- Diodes, Resistors, Capacitors .
- Bluetooth Device.

### Arduino UNO:

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

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



### Transformer, 5V Relays:

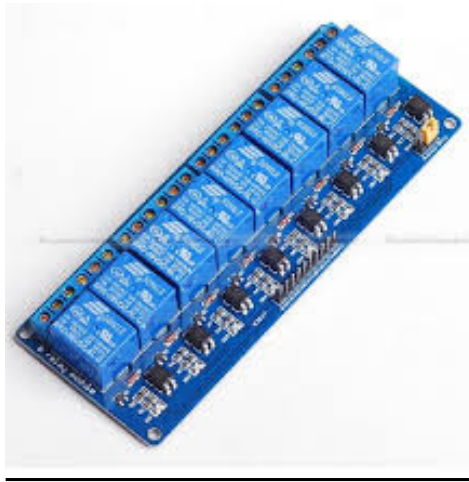
A suitable ready-built mains power supply unit, such as those used to control model trains, will include a transformer.

Considerations when dealing with mains voltages if such a unit does not incorporate smoothing, rectification, and regulation.

This fuse is in addition to the mains fuse in the unit's plug and is needed to protect the low voltage winding of the transformer and any circuits you connect to it.

Although we won't be building the transformer block of our 5V regulated power supply, it is interesting to know how it works.





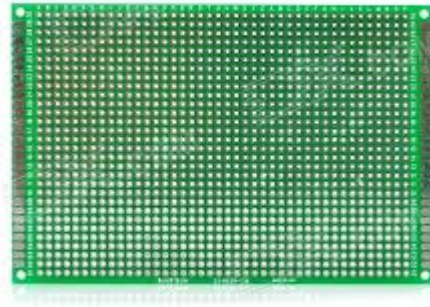
### **Connecting wires:**

Connecting wires allows an electrical current to travel from one point on a circuit to another, because electricity needs a medium through which to move. In the case of computers, wires are embedded into circuit boards, carrying pulses of electricity that are interpreted as binary signals of zeros and ones.



### **PCB Board:**

PCB or Printed Circuit Board is the traditional name for the bare board of which you supply us with the layout data and which you use to mount your components on once we have delivered it to you.



A printed circuit board, or PCB, is used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces etched from copper sheets laminated onto a non-conductive substrate.

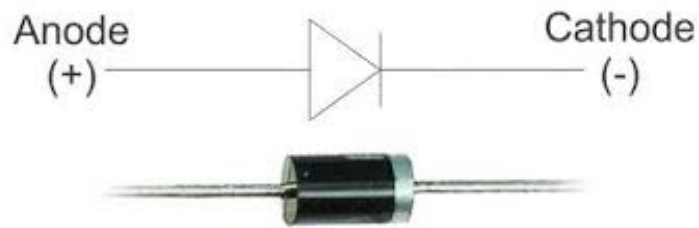
When the board has only copper tracks and features, and no circuit elements such as capacitors, resistors or active devices have been manufactured into the actual substrate of the board, it is more correctly referred to as printed wiring board (PWB) or etched wiring board. Use of the term PWB or printed wiring board although more accurate and distinct from what would be known as a true printed circuit board, has generally fallen by the wayside for many people as the distinction between circuit and wiring has become blurred.

Today printed wiring (circuit) boards are used in virtually all but the simplest commercially produced electronic devices, and allow fully automated assembly processes that were not possible or practical in earlier era tag type circuit assembly processes.

A PCB populated with electronic components is called a printed circuit assembly (PCA), printed circuit board assembly or PCB Assembly (PCBA). In informal use the term "PCB" is used both for bare and assembled boards, the context clarifying the meaning. The IPC preferred term for populated boards is CCA, *circuit card assembly*. This does not apply to backplanes; assembled backplanes are called *backplane assemblies* by the IPC.

### **Diodes, Resistors, Capacitors :**

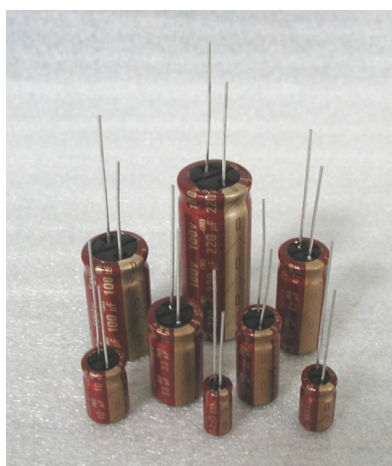
The most common function of a **diode** is to allow an electric current to pass in one direction (called the **diode's** forward direction), while blocking current in the opposite direction (the reverse direction). Thus, the **diode** can be viewed as an electronic version of a check valve.



