INTERNET OF THINGS FOR SMART CITY

A MINI PROJECT REPORT

Submitted by

K.V.KANDHA PRASATH	**Register No.:13BEC061**
K.NISHANTH	**Register No.:13BEC096**
L.NAVEEN	**Register No.:13BEC218**
K.SUDHAKAR	**Register No.:13BEC233**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641049

ANNA UNIVERSITY: CHENNAI

APRIL 2017

BONAFIDE CERTIFICATE

Certified that this project report **"INTERNET OF THINGS FOR SMART CITY"** is the bonafide work of **Mr K.V.KANDHAPRASATH [13BEC061], Mr K.NISHANTH [13BEC096], Mr L.NAVEEN [13BEC218] and Mr K.SUDHAKAR [13BEC233]** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr S.Umamaheswari M.E., Ph.D.,	Dr.K.Malarvizhi M.E., Ph.D,
PROJECT SUPERVISOR	**HEAD OF THE DEPARTMENT**
Department of ECE	Department of ECE
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore-641049	Coimbatore- 641049

The candidates with Register No: 13BEC061, 13BEC096, 13BEC218 and 13BEC233 are examined by us in the project viva-voce examination held on ____________________

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

ACKNOWLEDGEMENT

We express our sincere thanks to the Management of Kumaraguru College of Technology and Joint Correspondent **Shri.ShankarVanavarayar** for the kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr.R.S.Kumar, Ph.D.,**Kumaraguru College of Technology, who encouraged us in each and every step of the project.

We would like to thank **Dr.K.Malarvizhi, Ph.D.,** Head of the Department, Electronics and Communication Engineering, for her kind support and for providing necessary facilities to carry out the project work.

We wish to thank with everlasting gratitude to our Project Coordinator**Dr.A.Vasuki, Ph.D.,** Department of Electronics and Communication Engineering
for her consistent support throughout the course of this project work.

We are greatly privileged to express our deep sense of gratitude and heartfelt thanks to our Project Guide **Dr S.Umamaheswari Ph.D.,** Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success and also we wish to convey our regards to all teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support and abundant blessings in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

ABSTRACT

In order to reduce wastage of electricity and manual switching of the lights a solution is given which uses an application to control the switching and to check the health status of the street lights. This project recommends a SMS based intimation to the higher authority from the user. It also helps the authority and the user to check the health status of the bulbs. Two bulbs are being placed in a single post where one is always on with low intensity and the other bulb is on whenever an object is sensed by the ultrasonic sensor. In this method we are able to control the switching of the lights, check the health status of the bulbs, and to prevent the wastage of electricity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

RFID	**Radio Frequency Identification**
GSM	**Global System for Mobile**
GPS	**Global Positioning System**
GPRS	**General Packet Radio Service**
IR	**InfraRed**
LED	**Light Emitting Diode**
LCD	**Liquid Crystal Display**
TX	**Transmit**
RX	**Receive**
NPN	**Negative Positive Negative**
DC	**Direct Current**
AC	**Alternative Current**
USB	**Universal Serial Bus**
UART	**Universal Asynchronous Receiver Transmitter**
PWM	**Pulse Width Modulation**
NC	**Normally Closed**
NO	**Normally Opened**
IO	**Input/Output**
SRAM	**Static Random Access Memory**
EEPROM	**Electrically Erasable Programmable Read Only Memory**
MB	**Mega Bytes**
KB	**Kilo Bytes**

RTC	**Real Time Clock**
TCP/IP	**Transmission Control Protocol/Internet Protocol**
SIM	**Subscriber Identity Module**
TV	**Tele Vision**
DVD	**Digital Video Disk**
OS	**Operating System**
PLM	**Power Line Modem**
ICE	**In Circuit Emulator**
JVM	**Java Virtual Machine**
SPI	**Serial Peripheral Interface**
ICSP	**InCircuit Serial Programming**
TTL	**Transistor Transistor Logic**
I2C	**Inter Integrated Circuit**
ASCII	**American Standard Code for Information Interchange**
GNSS	**Global Navigation Satellite System**
CMOS	**Complementary Metal Oxide Semiconductor**
EGSM	**Extended Global System for Mobile**
PCM	**Pulse Code Modulation**
PCB	**Printed Circuit Board**
IC	**Integrated Circuit**
RF	**Radio Frequency**
DPDT	**Double Pole Double Throw**
SPDT	**Single Pole Double Throw**

