Chapter I
Introduction

Introduction

Home automation refers to the use of computer and information technology to control home appliances and features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/micro-controller based networks with varying degrees of intelligence and automation[1]. Home automation is adopted for reasons of ease, security and energy efficiency. It is an integral part of human life which is ever evolving with technology. Some devices are left plugged into power sockets whereas others are supposed to be plugged into and out of power sockets at different intervals depending on the time of the day. All this requires an individual to manually attend to each of the devices independently from time to time. Some devices if not controlled properly consume a lot of energy which leads to extra expenditure on electricity. Hence, the motive behind this project is inevitably justified.

1.1 Software (Android Application Development):

Android is a software stack for that includes an operating system for mobile devices, middleware and key applications. The Android OS is based on an operating system Linux. Android Applications are made in a Java-like language with many android based libraries running on a virtual machine called ‘Dalvik’ created by Google. The Android Software Development Kit i.e. SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language and Extended Markup language[5].

1.1 Hardware (ARM’s Mbed Microcontroller & GSM Module):

Mbed is a platform consisting of 32-bit ARM Cortex M microcontrollers for developing smart devices. It is a project developed and handled by ARM, its technology partners and a strong community of core developers have resulted into an overwhelming support of tens of thousands of professional developers to create intelligent products that take advantage of the power of modern microcontroller[3]. FRDM-kl25z is used as a microcontroller development unit for this purpose. GSM Module SIM300 can accept SIM card of any GSM network operator and act just like a mobile phone with its own unique phone number[2]. Advantage of using this modem will be that you can use its RS232 port to communicate and develop embedded applications.

Applications like SMS Control, data transfer, remote control and logging can be developed easily.

Chapter II
Literature Survey

Literature Survey

As per our survey currently there exists no system at cheaper rates. Various systems are hard to install, difficult to use and maintain. Current systems are generally proprietary and closed, not very customizable by the end user. So, with this project it goes without saying that our intention is to implement a completely user-friendly and cost effective Home Automation System with minimalistic hardware and user-experienced-simplified Android Application. Our implementation is developed on the lines of the two papers referred below.

2.1 Paper 1 (Home Automation and Security System Using Android ADK IJECCT 2013):

This paper uses Android Open Accessory Protocol and Arduino Mega ADK Microcontroller board to control home appliances. Android Open Accessory support allows external USB hardware (an Android USB accessory) to interact with an Android-powered device in a special accessory mode. When an Android-powered device is in accessory mode, the connected accessory acts as the USB host (powers the bus and enumerates devices) and the Android-powered device acts in the USB accessory role. Android Open Accessory Protocol allows detecting Android-powered devices that support accessory mode. Accessory mode is ultimately dependent on the device's hardware and not all devices support accessory mode. The design consists of Android phone with home automation application, Arduino Mega ADK. User can interact with the android phone and send control signal to the Arduino ADK which in turn will control other embedded devices/sensors.

2.2 Paper 2 (Bluetooth Remote Home Automation System Using Android Application IJES 2013):

The system here is directly installed beside the conventional electrical switches on the wall. Bluetooth wireless connection enabled the system communicates with graphical user interface (GUI) on PC/laptop or smart phone without cable. The target home appliances are controlled by the system Main Control Board. Several control methods are performed as wireless remote control to the appliances. The first control method is by clicking on Window GUI on PC/laptop by using mouse or touch pad. This method provides facility to the computer user to control the home appliances without walk to the switches on the wall. After the smartphone’s Bluetooth connection is connected to personal computer or laptop, the Window GUI will be act as a server to forward or transmit any data from/to the smart phone and main control board.

Chapter III
System Architecture

3.1 Block Diagram

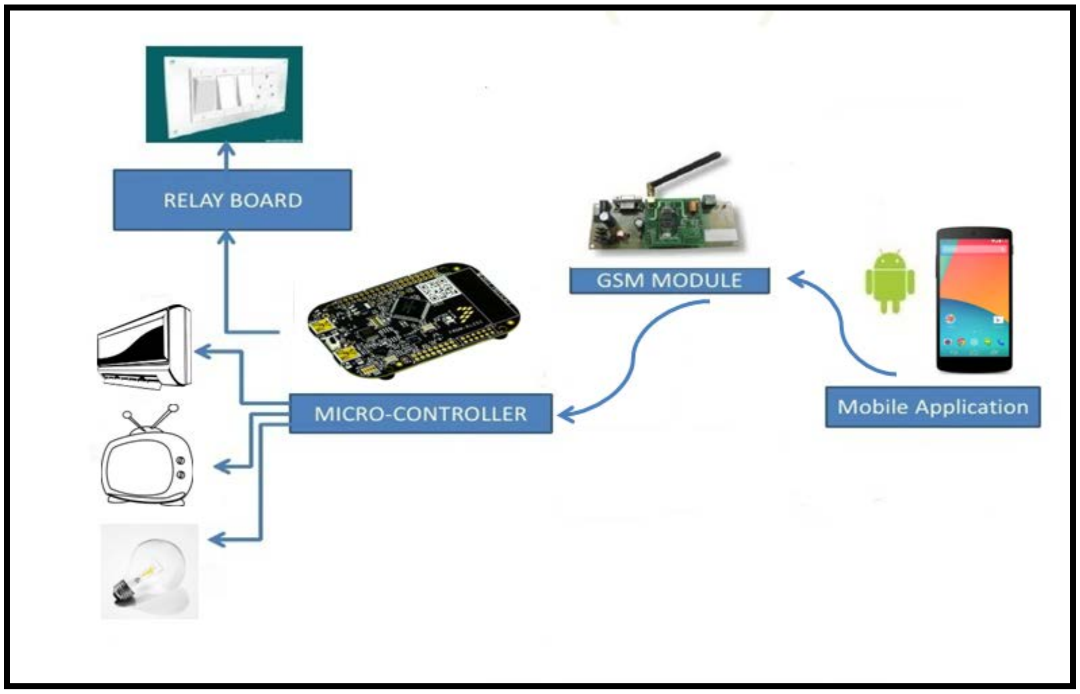


Fig. 3.1 Basic Block Diagram of Home Automation System

3.2 Flowchart

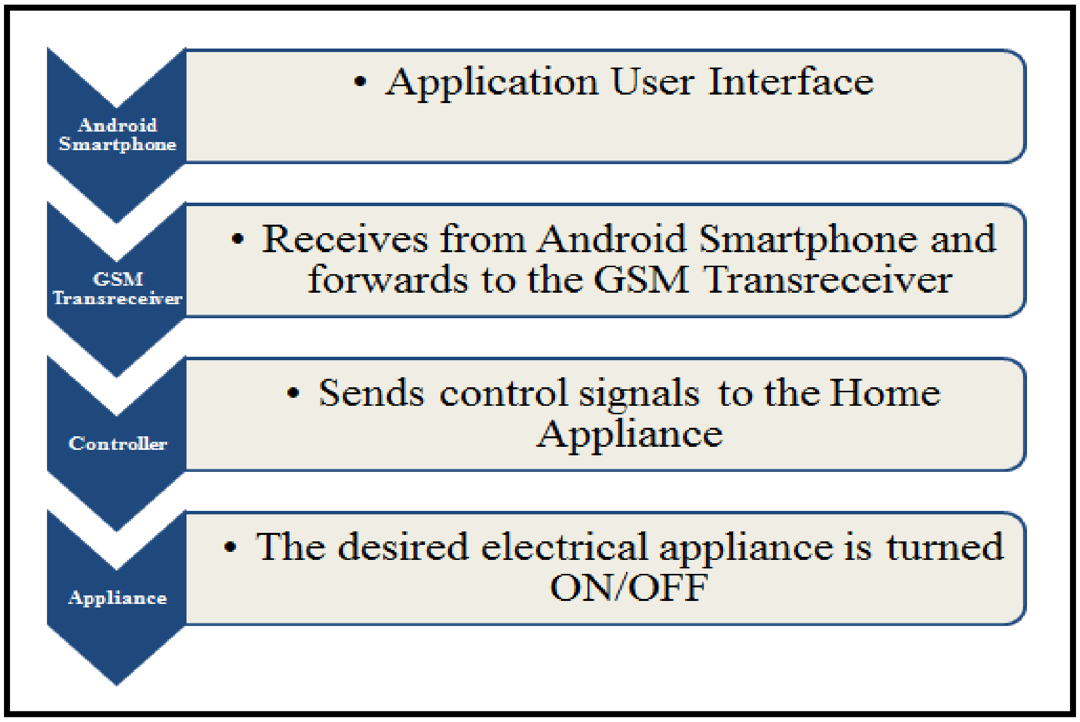

Android Smartphone
• Application User Interface
GSM Transreceiver
• Receives from Android Smartphone and forwards to the GSM Transreceiver
Controller
• Sends control signals to the Home Appliance
Appliance
• The desired electrical appliance is turned ON/OFF

Fig. 3.2 Flowchart Design and Block Diagram Description

Chapter IV
Software Components
(Appl. Development)

4.1 Software Elements

4.1.1 Android Application Overview:

For this home automation and security system we are targeting Android platform since it has huge market and open source. Android is a software stack for mobile devices that includes an operating system, middleware and key applications[3]. The Android OS is based on Linux. Android Applications are made in a Java-like language running on a virtual machine called ‘Dalvik’ created by Google. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. Thus the interface is a feature of Android OS since version 2.3.4 Gingerbread and 3.1 Honeycomb and above[5].

4.1.2 Android Operating System Overview:

Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team)[3].

Google wanted Android to be open and free. Hence, most of the Android code was released under the open source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code. Moreover, vendors (typically hardware manufacturers)can add their own proprietary extensions to Android and customize Android to differentiate their products from others[5].

This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phenomenon of Apple’s iPhone, a hugely successful product that revolutionized the smartphone industry. Such companies include Motorola and Sony Ericsson, which for many years have been developing their own mobile operating systems.

4.1.3 Android Architecture:

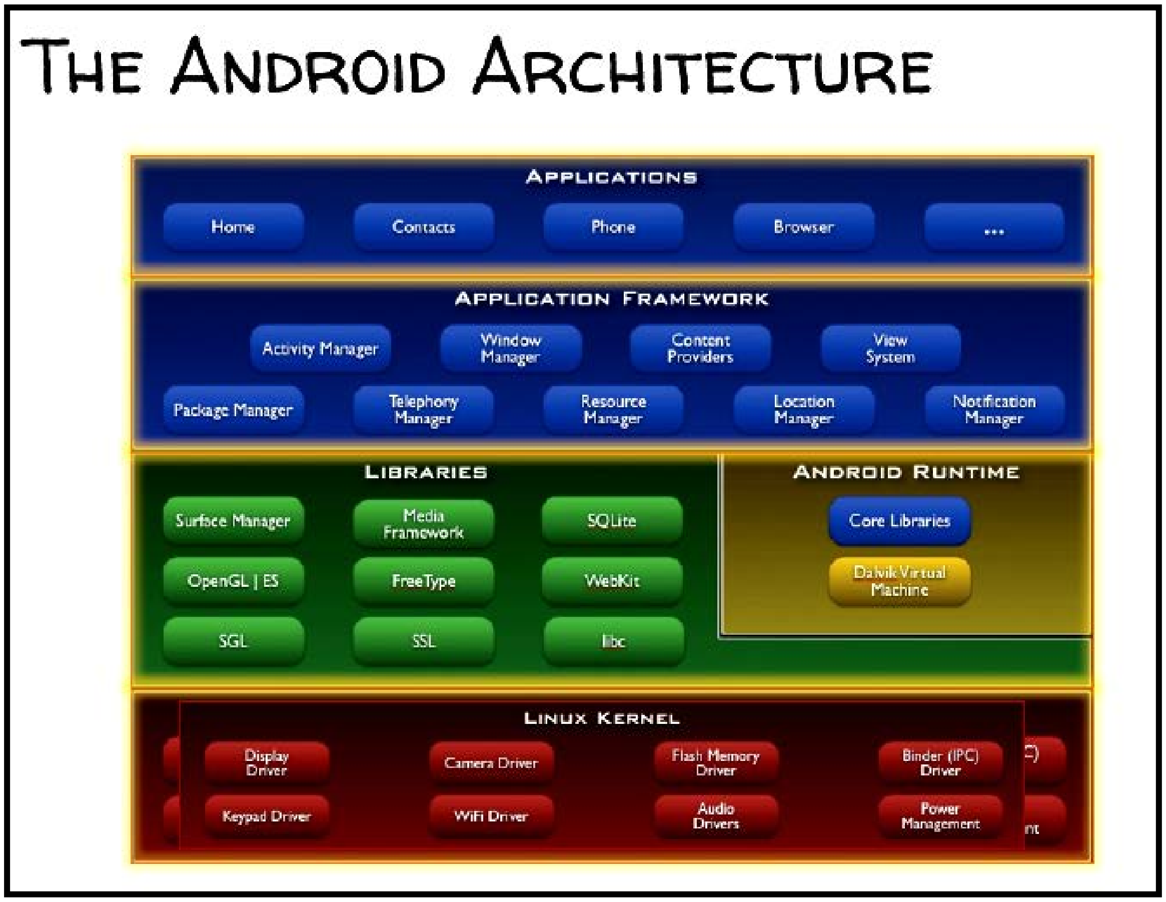


Fig. 4.1.3 Android Architecture (Programming Levels and OS Framework)

In order to understand how Android works, take a look at Figure 1-1, which shows the various layers that make up the Android operating system (OS).