A **resistor** is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. **Resistors** can also be **used** to provide a specific voltage for an active device such as a transistor.



**Capacitors** are components that are **used** to store an electrical charge and are **used** in timer circuits. A **capacitor** may be **used** with a resistor to produce a timer. Sometimes **capacitors** are **used** to smooth a current in a circuit as they can prevent false triggering of other components such as relays.



## Bluetooth Device:

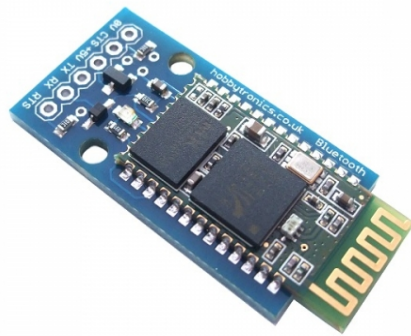
Bluetooth module consists two things one is Bluetooth serial interface module and a Bluetooth adaptor. Bluetooth serial module is used for converting serial port to Bluetooth.

Bluetooth module is the selected Bluetooth module for the designed system. Bluetooth module can cover a range of 10meters.The Bluetooth module creates the connection between the Domestic Automation and the android phone. It made the project workabale, because without the bluettoth module the android would not be able to have communication interaction with the sysyem.

It is preferable to use bluetooth because nowadays people have their smartphones with them all the time, since the smartphones have bluetooth facility in them, thus it's better to use bluetooth rather than using RF remotes or IR remotes.Have you ever seen people carryingremotes???

Using Bluetooth has many of its own advantages:

- 1.Easy to use.
- 2.Anyone can find free Bluetooth apps on android and many more.
- 3.Its secure



## **2.6 Software Components:**

- Arduino UNO software
- Android app

### **Arduino UNO software:**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

The Arduino Integrated Development Environment - or Arduino Software (IDE) - 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 and Genuino hardware to upload programs and communicate with them.

### **Android app:**

Operating Systems have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android.

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch, comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android.

### **3.DESIGN**

Design is what virtually every engineer wants to do. It is the place where customer requirements, business needs and technical considerations all come together in the formulation of a product or a system .Design creates a representation or model of a software but, unlike model(that focuses on describing data, function and behaviour),the design model provides details about software data structures, architecture, interfaces and components that are necessary to implement the system.

Design is a solution, a “how to” approach to the creation of a new system.

#### **UML DIAGRAMS**

UML is a common language for a business analysts, software architects and developers used to describe, specify, design and document existing or new business processes, structure and behaviour of artifacts of software systems.

UML is standards modelling language, not a software development process.

- Provides guidance as to the order of the teams activities.
- Specifies what artifacts should be developed, directs the task of individual developers and a team as a whole, and
- Offers criteria for monitoring and measuring a projects product and activities .