M2M	**Machine To Machine**
SMT	**Surface Mount Technology**
PC	**Personal Computer**
PDA	**Personal Digital Assistant**

1. INTRODUCTION

As a matter of fact, the trending demand of alternative sources of energy is required to the growing demands of the people. This can be achieved in two ways:(1)Finding an alternative resource to supply the power, and (2)By reducing the energy consumption of the present resources available .This project ,hereby, supports the second statement. This document report provides details about the study and implementation of streets with controlled lighting based on microcontroller. Proper program and maintenance has been implemented to every combination of Street lights in all particular streets. Heavy traffic conditions are also considered accordingly.

The objective of this project is to provide lighting to the streets such that minimum possible power is consumed. To replace the conventional halogen lamps with the power LED's in the lighting system. In this Smart Street light system ,they light gets ON once the vehicle is detected by the ultrasonic sensor placed between two street lamp posts, otherwise the low voltage light glows. The health status of the light is calculated so that it can be changed and maintained at the right time. Android application has been developed inorder for Switching of Lamps by GSM and the can be controlled by the application developed. This application will be having two login Interface, one for user and other for administrator where admin operates the system and the user can register complaint therefore that will resolved as soon as by the administrator.

The idea of this project can be implemented in a large scale in many big cities ,where most of the street lights are consuming useful power .It is a step forward to allot the power generated in a much better and systematic fashion across the roads. This is a sincere effort in reducing the amount of energy wasted which can be used for other purposes.

2. HARDWARE DESCRIPTION

2.1 TABLE

LDR SENSOR	To sense the streetlight when ON and OFF
Oscillator	To obtain the create an pulse
SIMcom 800 GSM Module	To transmit and receive the GSM messages.
Driver circuit	Is used to combine two hardware components
HC SR04 Ultrasonic sensor	To detect objects under the lamp post
microcontroller (16F877A)	To control an all the hardware parts
Amplifier	To used an boost up an input signal
modem	To uses through an app and gsm module
LCD 16 x 2	For analysing purpose.
Relay 12v	To turn off the ignition of the light when day time is detected on the light.
Power supply (12v/5v)	To give the supply voltage for all of the above DC components.

Table:01

2.2 PROJECT DESCRIPTION

The main aim of the project is to Save an electricity and time. Humans are prone to being careless and because of that it might result is wastage of time and electricity. In this fast world where everything around us is happening so swift manual power , the probability of time and electricity is wastage is increasing in massive. There is two light in a lamp post. One is low beam and other is high beam. whenever the motion is detected by the ultrasonic sensor, the high beam will glow otherwise low beam continuously glow using android application. when the lamp light is failed to glow then alert message will sent to the user automatically . The user can forward the message to higher authority. If the light glows under the dead level(depends on the threshold value), the message alert will sent to user.. Anybody can have an user ID to log in and check for the health status and complain if there is no proper maintenance. Authority is able to have an overall control of the street lights and its condition from wherever they are.

Hence we have come up with a solution to overcome this huge wastage of time and electric power.

3.BLOCK DIAGRAM

This is the block diagram of the project that we intend to implement. Block diagram is shown in Figure 1.