The Android OS is roughly divided into five sections in four main layers:

Linux kernel: This is the kernel on which Android is based. This layer contains all the low level device drivers for the various hardware components of an Android device.

Libraries: These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

Android runtime: At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process; with its own instance of the Dalvik virtual machine (Android applications are compiled into Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Application framework: Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

Applications: At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

4.2 Software Design

As discussed earlier we are developing Android application. The application consists of main function like light controlling, Door controlling, Smoke detection and Temperature sensing. When the application starts user is first authenticated, if user is authorized he will be navigated to main screen. The main screen has a list of all functions among which user can select any one function which he want to control. After selecting a function he would be able to see a current status of a particular device. If user wishes, he can enable or disable intended device.

The system is smart enough to activate alarm when smoke is detected or it is programmed to auto on/off lights during late night hours. If room temperature goes very high or low it can automatically adjust fan/AC as per the temperature. It has voice navigation which is specifically beneficial to blind people[5].

4.3 Features of Android

Android is open source and freely available to manufacturers for customization, there are no fixed hardware or software configurations. However, Android itself supports the following features:

Storage: Uses SQLite, a lightweight relational database, for data storage.

Connectivity: Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includesA2DP and AVRCP), Wi-Fi, LTE, and WiMAX.

Messaging: Supports both SMS and MMS.

Media support: Includes support for the following media: H.263, H.264 (in 3GP or MP4

container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or

3GP container), MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF, and BMP

4.4 Application Development Elements

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

Components	Description
Activities	They dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Table 4.4 Application Development Elements and their description

4.4.1 Activities:

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of Activity class as follows:

```
public class MainActivity extends Activity{}
```

4.4.2 Services:

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.A service is implemented as a subclass of Service class as follows:

```
public class MyService extends Service{}
```

4.4.3 Broadcast Receivers:

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcasted as an Intent object.

```
public class MyReceiver extends BroadcastReceiver{}
```

4.4.4 Content Providers:

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely. A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider{}
```

4.5 Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them[5]. These components are:

Components	Description
Fragments	Represents a behavior or a portion of user interface in an Activity.
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resources	External elements, such as strings, constants and drawables pictures.
Manifest	Configuration file for the application.

Table 4.5 Application Development Elements and their description

4.5.1 Description of Additional Components:

1. Android Manifest.xml:

An Android application is described in the file AndroidManifest.xml. This file must declare all Activities, Services, Broadcast Receivers and Content Provider of the application. It must also contain the required permissions for the application AndroidManifest.xml can be thought as the deployment descriptor for an Android Application.

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of. It names the classes that implement each of the components and publishes their capabilities (for example, which Intent messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.

2. R.java, Resources and Assets:

The directory gen in an Android project contains generated values. R.java is a generated class which contains references to resources of the res folder in the folder in the project. These resources are defined in the res directory and can be values, menus, layouts, icons or pictures or animations. For example, a resource can be a image or an XML file which defines strings. If we create a new resource the corresponding reference is automatically created in R.java. The references are static int values, the Android system provides methods to access the corresponding resource.

3. Reference to resource in XML files:

In our XML files, example our layout files we can refer to other resources via the @ sign. For example if we want to refer to a string “hello” as resource we can access it via @string/hello.

4. Layouts:

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways:

- Declare UI elements in XML: Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- Instantiate layout elements at runtime: The application can create View and ViewGroup objects (and manipulate their properties) programmatically.
- The Android framework gives you the flexibility to use either or both of these methods for declaring and managing your application's UI. For example, we could declare your application's default layouts in XML, including the screen elements that will appear in them and their properties. We could then add code in your application that would modify the state of the screen objects, including those declared in XML, at run time.
- The ADT Plugin for Eclipse offers a layout preview of your XML — with the XML file opened, select the Layout tab.

The advantage to declaring your UI in XML is that it enables you to better separate the presentation of your application from the code that controls its behavior. Your UI descriptions are external to your application code, which means that you can modify or adapt it without having to modify your source code and recompile[5].

4.6 Lifecycle of an Activity

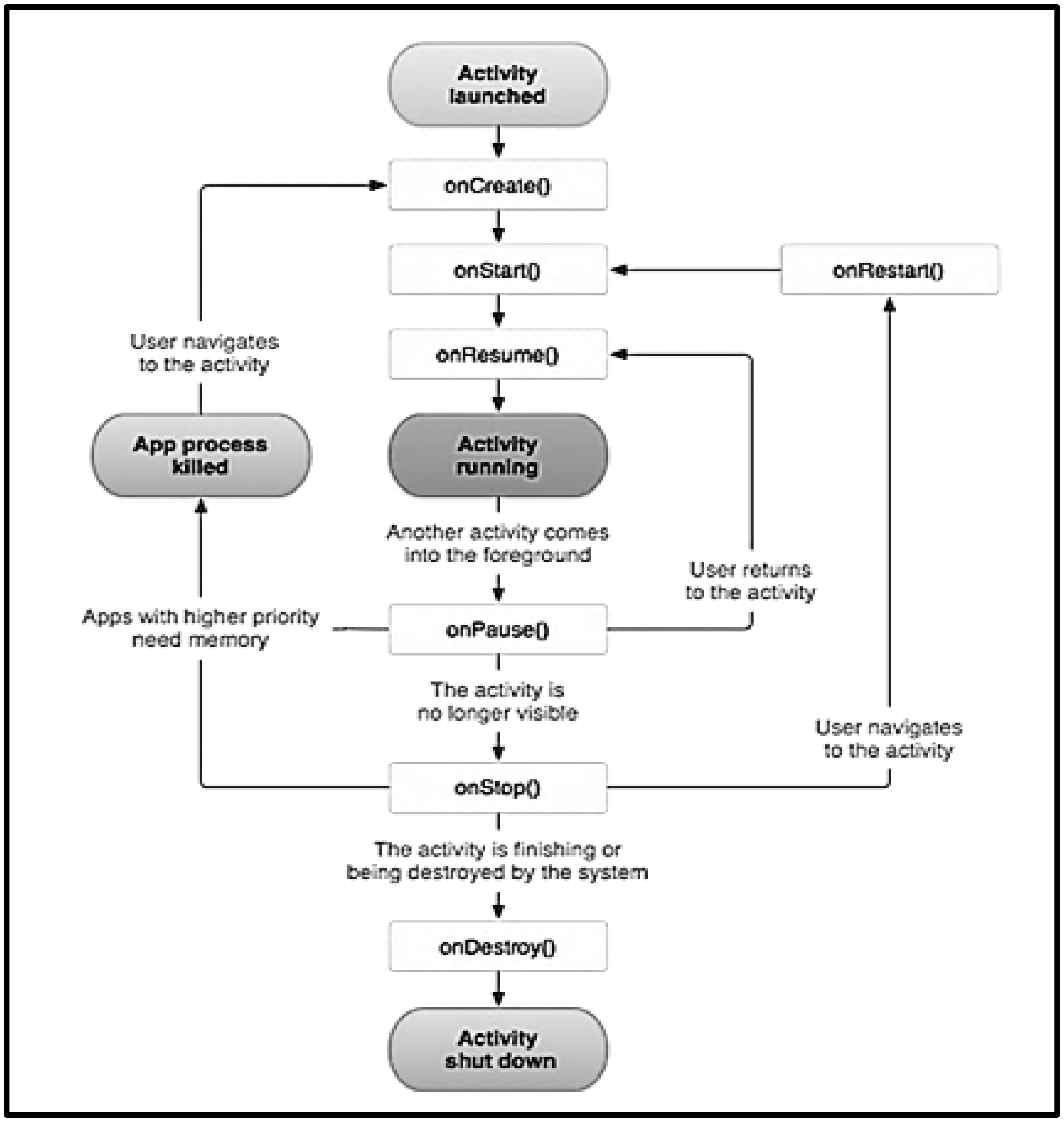

Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Fig. 4.6 Lifecycle of an Activity

Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits[5].

An activity has essentially four states:

- If an activity in the foreground of the screen (at the top of the stack), it is active or running.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is paused.
- If an activity is completely obscured by another activity, it is stopped.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.

4.7 Application Elements

4.7.1 Toggle Button:

- A toggle button allows the user to change a setting between two states.

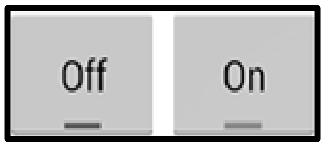


Fig. 4.7.1 Toggle Button Widget

- We can add a basic toggle button to layout with the ToggleButtonobject.
- Responding to Click Events.
- For example, here's a ToggleButton with the android:onClick attribute:
- <ToggleButton
  android:id="@+id/togglebutton"
  android:onClick="onToggleClicked"/>

4.7.2 Text View:

- Text view displays text to the user. it is a complete text editor, however the basic class is configured to not allow editing.

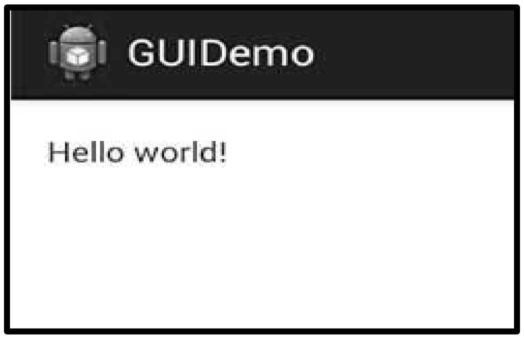


Fig. 4.7.2 Text View Widget

- For example, here's a TextView implementation in XML:
- <TextView

```
            android:id="@+id/welcome"
            android:layout_gravity="center" />
```

4.7.3 Edit Text:

- An EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.

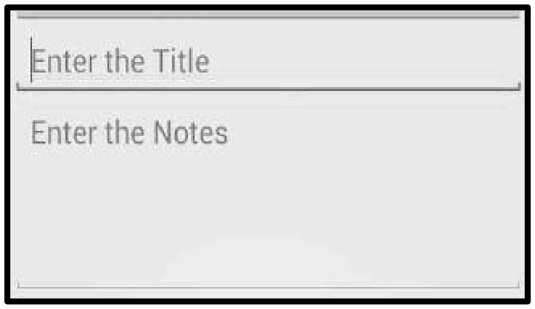


Fig. 4.7.3 Edit Text Widget

- For example, here's a TextView implementation in XML:
- <EditText