#### **Usecase diagram:**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

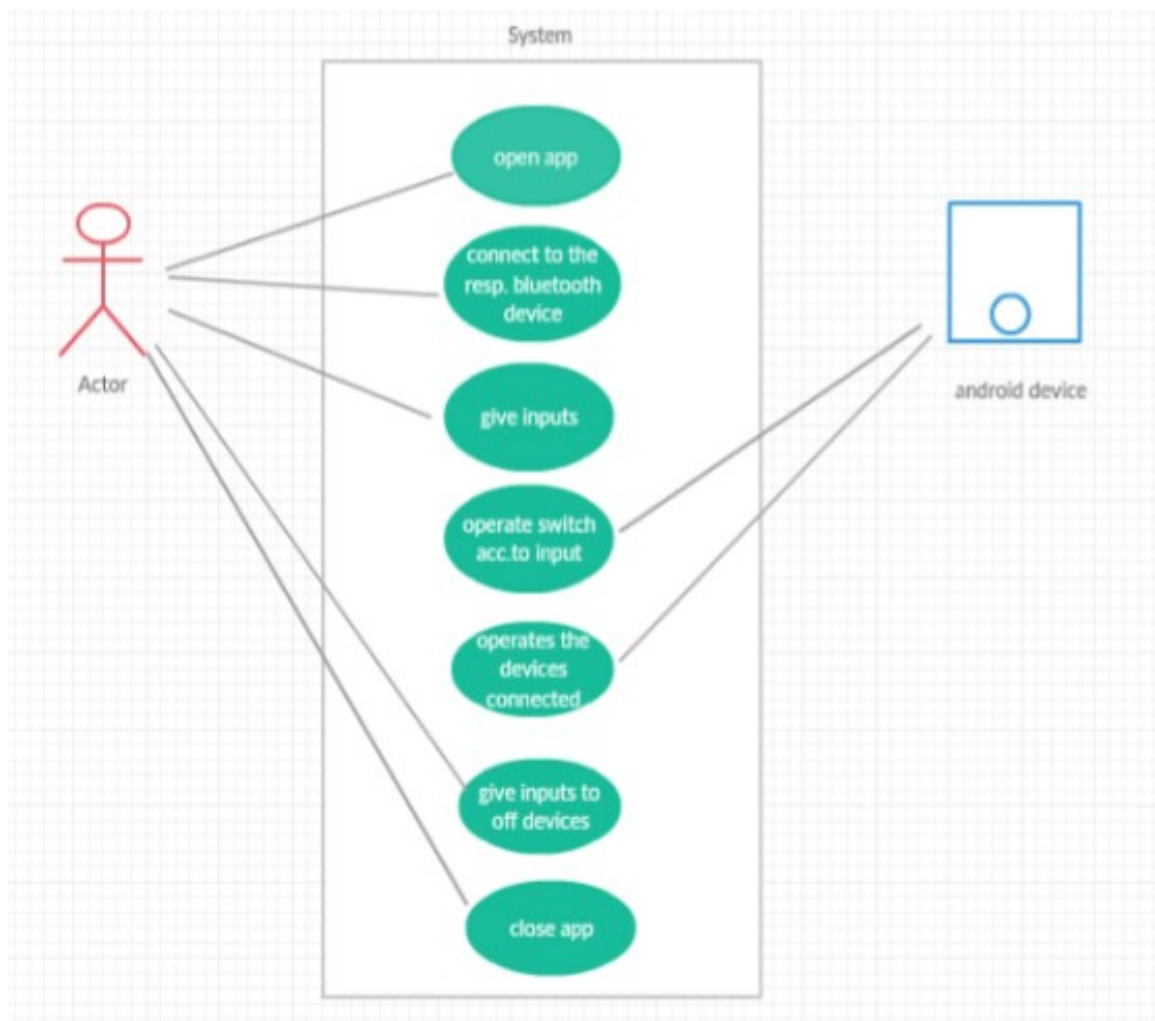
So we can say that use cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications.

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output and the function of the black box is known.

These diagrams are used at a very high level of design. Then this high level design is refined again and again to get a complete and practical picture of the system. A well

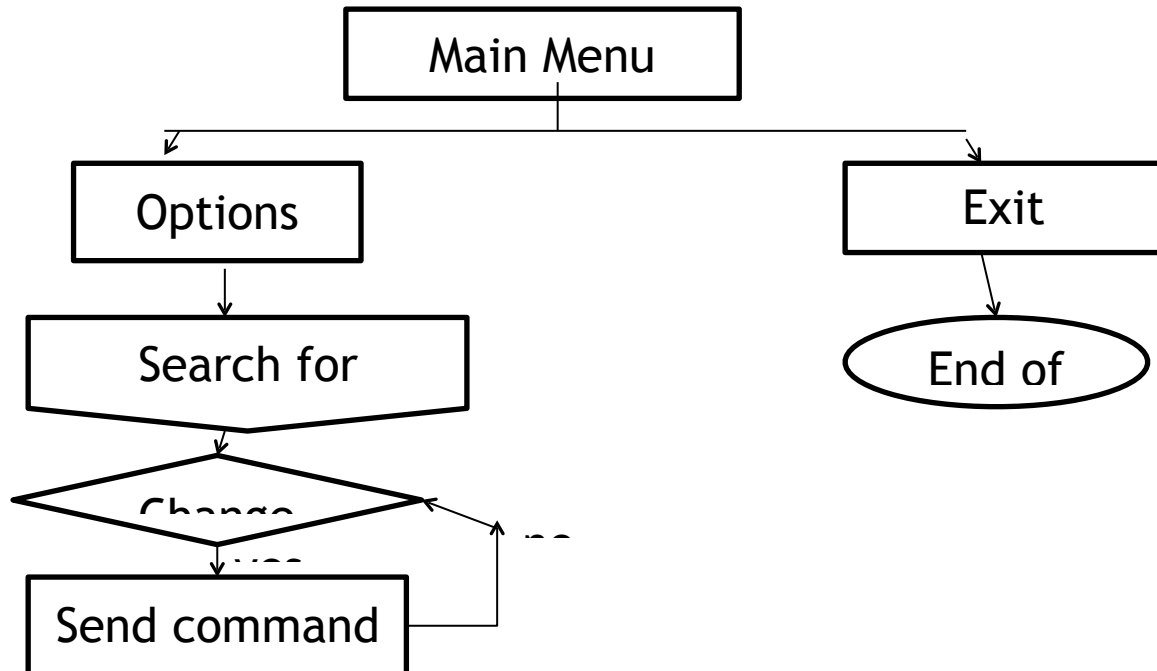
structured use case also describes the pre condition, post condition, exceptions. And these extra elements are used to make test cases when performing the testing.