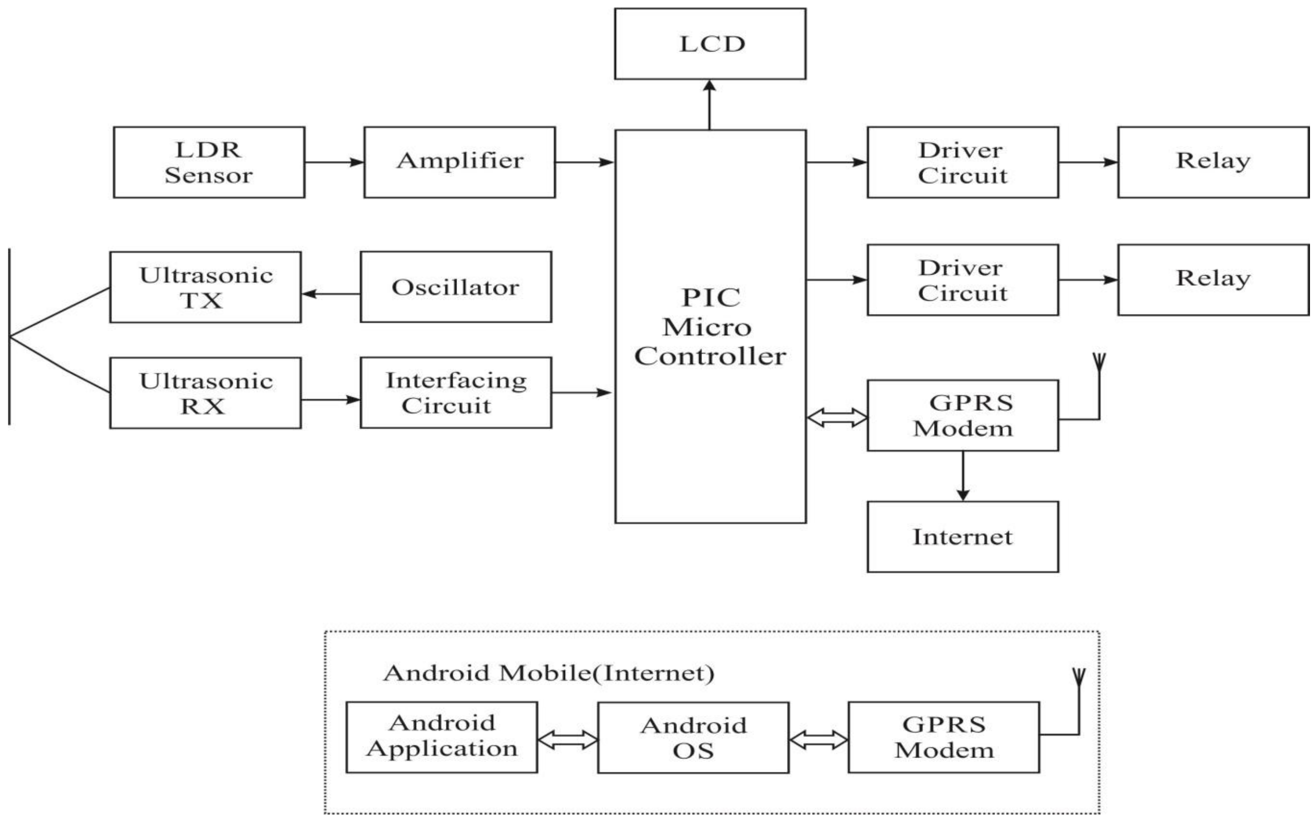

Figure 1:Block Diagram Of IOT based Street Light System

3.1 LDR(LIGHT DEPENDENT RESISTOR):

A photo resistor or **LIGHT DEPENDENT RESISTOR** or cadmium sulphide cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referred to as a photoconductor

A photo resistor is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the band is shown in figure 2. The resulting free electron (and its hole partner) conduct electricity, thereby lowering resistance.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, e.g. silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire band gap.

Extrinsic devices have impurities, also called dopants, added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (i.e., longer wavelengths and lower frequencies) are sufficient to trigger the device.