```
           android:id="@+id/edittext"
           android:text="@string/enter_text"
           android:inputType="text" />
```

4.8 Layouts in Android

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways. UI elements are declared in XML. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts. Layout elements are instantiated at runtime. Your application can create View and ViewGroup objects (and manipulate their properties) programmatically[5].

4.8.1. Linear Layout:

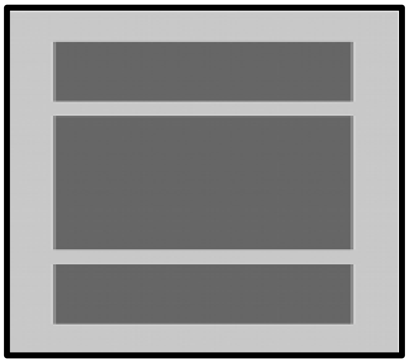


Fig. 4.8.1 Linear Layout

A linear layout organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

4.8.2 Relative Layout:

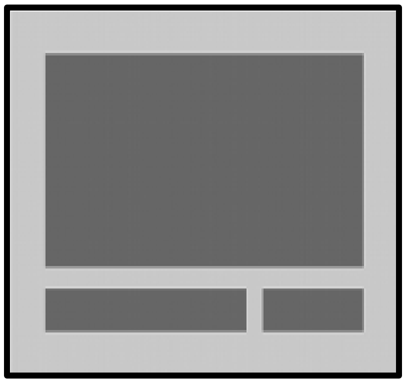


Fig. 4.8.2 Relative Layout

Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

4.8.3 Layout Overview:

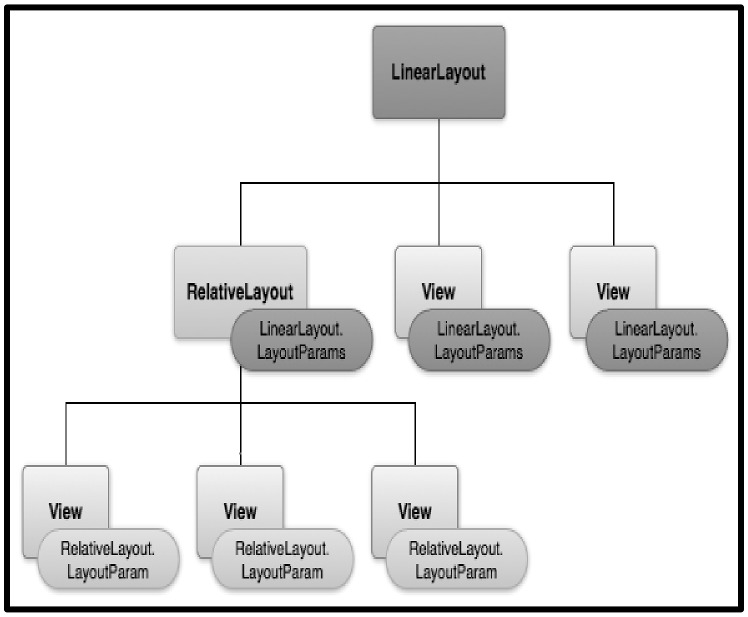

LinearLayout
RelativeLayout
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam

Fig. 4.8.3 Layout Structure in Android ADK

Layout implementation in XML:

-
  ```
  <LinearLayout>
        <RelativeLayout>
                <EditText
        </RelativeLayout>
        <RelativeLayout >
                <Button
        </RelativeLayout>
  </LinearLayout>
  ```

4.9 List Activity in Android

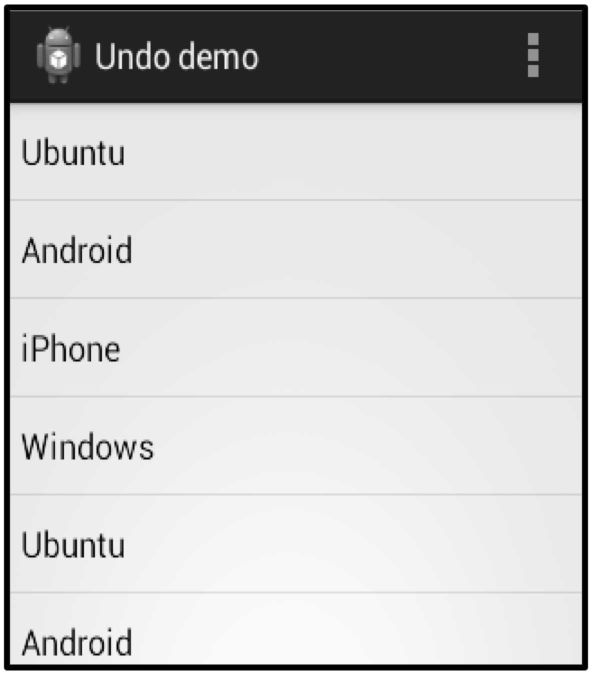


Fig. 4.9 List View Layout in Android

List activity displays a list of items by binding to a data source such as an array or Cursor, and exposes event handlers when the user selects an item. ListActivity hosts a Listview object that can be bound to different data sources, typically either an array or a Cursor holding query results. Binding, screen layout, and row layout are discussed in the following sections[5].

4.9.1 ListActivity implementation:

- For example, here's a ListActivity implementation in XML:

```
<RelativeLayout>
    <ListView
          android:entries="@array/appliance" >
   </ListView>
</RelativeLayout>
```

Chapter V
Hardware Components

5.1 ARM’s Mbed (FRDM-kl25z) Microcontroller

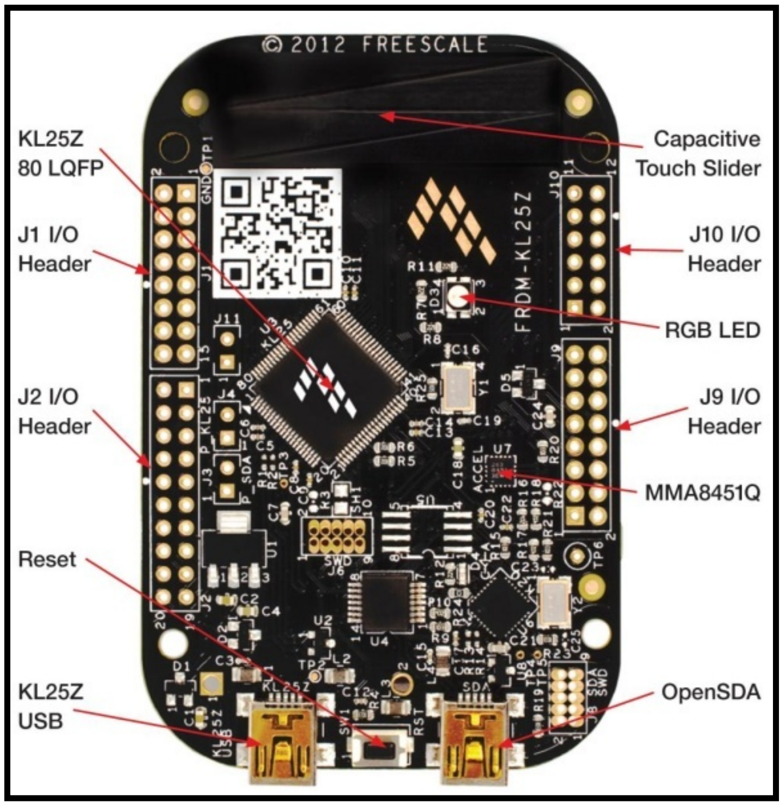


Fig. 5.1 ARM’s Mbed based Microcontroller Board FRDM-kl25z

5.1.1 Mbed Overview:

Mbed is a platform for developing smart devices that are based on 32-bit ARM Cortex M microcontrollers. It is designed to provide a highly productive solution for rapid prototyping and product development, with a focus on connected Internet of Things devices.

It is a project developed by ARM, its technology partners and a community of core developers, and it is used by tens of thousands of professional developers to create intelligent products that take advantage of the power of modern microcontrollers and connectivity[6].

5.1.2 Software Development Kit:

The mbed software development kit (SDK) provides the mbed C/C++ software platform and tools for creating microcontroller firmware that runs on smart devices. It consists of the core libraries that provide the microcontroller peripheral drivers, networking, RTOS and runtime environment, build tools and test and debug scripts[3].

5.1.3 Hardware Development Kit:

The mbed hardware development kit (HDK) provides the recipes to build custom hardware for devices that support the mbed SDK. This consists of interface firmware and schematics of the microcontroller subsystems that can be used to easily create development boards, OEM modules and re-programmable/hackable products suitable for production and that can take advantage of the mbed software platform and development tools[6].

5.1.4 FRDM-kl25z Pin Configuration & Description:

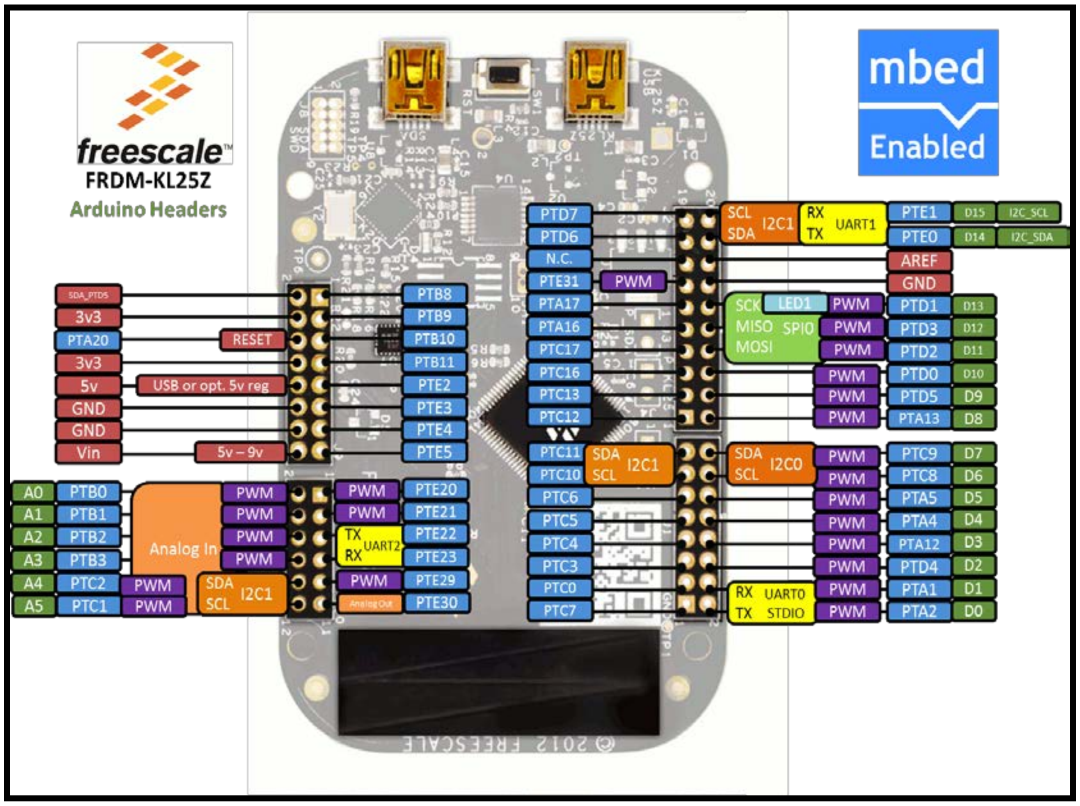


Fig. 5.1.4 FRDM-kl25z Pin Description and I/O Ports Configuration

The FRDM-KL25Z is the first hardware platform to feature the Freescale open standard embedded serial and debug adapter known as OpenSDA. This circuit offers several options for serial communications, flash programming and run-control debugging.

The FRDM-KL25Z is an ultra-low-cost development platform for Kinetis L Series KL1x (KL14/15) and KL2x (KL24/25) MCUs built on ARM® Cortex™-M0+ processor. Features include easy access to MCU I/O, battery-ready, low-power operation, a standard-based form factor with expansion board options and a built-in debug interface for flash programming and run-control.

5.1.5 Mbed Online Compilation:

Mbed consists of online compiler. The mbed Compiler provides a lightweight online C/C++ IDE that is pre-configured to let us quickly write programs, compile and download them to run on our mbed Microcontroller. In fact, we don't have to install or set up anything to get running with mbed. Because it is a web app, we can log in from anywhere and carry on where you left off, and you are free to work on Windows, Mac, iOS, Android, Linux, or all of them.

The compiler uses the professional ARMCC compiler engine, so it produces efficient code that can be used free-of-charge, even in commercial applications[6].

5.2 GSM Module (SIM 300)

Fig. 5.2 GSM SIM 300 Module (12V/2A)

Introduction:

The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit-switched network optimized for full duplex voice telephony[1]. This was expanded over time to include data communications, first by circuit-switched transport, then packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).

The GSM systems and services are described in a set of standards governed by ETSI, where a full list is maintained[2].

5.2.1 GSM SIM 300 Overview:

This GSM Modem can accept any GSM network operator SIM card and act just like a mobile phone with its own unique phone number. Advantage of using this modem will be that you can use its RS232 port to communicate and develop embedded applications. Applications like SMS Control, data transfer, remote control and logging can be developed easily[2].

The modem can either be connected to PC serial port directly or to any microcontroller. It can be used to send and receive SMS or make/receive voice calls.

It can also be used in GPRS mode to connect to internet and do many applications for data logging and control. In GPRS mode you can also connect to any remote FTP server and upload files for data logging.

This GSM modem is a highly flexible plug and play quad band GSM modem for direct and easy integration to RS232 applications. Supports features like Voice, SMS, Data/Fax, GPRS and integrated TCP/IP stack[1].

5.2.2 Wireless Communication:

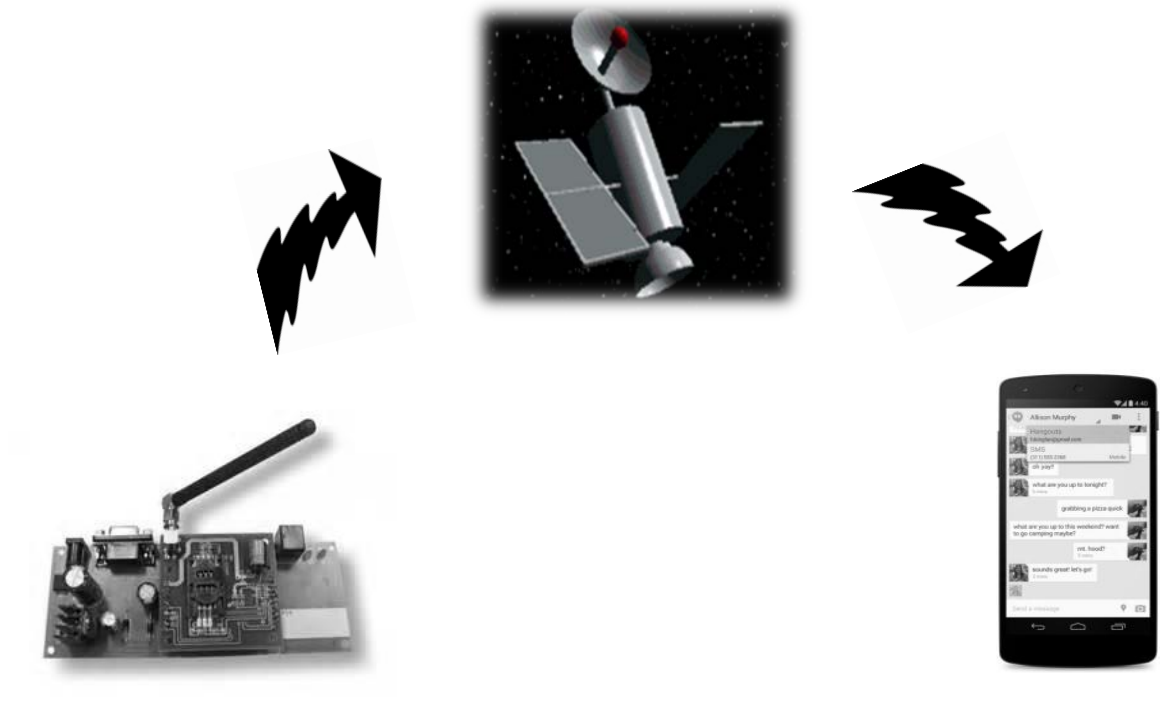


Fig. 5.2.2 Wireless Communication Overview

5.2.3 AT Commands Overview:

AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands[3].

AT commands with a GSM/GPRS MODEM or mobile phone can be used to access following information and services:

- Information and configuration pertaining to mobile device or MODEM and SIM card.
- SMS services.
- MMS services.

Commands used are as follows:

- AT+CSMP
- AT+CMGF
- AT+CMGS
- AT+CMGR
- AT+CNMI

5.2.4 AT Commands Operation:

1. AT+CSMP

Use: Set text mode parameters.

2. AT+CMGF

Use: Select SMS Message format. Format: AT+CMGF=<mode><CR><LF>Response: There are two modes a) Text mode   b) PDU mode.

3. AT+CMGS:

Response: It is used to send the message to the corresponding number.

4. AT+CMGR:

Use: Receive Message from a device capable of transmitting messages.

Response: It specifies the location of the SMS message to be read from the message storage area. It is used to read the message which is being transmitted.

5. AT+CNMI:

Use: It is used to forward the message received by the MS SIM Card to the serial port.

5.3 Five Volt Relay Module (2 Channels Active Low)

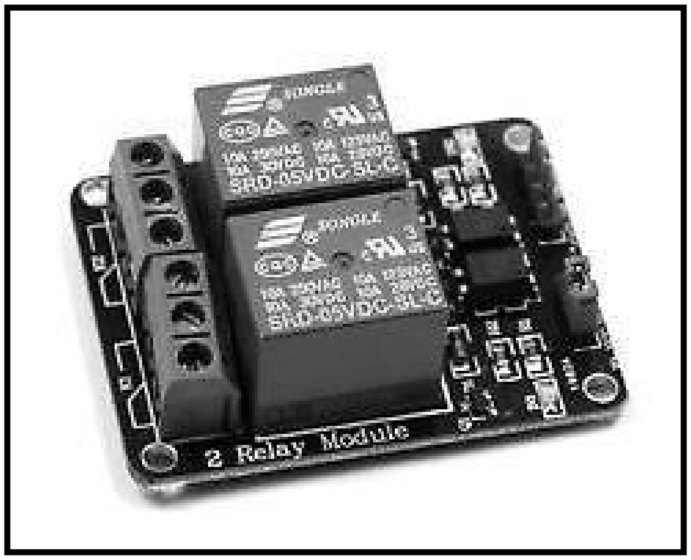


Fig. 5.3 Relay Module (5V/2 Channels)

The Relay module allows a wide range of microcontroller such as Arduino, AVR, PIC, and ARM with digital outputs to control larger loads and devices like AC or DC Motors, electromagnets, solenoids, and incandescent light bulbs. This module is designed to be integrated with 2 relays that it is capable of control 2 relays. The relay shield use one QIANJI JQC-3F high-quality relay with rated load 7A/240VAC,10A/125VAC,10A/28VDC.The relay output state is individually indicated by a light-emitting diode[2].

5.3.1 Features:

- Number of Relays: 2
- Control signal: TTL level
- Rated load: 7A/240VAC 10A/125VAC 10A/28VDC
- Contact action time: 10ms/5ms

5.3.2 Working:

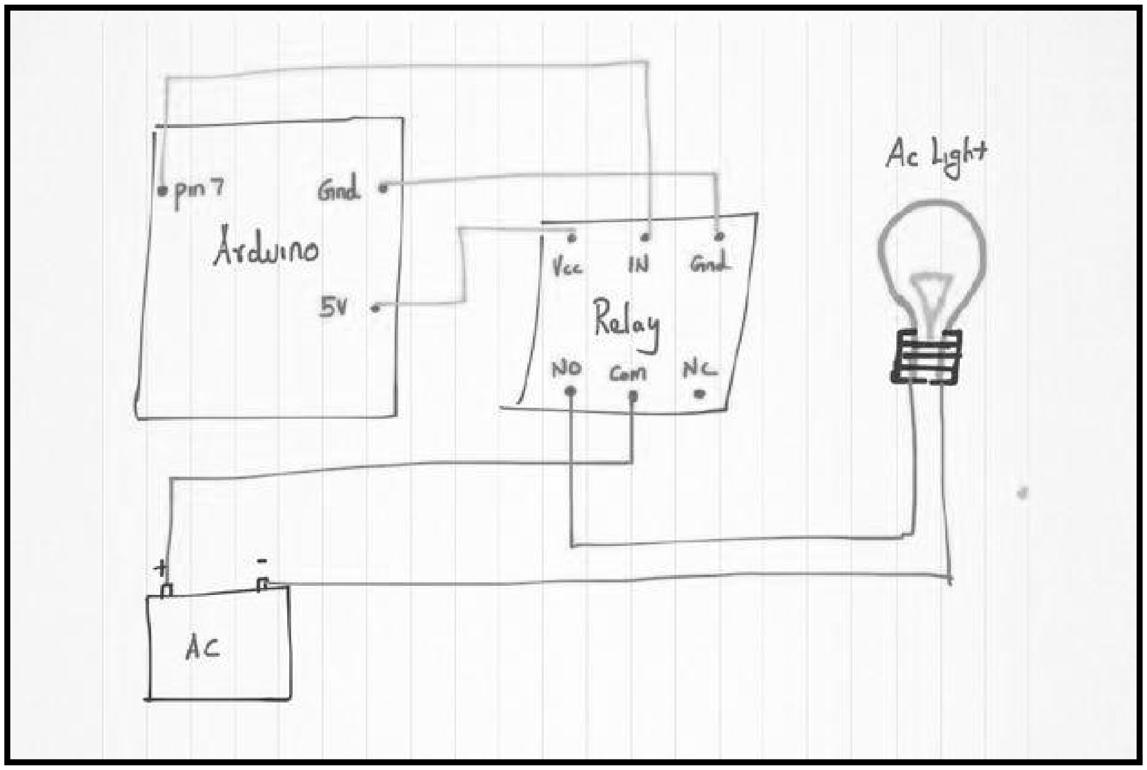


Fig. 5.3.2 Relay Module Connections and Working

Pin Description and Working:

- COM- Common pin
- NC- Normally Closed, in which case NC is connected with COM when INT1 is set low and disconnected when INT1 is high;
- NO- Normally Open, in which case NO is disconnected with COM1 when INT1 is set low and connected when INT1 is high.
- Terminal 2 is similar to terminal 1,except that the control port is INT2
- INT 1- Relay 1 control port
- INT 2- Relay 2 control port

This Active Low Relay Module (INT 1) is by default ON, as soon as the online compiled code is burned into the Microcontroller (FRDM-kl25z). With the help of GSM SIM 300 Module and Android Application, the A.C. Electric Bulb connected to the electromagnetically powered switch of the Relay is turned ON/OFF depending upon the relevant message from the Android Application which upon reception is then treated by the GSM SIM 300 Module.

5.4 Motor Driver IC L293D

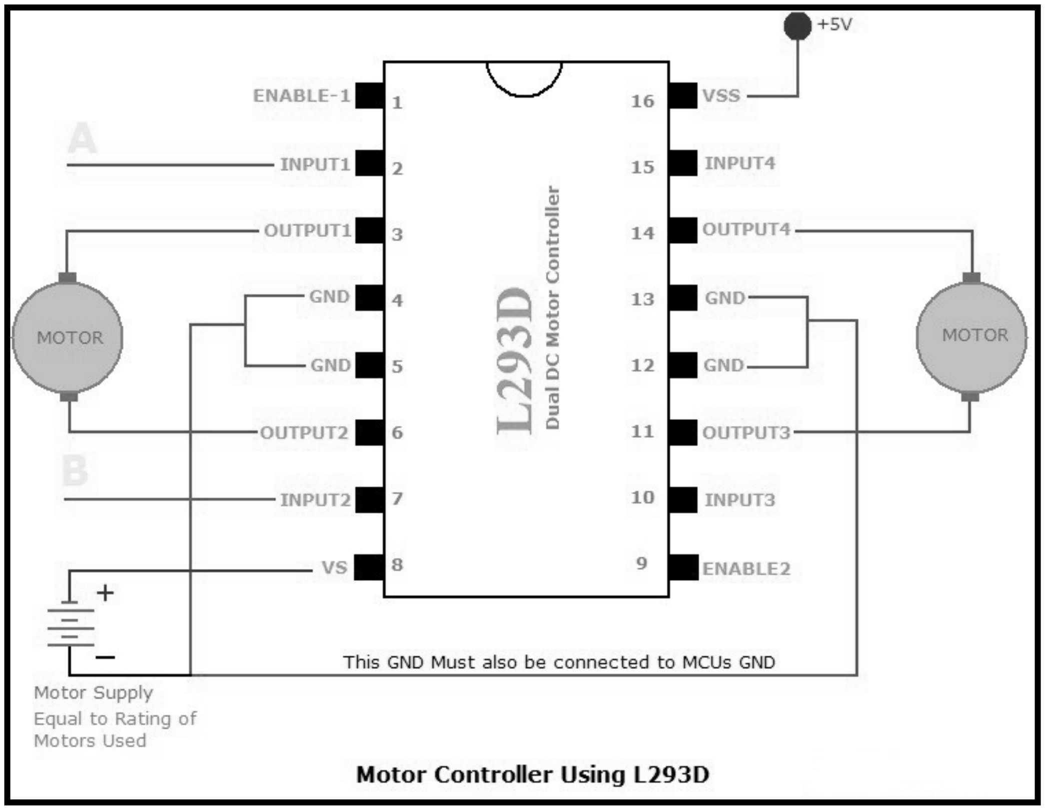


Fig. 5.4 IC L293D Motor Driver Configuration

5.4.1 General Description:

Motor gives power to your MCU and power to do physical works, for example to move your robot. So it is essential to know how to control a DC motor effectively with a MCU. We can control a DC motor easily with microcontrollers. We can start it, stop it or make it go either in clockwise or anti clock wise direction. We can also control its speed but it will be covered in latter tutorials.

5.4.2 DC Motor Understanding:

A DC motor is electromechanical device that converts electrical energy into mechanical energy that can be used to do many useful works. It can produce mechanical movement like moving the tray of CD/DVD drive in and out (you may like to try it out Go to My Computer, right click the drive icon and click "Eject").

This shows how software controls a motor. DC motors comes in various ratings like 6V and 12V. It has two wires or pins. When connected with power supply the shaft rotates. You can reverse the direction of rotation by reversing the polarity of input.

5.4.3 Control with MCUs:

As the MCUs PORT are not powerful enough to drive DC motors directly so we need some kind of drivers. A very easy and safe is to use popular L293D chips. It is a 16 PIN chip. This chip is designed to control 2 DC motors. There are 2 INPUT and 2 OUTPUT PINs for each motor.

Direction	Input A	Input B
Stop	Low	Low
Clockwise (Forward)	Low	High
Anticlockwise (Backward)	High	Low
Stop	High	High

Table 5.4.3 Control with MCU’s

5.4.4 DC Gear Motor:

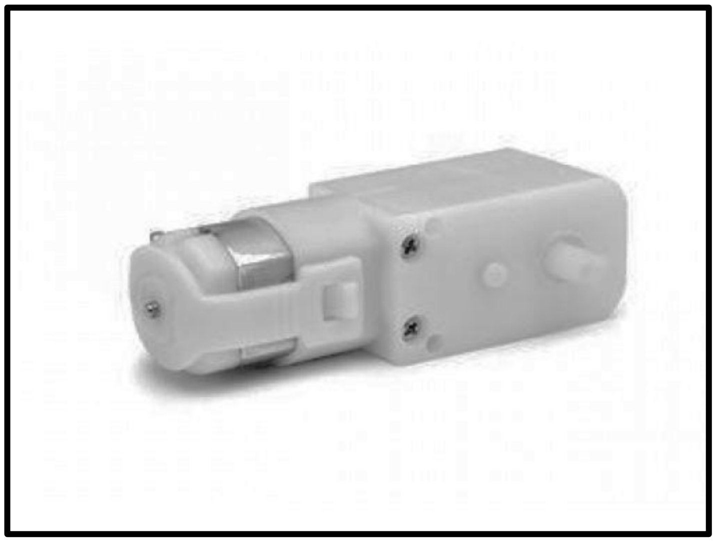


Fig. 5.4.4 9V DC Motor

5.4.5 L293D and DC Motor Assembling on Breadboard

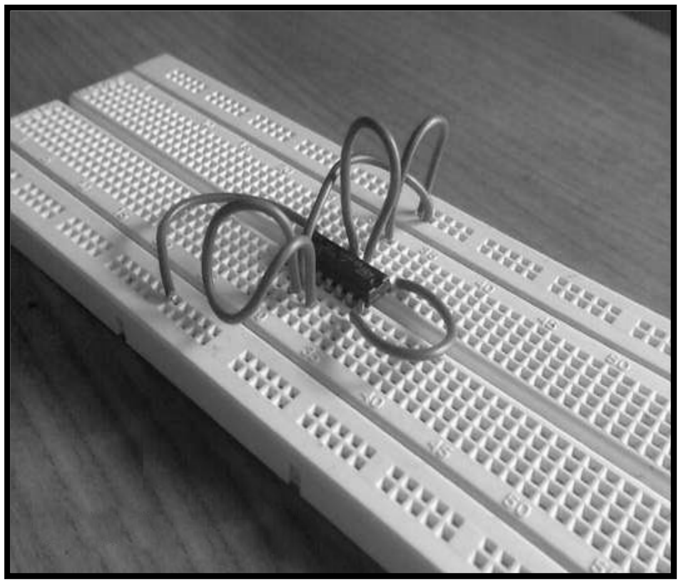


Fig. 5.4.5.1 IC L293D Assembly

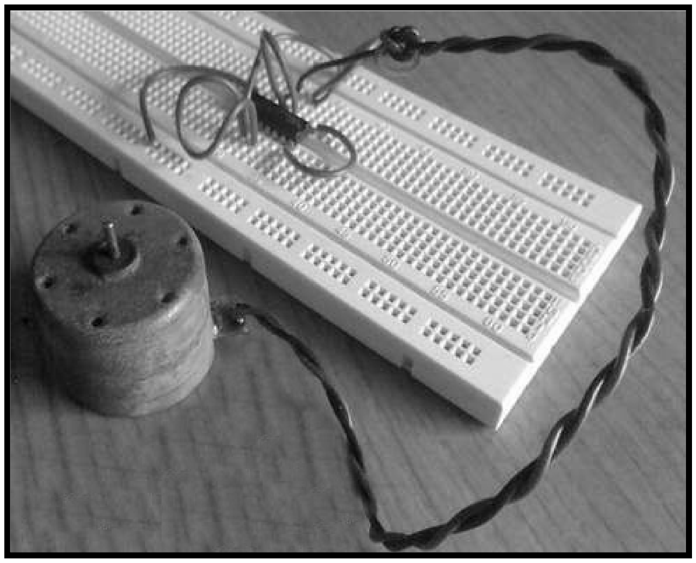


Fig. 5.4.5.2 DC Motor and IC L293D Assembly

Chapter VI
Working of Project

6.1 Software (Android Application) Development

6.1.1 Saving Data in an Android Activity:

The three entities of memory to save is divided as given below

- Saving Key Value Sets: Using a shared preferences file for storing small amounts of information in key-value pairs.
- Saving Files: Saving a basic file, such as to store long sequences of data that are generally read in order.