**Activity Diagram:**

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram are not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlane etc.

Before drawing an activity diagram we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities we need to understand how they are associated with constraints and conditions.





## 4.IMPLEMENTATION

### 4.1 Hardware setup:

```
void setup()
{
    char val;
    pinMode(13,OUTPUT);
    pinMode(12,OUTPUT);
    Serial.begin(9600);
}
```

The above indicates that setting up modes for outputs. we used pin numbers 12 and 13 for outputs. To control serially we use the command called Serial.begin(9600).

```
void loop()
{
    if (Serial.available())
    {
        val=Serial.read();
        if(val=='L')
        {
            if(led==0)
            {
                digitalWrite(13,HIGH);
                led=1;
                delay(1000);
            }

            else

                if(led=1)
                {
                    led=0;
                    digitalWrite(13,LOW);
                }
            }
        }
    }
```

```

if(val=='F')
{
    if(led1==0)
    {
        digitalWrite(12,HIGH);
        led1=1;
        delay(1000);
    }

else

    if(led1=1)
    {
        led1=0;
        digitalWrite(12,LOW);
    }
}
}
|

```

## 4.2 Software Setup

### MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    nameView = (TextView)findViewById(R.id.controllerName);
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null) {
        Toast.makeText(getApplicationContext(), R.string.Bluetooth_NA, Toast.LENGTH_LONG).show();
    }
    else if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivity(enableBtIntent);
    }
}
```

```

protected void onResume() {
    super.onResume();
    SharedPreferences preferences = getSharedPreferences("preferences", MODE_PRIVATE);
    deviceName = preferences.getString("controllerName", "NA");
    deviceAddress = preferences.getString("controllerAddress", "");
    nameView.setText(deviceName);
    if(!deviceAddress.equals("")) {
        device = mBluetoothAdapter.getRemoteDevice(deviceAddress);
        try {
            socket = device.createInsecureRfcommSocketToServiceRecord(MY_UUID);
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
        mBluetoothAdapter.cancelDiscovery();
        try {
            socket.connect();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
            Log.e(TAG, "socket connect failed: " + e.getMessage() + "\n");
            try {
                socket.close();
            } catch (Exception e1) {
                Log.e(TAG, "socket closing failed: " + e1.getMessage() + "\n");
                Toast.makeText(getApplicationContext(), e1.getMessage(), Toast.LENGTH_LONG).show();
            }
        }
        try {
            os = socket.getOutputStream();
        } catch (Exception e) {
            Log.e(TAG, "getting output stream failed: " + e.getMessage() + "\n");
            Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
    }
    else {
        Toast.makeText(getApplicationContext(), "Connect to a device", Toast.LENGTH_LONG).show();
    }
}

```

```
@Override
public void onPause() {
    super.onPause();
    if (os != null) {
        try {
            os.flush();
        } catch (Exception e) {
            Log.e(TAG, "flushing output stream failed: " + e.getMessage() + "\n");
            Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
    }

    try {
        socket.close();
    } catch (Exception e) {
        Log.e(TAG, "closing socket failed: " + e.getMessage() + "\n");
        Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
    }
}
```

---