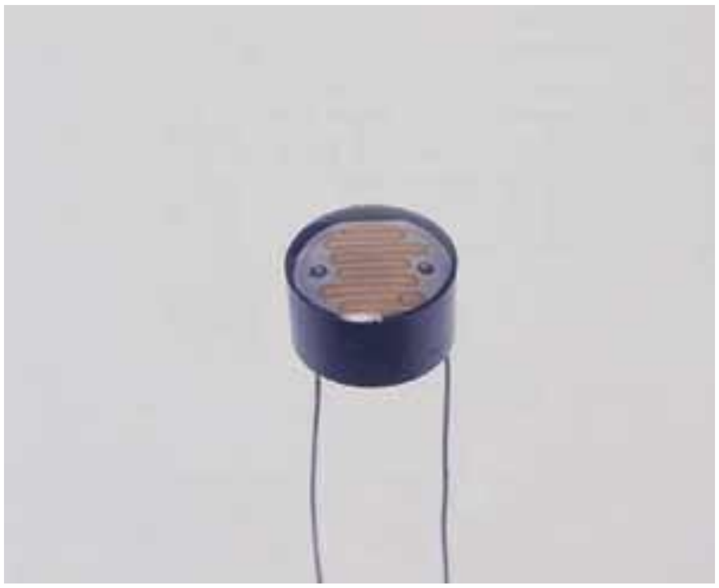

Figure 2: LDR Diagram

Everything has an electrical resistance, some more than others. An LDR will have a resistance that varies according to the amount of visible light that falls on it. The LDR is a light detecting resistor, which output voltage varies depends on the environmental light, it is connected with comparator for compare the reference voltage and input voltage, when the light is dark, the LDR sends low voltage to the comparator, so the reference voltage and LDR output voltage are not matched with each other, then the comparator triggers a high pulse from its output pin(Figure 3). If the environmental light is in normal power the LDR output voltage will be equal to the reference voltage then the comparator triggers a low pulse.

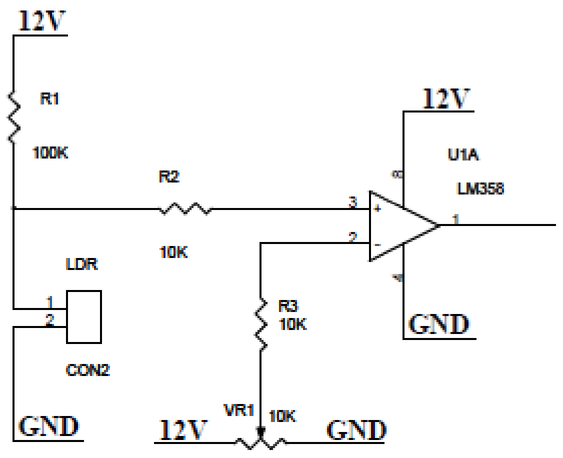

Figure 3: LDR COMPARATOR CIRCUIT

The light falling on the brown zigzag lines on the sensor, causes the resistance of the device to fall. This is known as a negative co-efficient

3.2. ULTRA SONIC SENSOR:

Ultrasonic sensors (also known as transceivers when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water.

To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical , burglar alarms and non-destructive testing.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 20,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed. shown in Figure 4 .

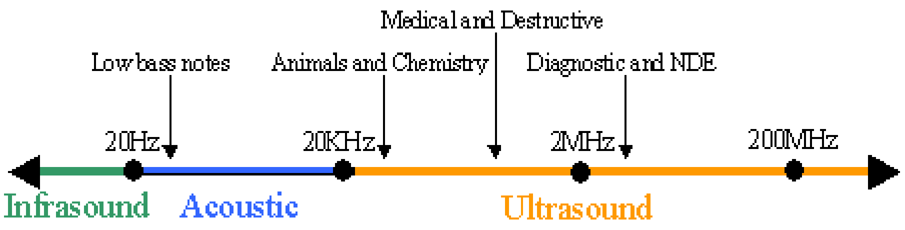

Figure 4:Ultrasonic Sensor Test

Ultrasonic refers to any study or application of sound waves that are higher frequency than the human audible range. Music and common sounds that we consider pleasant are typically 12 kHz or less, while some humans can hear frequencies up to 20 kHz. Ultrasonic waves consist of frequencies greater than 20 kHz and exist in excess of 25MHz. Ultrasonic waves are used in many applications including plastic welding, medicine, jewellery cleaning, and non-destructive test. Within non-destructive test, ultrasonic waves give us the ability to “see through” solid/opaque material and detect surface or internal flaws without affecting. .

3.2.2BASICS OF ULTRASONIC TEST

Ultrasonic wavelengths are on the same order of magnitude as visible light, giving them many of the same properties of light. For example, ultrasonic wavelengths can be focused, reflected, and refracted. Ultrasonic waves are transmitted by high frequency particle vibrations, and can be transmitted through air, water, and solids such as steel. These waves are transmitted in homogenous solid objects much like pointing a flashlight around a room with various objects that reflect light. The directed energy in an ultrasonic wave is reflected by boundaries between materials regardless of whether the material is gas, liquid, or solid. Ultrasonic waves are also reflected by any cracks or voids in solid materials. These reflected waves, which are caused by internal defects, can be compared to the reflected waves from the external surfaces, enabling the size and severity of intern defects to be identified.