- Saving Data in SQL Databases called as SQLite: Using a SQLite database to read and write structured data.

Since, we have to save states of appliances like lights, fans, air-conditioners in corresponding to ON/OFF. Hence, we will use the first method of saving states using key-value sets. For this purpose, we will use 'SharedPreferences' class also called as API (Application Programming Interface)

A Shared Preferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each Shared Preferences file is managed by the framework and can be private or shared. This class shows you how to use the Shared Preferences APIs to store and retrieve simple values[5].

6.2 Shared Preferences:

You can create a new shared preference file or access an existing one by calling one of two methods:

- getSharedPreferences() — Use this if you need multiple shared preference files identified by name, which you specify with the first parameter. You can call this from any Context in your app.
- getPreferences() — Use this from an Activity if you need to use only one shared preference file for the activity. Because this retrieves a default shared preference file that belongs to the activity, you don't need to supply a name.

For example, the following code is executed inside a Fragment. It accesses the shared preferences file that's identified by the resource string R.string.preference_file_key and opens it using the private mode so the file is accessible by only your app.

When naming your shared preference files, you should use a name that's uniquely identifiable to your app, such as "com.example.myapp.PREFERENCE_FILE_KEY". Alternatively, if you need just one shared preference file for your activity, you can use the getPreferences() method:

SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);

Also, if you create a shared preferences file with MODE_WORLD_READABLE or MODE_WORLD_WRITEABLE, then any other apps that know the file identifier can access your data.

6.2.1 Creating Objects of Shared Preferences:

As stated above, SharedPreferences stores key-value sets in multiple or single user-named XML files to avoid confusion of creating an XML file and placing it in the project folder properly and not hidden. Below is the description of the method in class Preferencemanager:

public static SharedPreferences getDefaultSharedPreferences (Context context);

Gets a SharedPreferences instance that points to the default file that is used by the preference framework in the given context[5].

Parameters:

- context -The context of the preferences whose values are wanted.
- Returns -A SharedPreferences instance that can be used to retrieve and listen to values of the preferences.

6.2.2 Write to Shared Preferences:

To write to a shared preferences file, create a SharedPreferences. Editor by calling edit() on your SharedPreferences. Pass the keys and values you want to write with methods such as putInt() and putString(). Then call commit() to save the changes.

6.2.3 Read from Shared Preferences:

To retrieve values from a shared preferences file, call methods such as getInt() and getString(), providing the key for the value you want, and optionally a default value to return if the key isn't present.

6.3 Working of Android Application

The application consists of different sequential screens individually called as ‘Activity’. The different Activities or java source code files are as follows:

- MainActivity.java
- Fans.java
- Lights.java
- ACS.java
- Drawer.java
- Member.java
- Number.java
- Myalyout.java

6.3.1 Activity 1 (Screen 1): MainActivity.java:

The first Activity is a typical Authentication screen. The username and password need to be given in the Edittext or space provided. By default the username is ‘a’ and password is ‘b’.

If anytime the username or password is incorrect then on pressing ‘Login’ button, it will show “ACCESS DENIED”. This ensures that only authentic person can access the appliances of home.

6.3.2 Activity 2 (Screen 2) – Drawer.java:

The second Activity is a ListView in Android which displays the list of Fans, Lights, Air-conditioners and also an additional feature of looking at all the current states of appliances at a glance and also to change/alter the states instantly.

On this screen, the user can also discover 2 options on the options panel (seen on upper right space of screen for Android version 3.2 and above). The two options are as follows:

- Send Message
- Saving GSM Mobile Number

6.3.3 Send Message:

When this option is clicked, no new Activity is opened but a message is sent with a acknowledgement called as ‘Toast’ in Android to the user using SMSManager class instance and by using getDefault() method. We can then use sendTextMessage () method to send message.

The generation of message is by is done by a method devised by us, this method is encoded in a very compact way such that it consists the states of all appliances.