```

public void sendSignal(View view) {
    try {
        os.write(view.getResources().getResourceName(view.getId()).getBytes());
    }
    catch (Exception e) {
        Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
    }
}

public void selectFromList(View view) {
    Intent intent = new Intent(this, SelectController.class);
    startActivity(intent);
}
}

```

## SelectController.java

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_select_controller);

    myBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    myListView = (ListView) findViewById(R.id.deviceListView);
    BTArrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
    myListView.setAdapter(BTArrayAdapter);

    pairedDevices = myBluetoothAdapter.getBondedDevices();

    for(BluetoothDevice device : pairedDevices)
        BTArrayAdapter.add(device.getName() + "\n" + device.getAddress());

    if (myBluetoothAdapter.isDiscovering()) {
        // the button is pressed when it discovers, so cancel the discovery
        myBluetoothAdapter.cancelDiscovery();
    }
    else {
        BTArrayAdapter.clear();
        myBluetoothAdapter.startDiscovery();
        registerReceiver(bReceiver, new IntentFilter(BluetoothDevice.ACTION_FOUND));
    }

    myListView.setOnItemClickListener((parent, view, position, id) → {
        String[] deviceDetails = ((TextView) view).getText().toString().split("\n");
        BluetoothDevice device = myBluetoothAdapter.getRemoteDevice(deviceDetails[0]);
        device.createBond();
        SharedPreferences preferences = getSharedPreferences("preferences", MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("controllerName", deviceDetails[0]);
        editor.putString("controllerAddress", deviceDetails[1]);
        editor.commit();
        finish();
    });
}

```

## 5. TESTING

Testing is the process of detecting the errors present in the project. It does not stop with just detecting, but continues with the correction of errors. Testing focuses on all the minute errors that occur while executing the project. A good test case is a one that has a high probability of finding an as yet undiscovered error. A software testing usually performs for one of these two reasons :

- Defect detection .
- Reliability estimation.

## **TESTING OBJECTIVES**

- Testing is the process of executing a program with the intent of finding an error.
- A good test case is one that has a higher probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

These objectives imply a dramatic changes in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically uncover different classes of errors and to do so with a minimum of time and effort.

If testing is conducted successfully, It will uncover errors in the software . As a secondary benefit, demonstrates that software functions appear to be working according to specifications, that behavioural and performance requirements appear to have been met. In addition data collected as testing is conducted provide a good of software reliability and some indication of software quality as a whole. But, testing cannot show the absence of errors and defects it can show only that software errors and defects are present.

## **System Testing**

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

## **Software Testing - Validation Testing**

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

### **Black box Testing**

Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle.