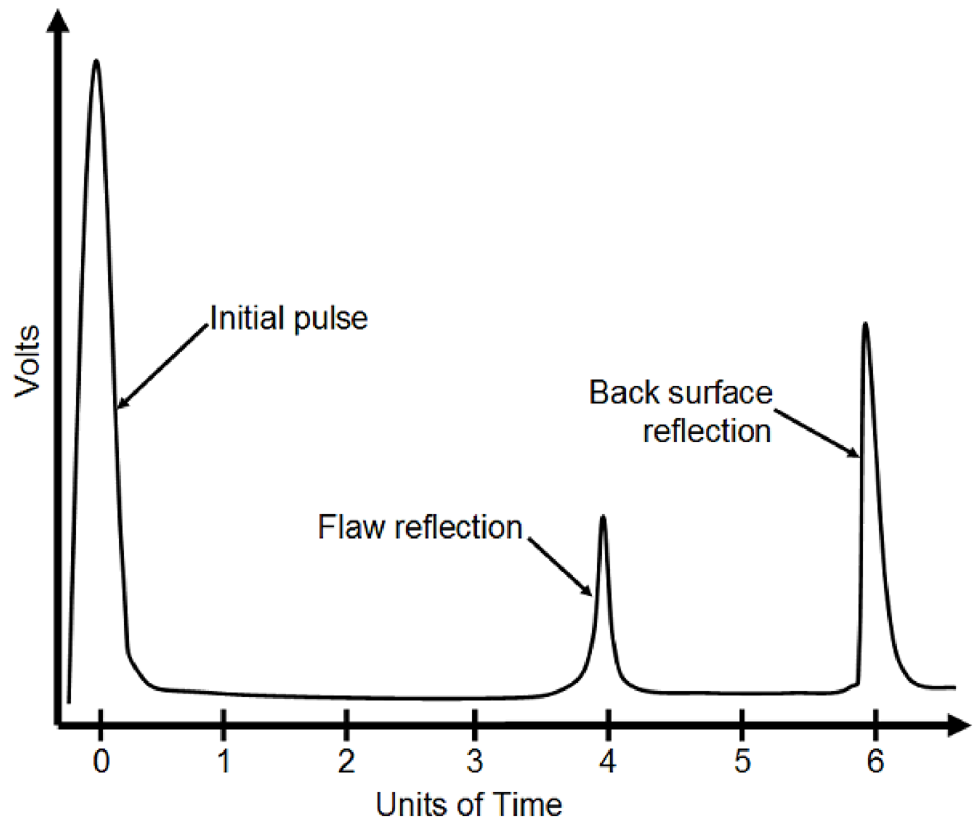

Figure 5: Ultrasonic Measure

Ultrasonic beam refraction and mode conversion is comparable to light as it passes from one medium to another. Remember how the straw in the glass of water looks broken if observed from the side.The same phenomenon occurs with ultrasonic waves as they are passed into a UUT. The Figure 5 depicts an ultrasonic transducer that transmits an ultrasonic wave through water into a block of steel. Because the direction of the ultrasonic wave is at a 90 degree angle with the surface of the steel block, no refraction occurs and the L-wave is preserved. As the angle of the ultrasonic transducer continues to increase, L-waves move closer to the surface of the UUT. The angle at which the L-wave is parallel with the surface of the UUT is referred to as the first critical angle. This angle is useful for two reasons.

3.3 AMPLIFIER:

An ELECTRONIC AMPLIFIER is a device for increasing the power of a signal. It does this by taking energy from a power supply and controlling the output to match the input signal shape but with a larger amplitude. In this sense, an amplifier may be considered as modulating the output of the power supply.

Here we use inverting amplifier as a gain amplifier. We can change the gain by adjusting the value of feedback resistance value.

As the open loop DC gain of an operational amplifier is extremely high we can afford to lose some of this gain by connecting a suitable resistor across the amplifier from the output terminal back to the inverting input terminal to both reduce and control the overall gain of the amplifier. This then produces and effect known commonly as Negative Feedback, and thus produces a very stable Operational Amplifier system.