The message that is sent consists of lower case characters starting from character ‘a’ to ‘j’. Each character represents either ON or OFF state of fan1, fan2, light1, light2, AC (air-conditioners).

- a - Signify ON state of fan1 & b- Signify OFF state of fan1
- c - Signify ON state of fan2 & d - Signify OFF state of fan2
- e - Signify ON state of light1 & f - Signify OFF state of light1
- g - Signify ON state of light2 & h - Signify OFF state of light2
- i - Signify ON state of AC & j - Signify OFF state of AC

6.3.4 Speed Control of Fan & Intensity Control of Light:

In order to change or vary the speed/intensity, a ‘Seekbar’ in Android application is added, so that the levels can be varied as per user by simple dragging along or pressing on the level of seekbar.

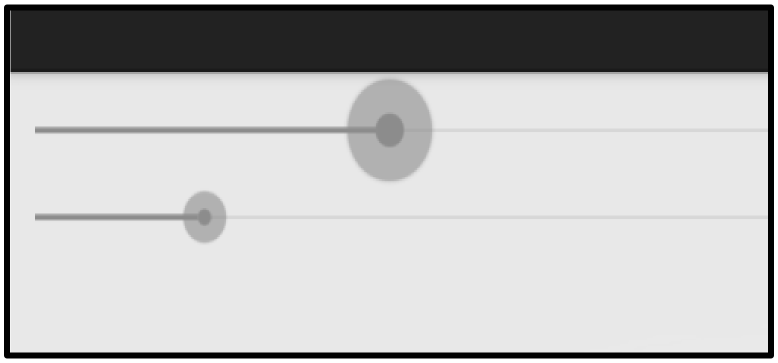


Fig. 6.3.4 Seek Bar in Android Application Environment

We have set 4 levels in seek bar which will be decoded at receiver side MBED controller board for different PWM (Pulse Width Modulation) values

- Level 0 – for 0.0f value at controller for zero (0% of duty cycle) PWM value.
- Level 1 – for 0.25f value at controller for 0.25 (25% of duty cycle) PWM value.
- Level 2 – for 0.50f value at controller for 0.5 (50% of duty cycle) PWM value.
- Level 3 – for 1.0f value at controller for 1 (100% of duty cycle) PWM value.

6.3.5 Saving GSM Mobile Number:

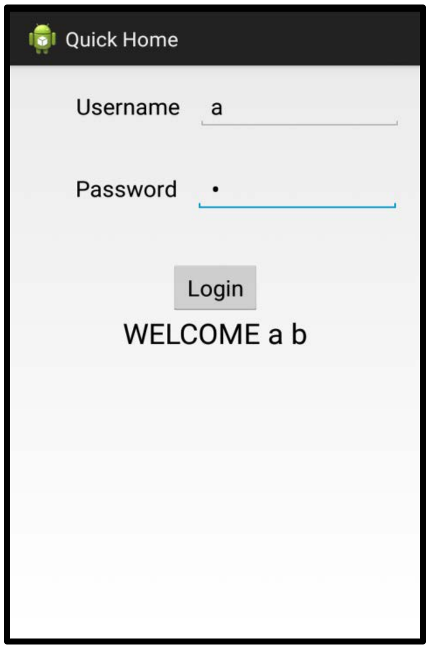
The mobile number is saved as an integer data type and passed to sendTextMessage() in Drawer.java along with the compact message generated as explained above.

User should provide proper 10- digit number and should compulsorily press "Save" button in order to save the number in application. The user is then showed the ListView screen containing appliances.

6.4 Application Screenshots

The entire Android Application was developed using Android Eclipse IDE. Some of the screenshots of the custom made Application are given below.

6.4.1. User Authentication Screen: This screen on the application ensures that only valid users in the Home Automation environment have access to the home appliances.

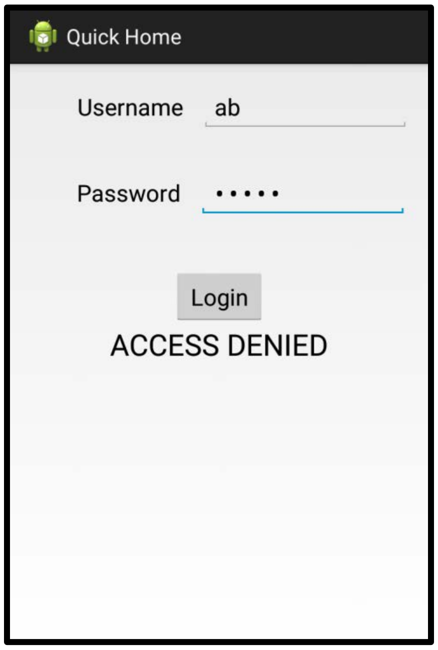


Username a

Password •

WELCOME a b

Fig. 6.4.1.1 Authentication – Access Granted



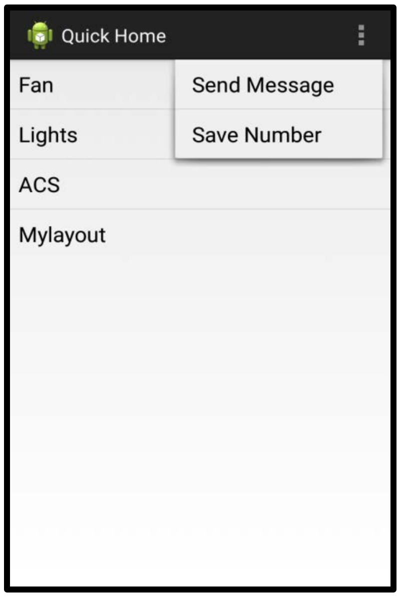
Username ab

Password •••••

ACCESS DENIED

Fig. 6.4.1.2 Authentication – Access Denied

6.4.2 List View Layout: This screenshot shows the List view of Appliances along with options of saving number and sending message to GSM Module.



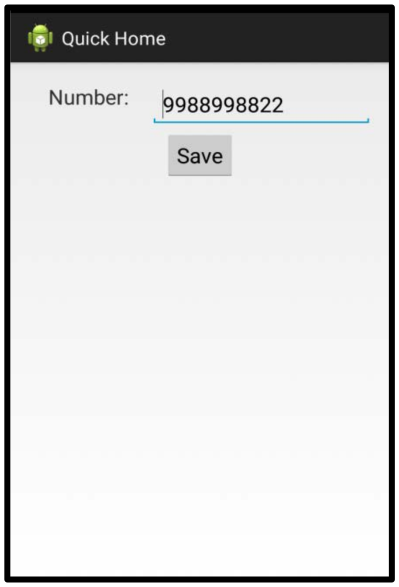


Fig. 6.4.2 List View Layout

Fig. 6.4.3 GSM SIM Card Mobile Number

6.4.3 GSM Mobile Number: This screen is displayed when the user taps on the three vertical dots which appear at the top right corner of the screen.

6.4.4 Appliance Fan Screen: The screenshot below shows the activity screen which is opened when the user taps on Fan item in the List View screen. Fan 1 is coupled with speed control whereas Fan 2 is a basic switch ON/OFF LED.

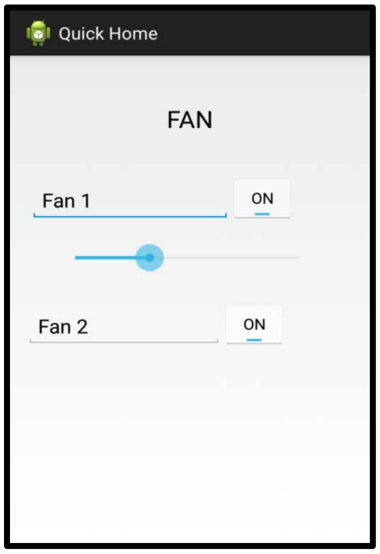


Fig. 6.4.4 Appliance Fan Screen

6.4.5 Appliance Light Screen: This screen is powered by two components. Light 1 is A.C. driven from mains (230V AC) controlled with the help of Relay Module. Light 2 is an LED Source which uses PWM for its intensity variation.

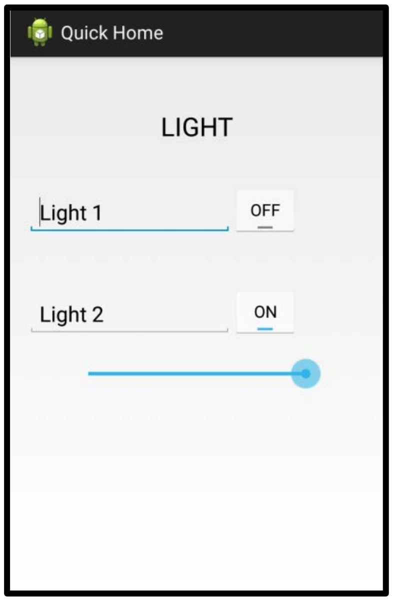


Fig. 6.4.5 Appliance Light Screen

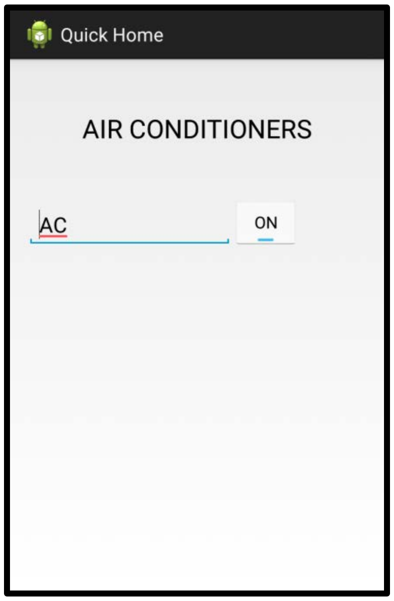


Fig. 6.4.6 Appliance AC Screen

6.4.6 Appliance AC Screen: This screen is composed of a basic LED with ON/OFF states.

6.4.7 Custom Layout Screen: This screen offers a quick view and control of all appliances on a specimen layout.

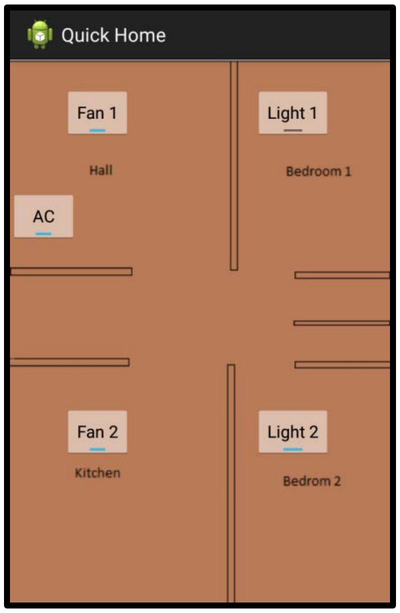


Fig. 6.4.7 Custom Layout Screen

6.5 Working of Hardware

The working of entire Home Automation project is divided into two categories:

- Software Integration
- Hardware Integration

Software Integration is already presented in the report. Hardware Integration is achieved with the help of GSM SIM 300 Module and ARM's mbed platform based FRDM-kl25z Microcontroller board along with other subsets which include Relay Module and Motor Driver IC L293D. The user interacts with the android phone and send control signal to the Microcontroller which in turn will control other embedded devices/sensors.

6.5.1 Hardware Assembly Screenshots:

The figures presented below indicate the complete assembly of hardware from different angles.

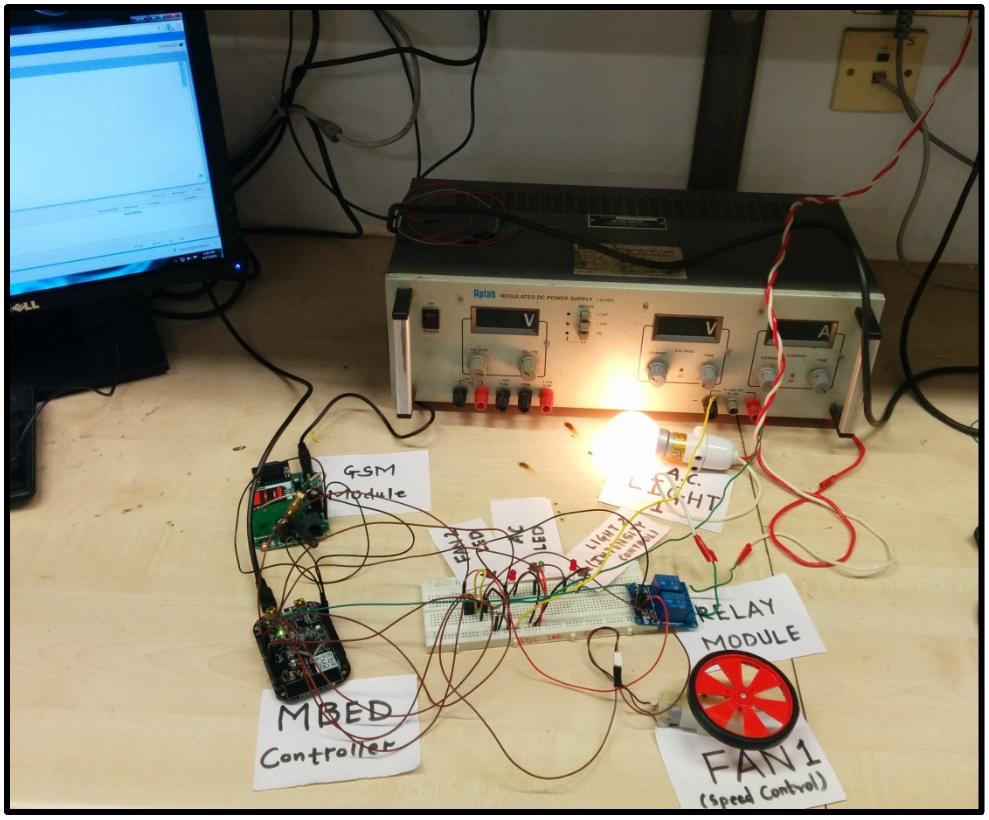


Fig. 6.5.1.1

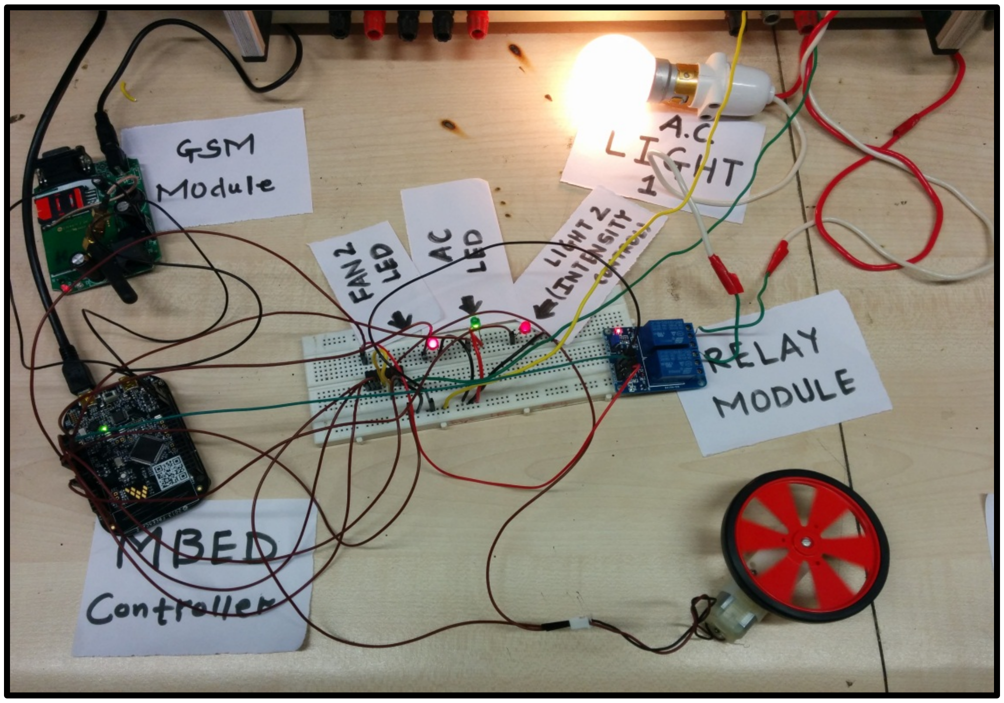


Fig. 6.5.1.2

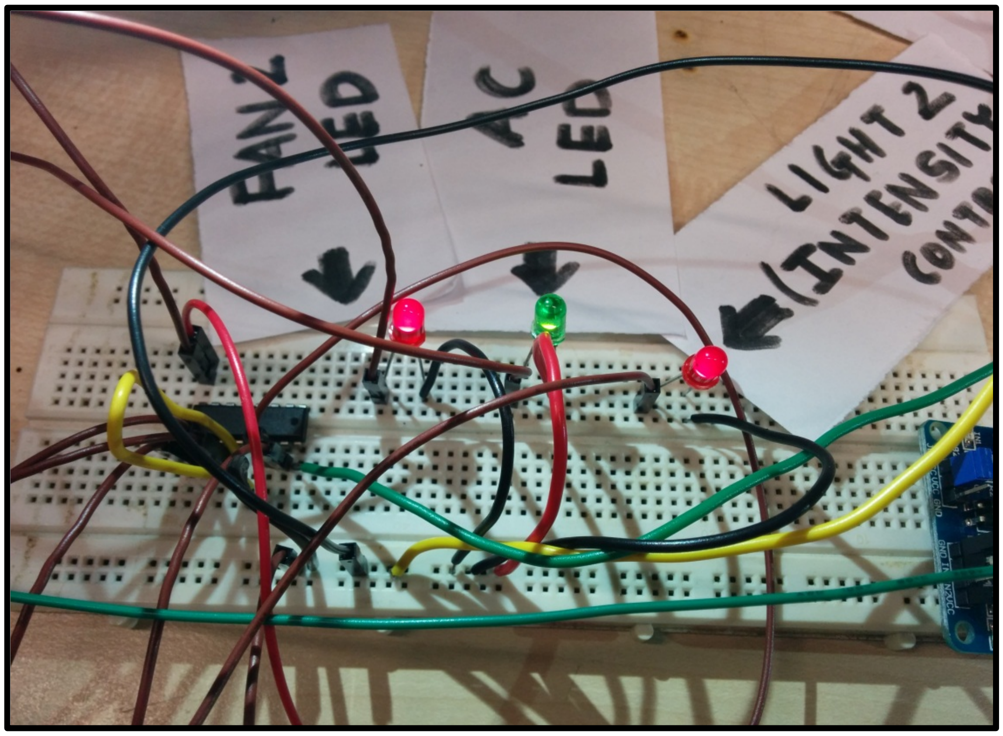
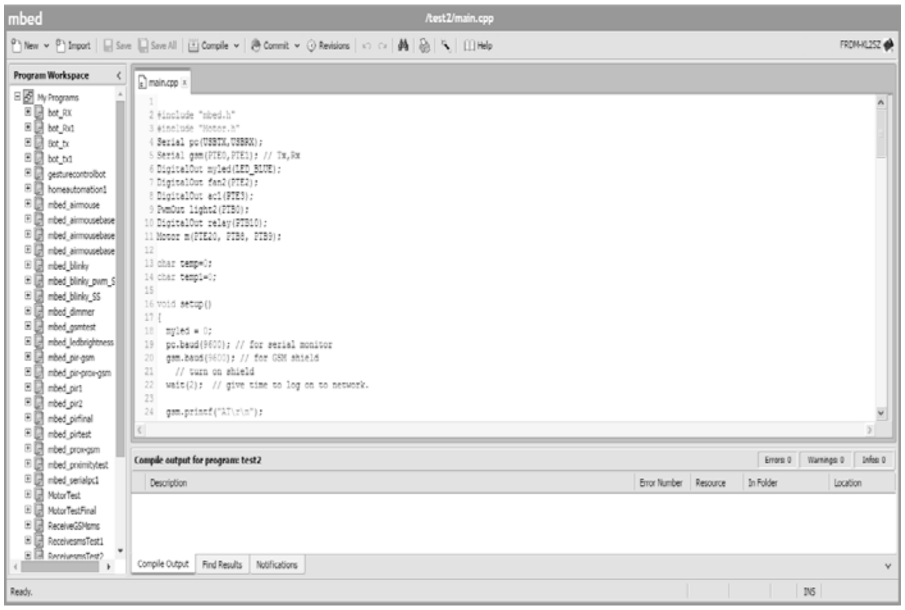


Fig. 6.5.1.3

6.6 ARM's Mbed Online Compilation

ARM's Mbed offers online compilation of the Embedded C/C++ Code written with the help of in-built API's a constructed around libraries.