This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

### **White Box Testing**

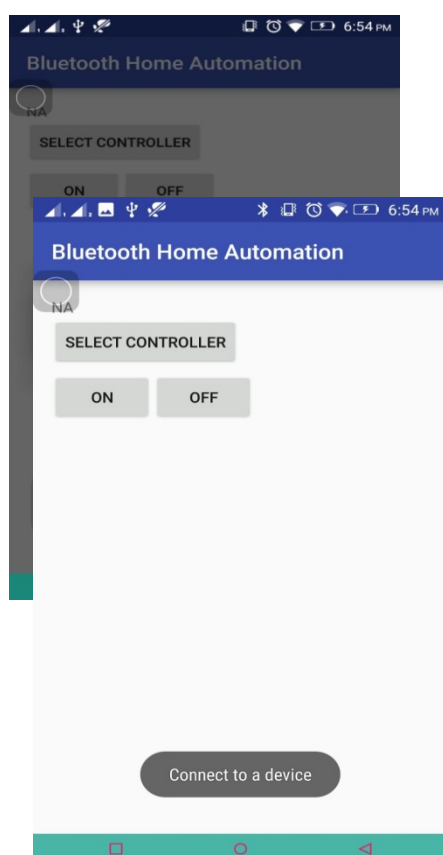
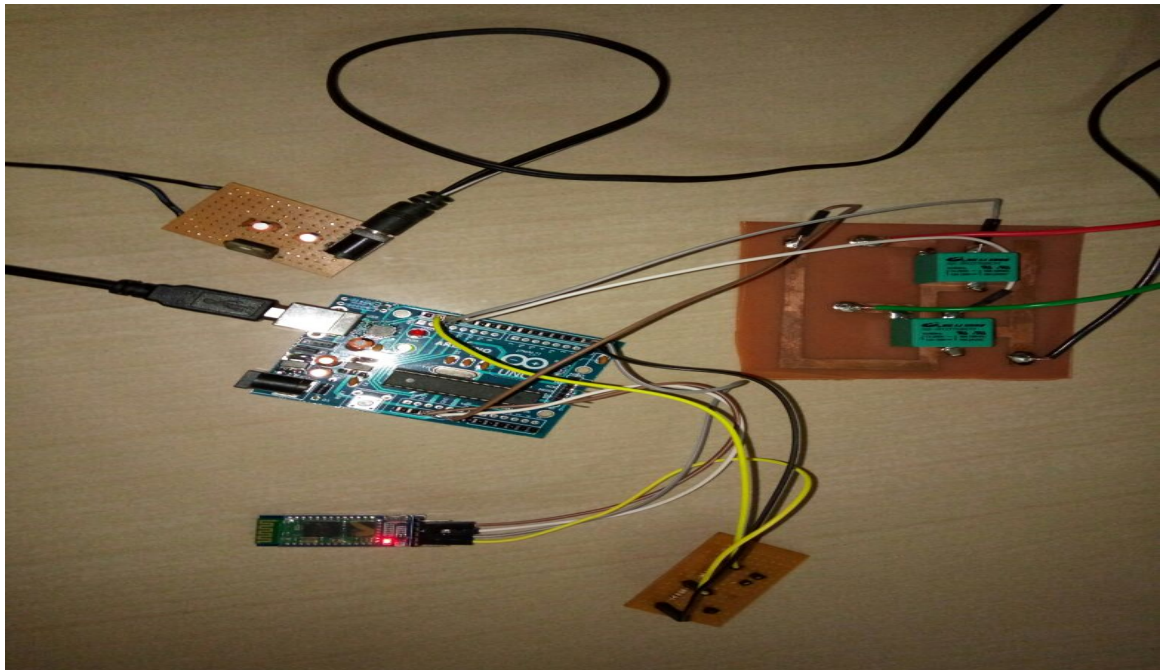
White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

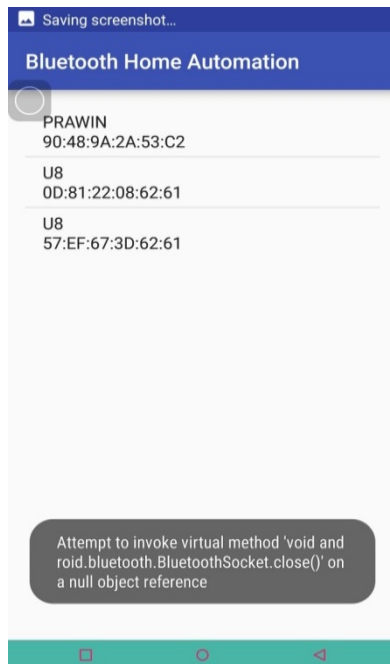
#### **Test cases:**

Test case id	Test case name	Description	Prerequisite	Expected output	Observed output
TC_01		user shall enable the app and connect to the Bluetooth module	Network available	User receives a message that pairing has done	Yes the user has done the pairing
TC_02		User need to check whether the relays are getting the power supply	Network Available	User knows that the relay is getting the power supply	Yes the relay is getting the power supply



TC_03		User check whether the appliances are properly working	Network Available	User knows That the appliances are working properly	Yes the appliances are functioning properly
-------	--	--	-------------------	---	---





## CONCLUSION

In conclusion, this low cost system is designed to improve the standard living in home. The remote control function by smart phone provides help and assistance especially to disabled and elderly. In order to provide safety protection to the user, a low voltage activating switches is replaced current electrical switches. Moreover, implementation of wireless Bluetooth connection in control board allows the system install in more simple way. The control board is directly installed beside the electrical switches whereby the switching connection is controlled by relay. Furthermore, flexible types of connections are designed as backup connections to the system. The connected GUIs are synchronized to the control board. They indicate the real-time switches status. The system is designed in user-friendly interface. The easy to use interface on Window and Android GUI provides simple control by the elderly and disabled people. For future work, the Window

GUI will be implemented with speech recognition voice control. The android GUI will be implemented as a remote Bluetooth microphone to the Window GUI. All the voice signal inputs to the smart phone will be transmitted to the Window GUI for signal processing. Also, the push buttons implemented in low voltage activating switches will be replaced by capacitive sensing switches. All the future work is expected without spend extra cost, even one cent from the current system.