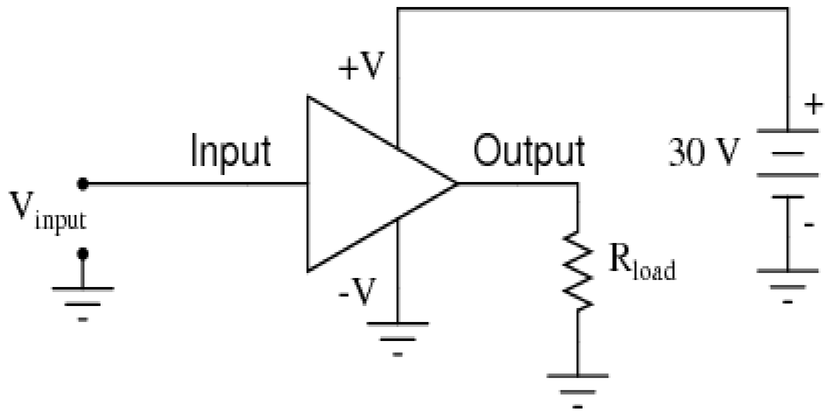

Figure 6: **OPERATIONAL AMPLIFIER**

Negative Feedback is the process of "feeding back" some of the output signal back to the input, but to make the feedback negative we must feed it back to the "Negative input" terminal using an external Feedback Resistor called Rf(Figure 6). This feedback connection between the output and the inverting input terminal produces a closed loop circuit to the amplifier resulting in the gain of the amplifier now being called its Closed-loop Gain.

As the open loop DC gain of an operational amplifier is extremely high we can afford to lose some of this gain by connecting a suitable resistor across the amplifier from the output terminal back to the inverting input terminal to both reduce and control the overall gain of the amplifier.

3.4 PIC MICROCONTROLLER(16F877A):

The microcontroller that has been used for this project is from PIC series. PIC microcontroller is the first RISC based microcontroller fabricated in CMOS (complementary metal oxide semiconductor) that uses separate bus for instruction and data allowing simultaneous access of program and data memory.

The main advantage of CMOS and RISC combination is low power consumption resulting in a very small chip size with a small pin count. The main advantage of CMOS is that it has immunity to noise than other fabrication techniques.

The PIC start plus development system from microchip technology provides the product development engineer with a highly flexible low cost microcontroller design tool set for all microchip PIC micro devices. The picstart plus development system includes PIC start plus development programmer and mplab ide.

The PIC start plus programmer gives the product developer ability to program user software in to any of the supported microcontrollers. The PIC start plus software running under provides for full interactive control over the programmer.

3.4.1 CORE FEATURES :

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
- DC - 200 ns instruction cycle