```
#include "mbed.h"
#include "Motor.h"
Serial pc(USBTX,USBRX);
Serial gsm(PTE0,PTE1); // Tx,Rx
DigitalOut myled(LED_BLUE);
DigitalOut fan2(PTE2);
DigitalOut ac1(PTE3);
PwmOut light2(PTB0);
DigitalOut relay(PTB10);
Motor m(PTE20, PTB8, PTB9);

char temp=0;
char temp1=0;

void setup()
{
  myled = 0;
  pc.baud(9600); // for serial monitor
  gsm.baud(9600); // for GSM shield
    // turn on shield
  wait(2);  // give time to log on to network.

  gsm.printf("AT\r\n");
```

Fig. 6.6 ARM'S Mbed Compiler

6.7 Code Snippet

The code snippet given below gives the program structure written in Embedded C:

- #include<mbed.h>
- #include …..
- void setup()
  {
  // Intitialization instructions for GSM Module and other components.
  }
  void loop()
  {
  // The entire receive-message-and-act-action is done here.
  }
  int main()
  {
  setup(); loop(); // Program starts at setup() and ends at loop() function respectively.
  }

Chapter VII
Results & Conclusion

7.1 Results

When software works in synchronisation with Hardware, results are inevitably expected. This project was successfully implemented, debugged and tested. The result of all this work paved way for reliable output from all the hardware devices working in sync with the software application.

Home Automation System offers a great amount of deliverables to the old-aged and handicapped people. With that in mind, cost-effective and fast automation was employed. The same was clearly monitored with the help of a third party terminal application such as Hyper Terminal or Tera Term on a personal computer.

Within seconds, it was observed that the entire system collaborated and worked in sync with the messages transmitted from the Android Application which were subsequently received and acted upon by the GSM module, thereby controlling the end-user appliances at ease.

7.2 Conclusion

This is an on-going project. Our prime objective is to assist handicapped/old aged people. This report gives a basic idea of how to control various home appliances and provide a security using Android OS powered smartphone. This project is based on Android and ARM’s Mbed platform both of which are Free Open Source Software. So the overall implementation cost is very cheap and it is affordable by a common person. Looking at the current scenario we have chosen Android platform so that most of the people can get benefitted.

The design consists of Android phone with Android OS based home automation application, ARM’s Mbed based FRDM-kl25z microcontroller along with other subsets.

Chapter VIII Applications

8.1 Applications of Home Automation and Security System

Home automation can be a technological marvel. It can be as simple as a system to regulate the on/off of outdoor lighting or complex enough to raise and lower window coverings each morning and night, remember and play a favourite music playlist at your parties and monitor your home for safety from intruders.

Two main application areas of Home Automation System include:

- Life Convenience:
  Life, and sometimes traffic can make it difficult to be home the moment the kids come in the door or when the repair person shows up. And since you can’t be in two places at the same time, having the convenience of accessing your home systems remotely can be a real time saver

- Energy Management & Savings:
  When used properly many home automation products, also known as "smart" products, help you manage your home’s energy consumption. For example, we can automate our thermostat to adjust settings throughout the day based around the times someone is home or the house is empty.

Some other application areas of Home Automation System are listed below:

- Medical alert / Tele-assistance.
- Precise and safe blind control.
- Detection of fire, gas leaks and water leaks.
- Smoke detector can detect a fire or smoke condition, causing all lights in the house to blink to alert any person of the house to the possible emergency.
- The system can call the home owner on their mobile phone to alert them, or call the fire department or alarm monitoring company.
- In terms of lighting control, it is possible to save energy when hours of wasted energy in both residential and commercial applications by auto on/off light at night time in all major city office buildings, say after 10pm.

Chapter IX
Future Scope

9.1 Future Scope

Looking at the current situation we can build cross platform system that can be deployed on various platforms like iOS, Windows. Limitation to control only several devices can be removed by extending automation of all other home appliances. Security cameras can be controlled, allowing the user to observe activity around a house or business. Security systems can include motion sensors that will detect any kind of unauthorized movement and notify the user. Scope of this project can be expanded to many areas by not restricting to only home. It will be flexible to support various wired as well as wireless technologies like Bluetooth, ZigBee, Wi-Fi, World Wide Web.

Apart from the above mentioned areas of future scope, Home Automation System can also include following services in the near future:

- It is possible to control and integrate security systems and also the potentially central lock all perimeter doors and windows.
- Security cameras could be controlled, allowing the user to observe activity around a house or business right from a Monitor or touch panel.
- An intercom system could allow communication via a microphone and loud speaker between multiple rooms.
- Complete home automation systems have many security benefits. As mentioned, they allow us to check in on our home from a remote location, giving us true peace of mind. Some systems will let us interact with your home security system, providing us with the ability to arm and disarm our home remotely. Some complete home automation systems will alert us by phone, text or email if there is any unusual movements within your home.

Chapter X Bibliography & References

10.1 Bibliography

1. www.developers.android.com
2. www.mbed.org
3. www.stackoverflow.com
4. www.google.co.in
5. www.en.wikipedia.org

10.2 References

[1] Neng- Shiang Liang; Li-Chen Fu; Chao-Lin Wu. "An integrated, flexible, and Internet-based control architecture for home automation system in the internet era". Proceedings ICRA' 02. IEEE International Conference on Robotics and Automation, Vol. 2, pp.1101-1106, 2002.

[2] K.Tan, T.Lee and C.YeeSoh. "Internet-Based Monitoring of Distributed Control Systems-An Undergraduate Experiment". IEEE Transaction on Education, Vol. 45, No. 2, May 2002.

[3] Al-Ali, Member, IEEE & M. AL-Rousan. "Java-Based Home Automation System R." IEEE Transactions on Consumer Electronics, Vol. 50, No. 2, MAY 2004.

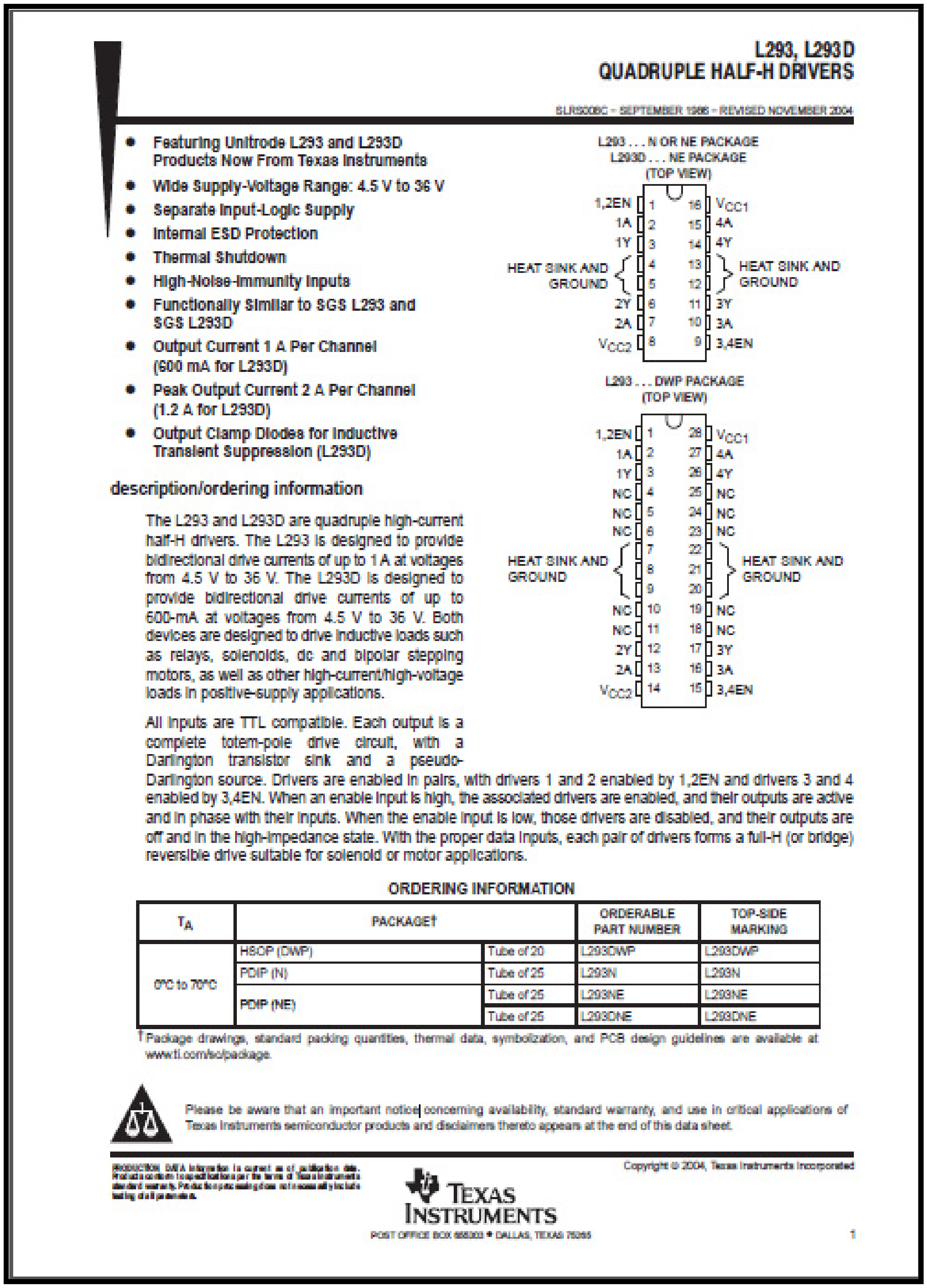
[4] R.Piyare, M.Tazil" Bluetooth Based Home Automation System Using Cell Phone", 2011 IEEE 15th International Symposium on Consumer Electronics.

[5] Official Android Developers Website – http://www.developers.android.com

[6] ARM Mbed Development Website – http://www.developer.mbed.org

Chapter XI
Appendix

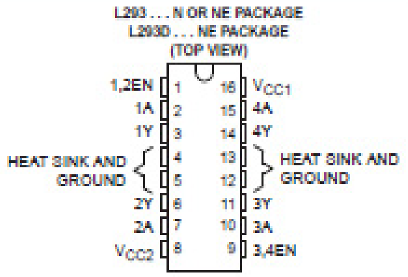
A. Motor Driver IC L293D Datasheet

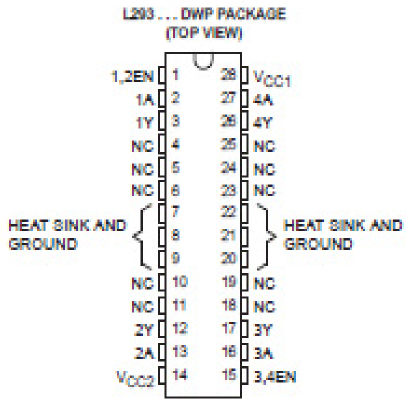


L293, L293D
QUADRUPLE HALF-H DRIVERS

SLRS008C – SEPTEMBER 1986 – REVISED NOVEMBER 2004

- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)





description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	HSOP (DWP)	Tube of 20	L293DWP	L293DWP
	PDIP (N)	Tube of 25	L293N	L293N
	PDIP (NE)	Tube of 25	L293NE	L293NE
		Tube of 25	L293DNE	L293DNE

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

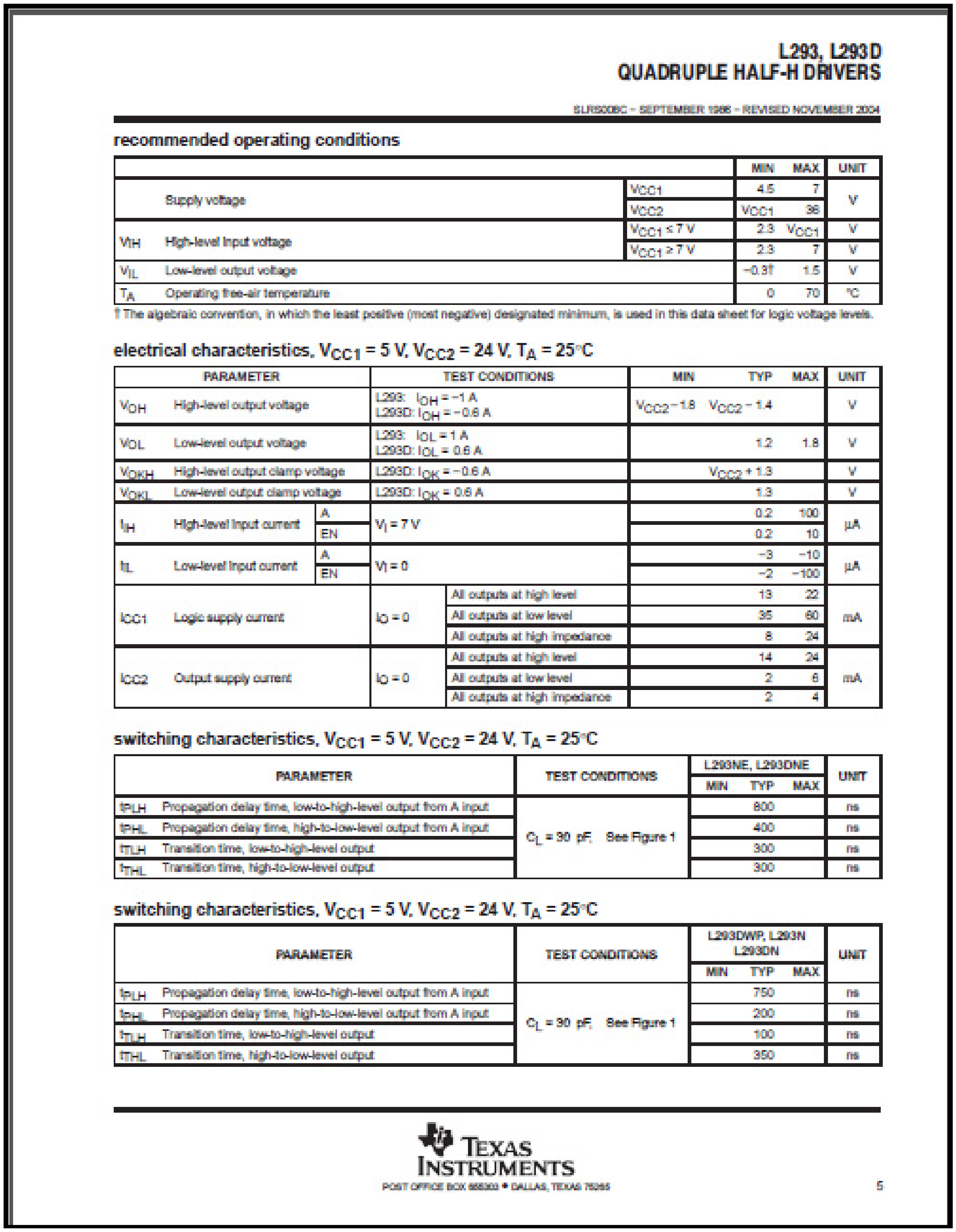
Copyright © 2004, Texas Instruments Incorporated

TEXAS INSTRUMENTS

POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

1

Fig. A.1



L293, L293D
QUADRUPLE HALF-H DRIVERS

SLRS008C – SEPTEMBER 1986 – REVISED NOVEMBER 2004

recommended operating conditions

			MIN	MAX	UNIT
	Supply voltage	V_{CC1}	4.5	7	V
		V_{CC2}	V_{CC1}	36	
V_{IH}	High-level input voltage	$V_{CC1} \le 7$ V	2.3	V_{CC1}	V
		$V_{CC1} \ge 7$ V	2.3	7	V
V_{IL}	Low-level output voltage		−0.3†	1.5	V
T_A	Operating free-air temperature		0	70	°C

† The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

electrical characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = −1 A L293D: I_{OH} = −0.6 A		V_{CC2} − 1.8	V_{CC2} − 1.4		V
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A L293D: I_{OL} = 0.6 A			1.2	1.8	V
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = −0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	μA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			−3	−10	μA
		EN				−2	−100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293NE, L293DNE			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		800		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			400		ns
t_{TLH}	Transition time, low-to-high-level output			300		ns
t_{THL}	Transition time, high-to-low-level output			300		ns

switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293DWP, L293N L293DN			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		750		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			200		ns
t_{TLH}	Transition time, low-to-high-level output			100		ns
t_{THL}	Transition time, high-to-low-level output			350		ns

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

5

Fig. A.2