## **FUTURE ENHANCEMENT**

### **Applications and Advantages:**

- Home automation – This project can be used to control various Home Appliances
- We can control device from a long distance, thus it gives ease of access.
- Faster operation and efficient.
- No need to carry separate remote or any other controlling unit.
- It is a robust and easy to use system.
- There is no need for extra training of that person who is using it.
- All the control would be in your hands by using this home automation system.

- ### Disadvantages:

- ### Future Development:

## REFERENCES

<https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjNqu3azLLPAhUHpY8KHbi-CPMQjhwIBQ&url=http%3A%2F%2Fwww.amazon.in%2FGL-Electronics-INFILM-Bluetooth-Headset%2Fdp%2FB00NW2JBYO&psig=AFQjCNFYQaWfxdc28B0XEVZ58DXVxfM3Rg&ust=1475170548722885>

36

1. <http://www.projectsof8051.com/home-appliances-controlling-using-android-mobile-via-bluetooth/>
2. [http://www.electronicshub.org/bluetooth-controlled-electronic-home-appliances/#Bluetooth\\_Controlled\\_Electronic\\_Home\\_Appliances\\_Circuit\\_Diagram](http://www.electronicshub.org/bluetooth-controlled-electronic-home-appliances/#Bluetooth_Controlled_Electronic_Home_Appliances_Circuit_Diagram)
3. <http://circuitdigest.com/microcontroller-projects/bluetooth-controlled-home-automation-using-8051>
4. <http://microcontrollerslab.com/bluetooth-home-automation-system-android/>

## **APPENDIX A**

### **Arduino UNO software:**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

The Arduino Integrated Development Environment - or Arduino Software (IDE) - 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 and Genuino hardware to upload programs and communicate with them.

## Writing Sketchs

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension `.ino`. The editor 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 Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.



### *Verify*

Checks your code for errors compiling it.



### *Upload*

Compiles your code and uploads it to the configured board.

See uploading below for details.

Note: If you are using an external programmer with your board, 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 overwriting its content.

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 | Sketchbookmenu 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.

## Sketch Book

The Arduino Software (IDE) 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.

## 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 visible 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 > Port menus. 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/ttyACMx ,/dev/ttyUSBx 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 Sketch 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 Software (IDE) 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).

### **Serial Monitor**

Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin** in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuino board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor.

### **Processing**

The Arduino Uno can be programmed with the Arduino software. Select "Arduino Uno" from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a boot loader 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). We can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU boot loader, 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. The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard



USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

## **APPENDIX B**

### **Android app:**

Operating Systems have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android.

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch, comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by

Google in 2005. After original release there have been number of updates in the original version of Android.

## **Installing Android Studio**

Android Studio provides everything you need to start developing apps for Android, including the Android Studio IDE and the Android SDK tools.

Before you set up Android Studio, be sure you have installed JDK 6 or higher (the JRE alone is not sufficient)—JDK 7 is required when developing for Android 5.0 and higher. To check if you have JDK installed (and which version), open a terminal and type `javac -version`. If the JDK is not available or the version is lower than version 6, download the [Java SE Development Kit 7](#).

### **To set up Android Studio on Windows:**

- Launch the .exe file you just downloaded.
- Follow the setup wizard to install Android Studio and any necessary SDK tools.

On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

Select **Start menu > Computer > System Properties > Advanced System Properties**. Then open **Advanced tab > Environment Variables** and add a new system variable `JAVA_HOME` that points to your JDK folder, for example `C:\Program Files\Java\jdk1.7.0_21`.

The individual tools and other SDK packages are saved outside the Android Studio application directory. If you need to access the tools directly, use a terminal to navigate to the location where they are installed. For example:


```
\Users\<user>\sdk\
```

Android Studio is now ready and loaded with the Android developer tools, but there are still a couple packages you should add to make your Android SDK complete.