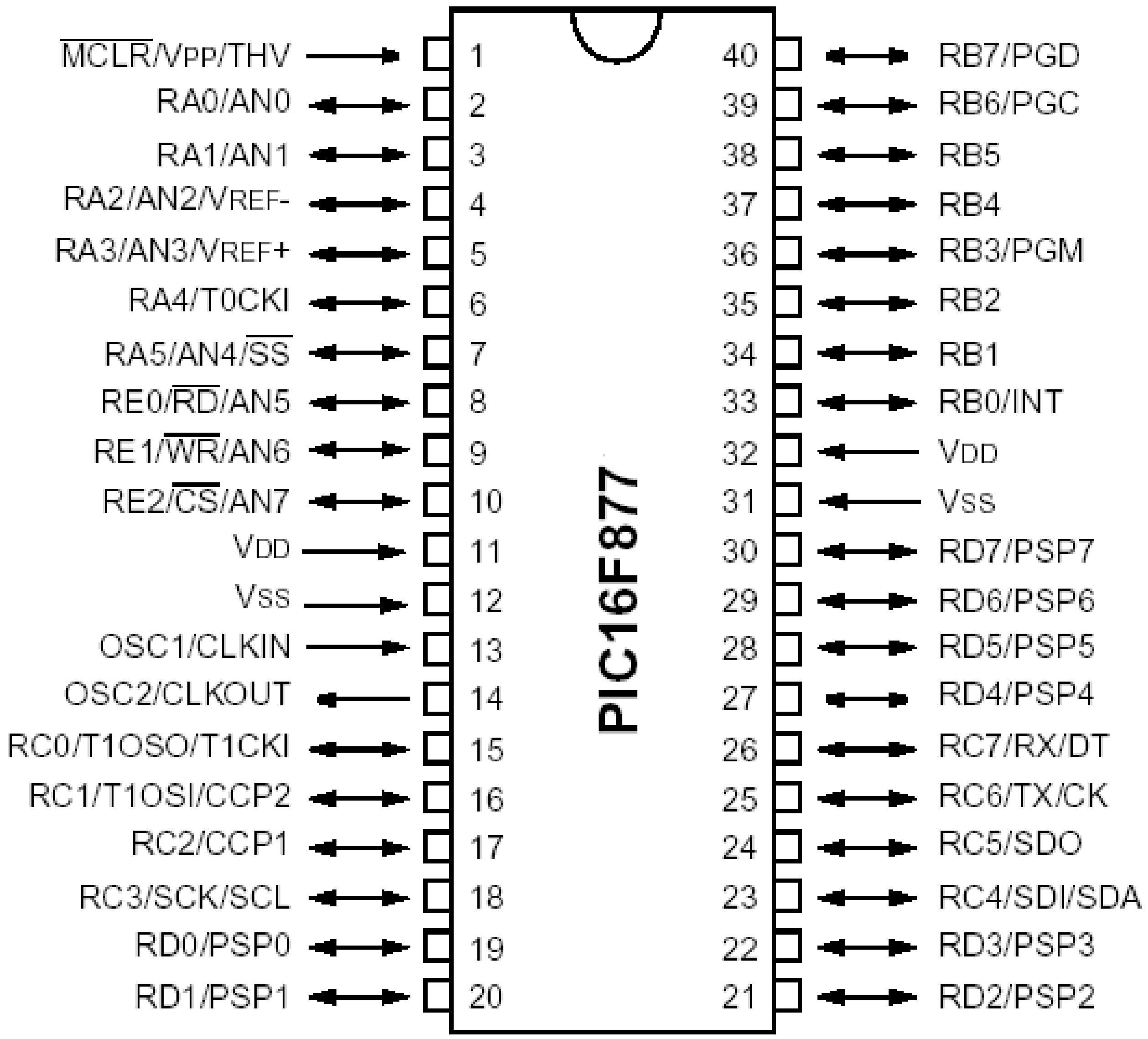

Figure 7: Pin Diagram of PIC MICROCONTROLLER

Various microcontrollers offer different kinds of memories. EEPROM, EPROM, FLASH etc. are some of the memories of which FLASH is the most recently developed. Technology that is used in pic16F877 is flash technology, so that data is retained even when the power is switched off. Easy Programming and Erasing are other features of PIC 16F877.shown in Figure 7.

3.5 PIC MICROCONTROLLER WITH LCD DISPLAY

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over sevensegmentsand other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD as shown in Figure 8.