## Adding SDK Packages

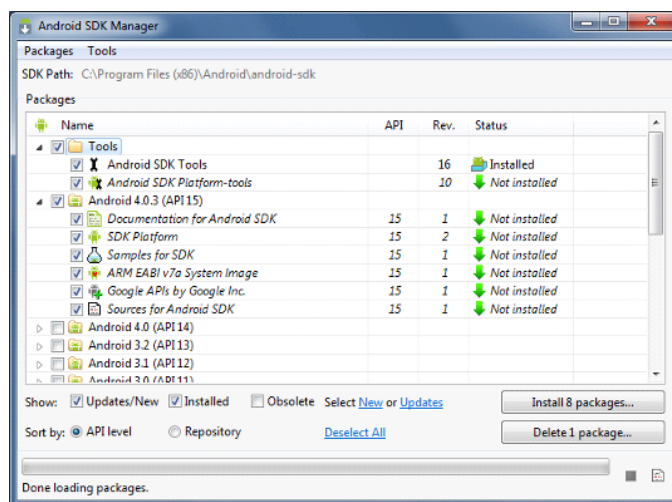
By default, the Android SDK does not include everything you need to start developing. The SDK separates tools, platforms, and other components into packages you can download as needed using the [Android SDK Manager](#). So before you can start, there are a few packages you should add to your Android SDK.

To start adding packages, launch the Android SDK Manager in one of the following ways:

- In Android Studio, click **SDK Manager**  in the toolbar.
- If you're not using Android Studio:
- Windows: Double-click the SDK Manager.exe file at the root of the Android SDK directory.
- Mac/Linux: Open a terminal and navigate to the `tools/` directory in the location where the Android SDK was installed, then execute `android sdk`.

When you open the SDK Manager for the first time, several packages are selected by default. Leave these selected, but be sure you have everything you need to get started by following these steps:

- Get the latest SDK tools



As a minimum when setting up the Android SDK, you should download the latest tools and Android platform:

- Open the Tools directory and select:

- **Android SDK Tools**
- **Android SDK Platform-tools**
- **Android SDK Build-tools** (highest version)
  - Open the first Android X.X folder (the latest version) and select:
- **SDK Platform**
  - A system image for the emulator, such as **ARM EABI v7a System Image**
- Get the support library for additional APIs

The support library is required for:

- Android Wear
- Android TV
- Google Cast

It also provides these popular APIs:

- Navigation drawer
- Swipe views
- Backward-compatible action bar

The Android Support Library provides an extended set of APIs that are compatible with most versions of Android.

Open the **Extras** directory and select:

- **Android Support Repository**
- **Android Support Library**

1) Get Google Play services for even more APIs

The Google Play services APIs provide a variety of features and services for your Android apps, such as:

- User authentication
- Google Maps
- Google Cast

- Games achievements and leaderboards
- And much more

To develop with Google APIs, you need the Google Play services package:

Open the **Extras** directory and select:

- **Google Repository**
- **Google Play services**

**Note:** Google Play services APIs are not available on all Android-powered devices, but are available on all devices with Google Play Store. To use these APIs in the Android emulator, you must also install the **Google APIs** system image from the latest Android X.X directory in the SDK Manager.

1) Install the packages

Once you've selected all the desired packages, continue to install:

1. Click **Install X packages**.
2. In the next window, double-click each package name on the left to accept the license
3. Click **Install**.

The download progress is shown at the bottom of the SDK Manager window. **Do not exit the SDK Manager** or it will cancel the download.

1) Build something!

With the above packages now in your Android SDK, you're ready to build apps for Android. As new tools and other APIs become available, simply launch the SDK Manager to download the new packages for your SDK.

Here are a few options for how you should proceed:

Android Studio IDE and the Android SDK tools.

If you didn't download Android Studio, go **[download Android Studio now](#)**, or switch to the **[stand-alone SDK Tools install](#)** instructions.

Before you set up Android Studio, be sure you have installed JDK 6 or higher (the JRE alone is not sufficient)—JDK 7 is required when developing for Android 5.0 and higher. To check if you have JDK installed (and which version), open a terminal and type `javac -version`. If the JDK is not available or the version is lower than version 6, download the [Java SE Development Kit 7](#).

**To set up Android Studio on Windows:**

- Launch the .exe file you just downloaded.
- Follow the setup wizard to install Android Studio and any necessary SDK tools.

On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

Select **Start menu > Computer > System Properties > Advanced System Properties**. Then open **Advanced tab > Environment Variables** and add a new system variable `JAVA_HOME` that points to your JDK folder, for example `C:\Program Files\Java\jdk1.7.0_21`.