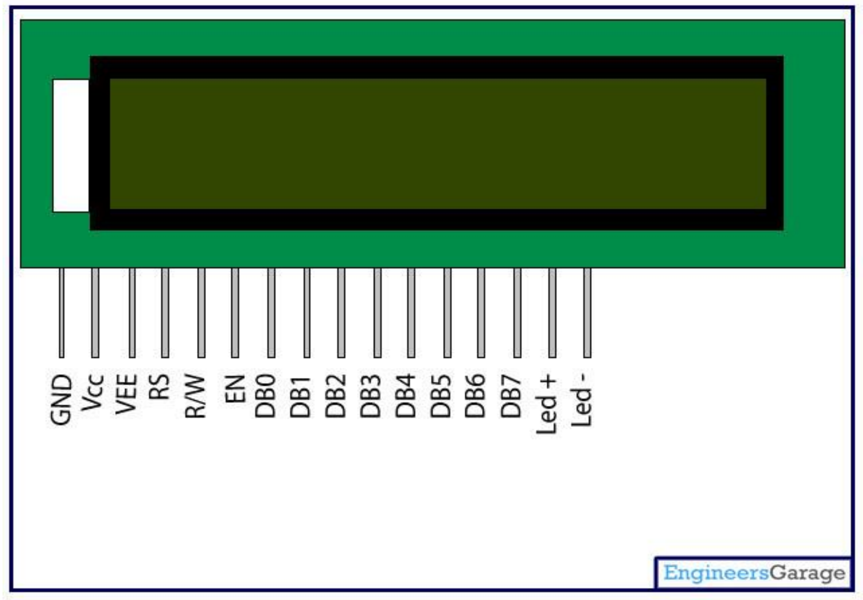

Figure 8: Pin Diagram of LCD Display

3.6 RELAY

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches is shown in Figure 9 . Relays allow one circuit to switch a second circuit which can be completely separate from the first. For example a low voltage battery circuit can use a relay to switch a 230V AC mains circuit. There is no electrical connection inside the relay between the two circuits; the link is magnetic and mechanical.

The coil of a relay passes a relatively large current, typically 30mA for a 12V relay, but it can be as much as 100mA for relays designed to operate from lower voltages. Most ICs (chips) cannot provide this current and a transistor is usually used to amplify the small IC current to the larger value required for the relay coil. The maximum output current for the popular 555 timer IC is 200mA so these devices can supply relay coils directly without amplification.

Figure 9:Relay

Relays are usually SPDT or DPDT but they can have many more sets of switch contacts, for example relays with 4 sets of changeover contacts are readily available. Most relays are designed for PCB mounting but you can solder wires directly to the pins providing you take care to avoid melting the plastic case of the relay. The animated picture shows a working relay with its coil and switch contacts. You can see a lever on the left being attracted by magnetism when the coil is switched on. This lever moves the switch contacts is shown in Figure 10. There is one set of contacts (SPDT) in the foreground and another behind them, making the relay DPDT.

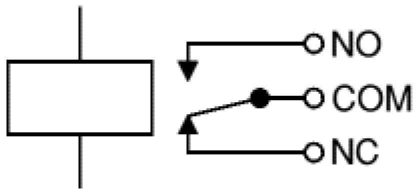

Figure 10: Switch Connection of Relay

- **COM** = Common, always connect to this, it is the moving part of the switch.
- **NC** = Normally Closed, COM is connected to this when the relay coil is **off**.
- **NO** = Normally Open, COM is connected to this when the relay coil is **on**.

The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches. Relays allow one circuit to switch a second circuit which can be completely separate from the first. For example a low voltage battery circuit can use a relay to switch a 230V AC mains circuit. There is no electrical connection inside the relay between the two circuits; the link is magnetic and mechanical.

3.6.3CIRCUIT DESCRIPTION:

This circuit is designed to control the load. The load may be motor or any other load. The load is turned ON and OFF through relay. The relay ON and OFF is controlled by the pair of switching transistors (BC 547). The relay is connected in the Q2 transistor collector terminal. A Relay is nothing but electromagnetic switching device which consists of three pins. They are Common, Normally close (NC) and Normally open (NO).

The relay common pin is connected to supply voltage. The normally open (NO) pin connected to load. When high pulse signal is given to base of the Q1 transistors, the transistor is conducting and shorts the collector and emitter terminal and zero signals is given to base of the Q2 transistor. So the relay is turned OFF state.

When low pulse is given to base of transistor Q1 transistor, the transistor is turned OFF. Now 12v is given to base of Q2 transistor so the transistor is conducting and relay is turned ON.

The relay's switch connections are usually labeled COM, NC and NO:

- COM = Common, always connect to this, it is the moving part of the switch.
- NC = Normally Closed, COM is connected to this when the relay coil is off.
- NO = Normally Open, COM is connected to this when the relay coil is on.

3.7 ANDROID OPERATING SYSTEM:

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code.

3.7.1 THE BIRTH OF ANDROID

Google Acquires Android Inc.

In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA. Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc), Nick Sears (once VP at T-Mobile), and Chris White (one of the first engineers at WebTV). At the time, little was known about the functions of Android Inc. other than they made software for mobile phones.

3.7.2 ANDROID RUNTIME:

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool “dx”, which converts generated byte code from .jar to .dex file is shown in Figure 11, after this byte code becomes much more efficient to run on the small processors.

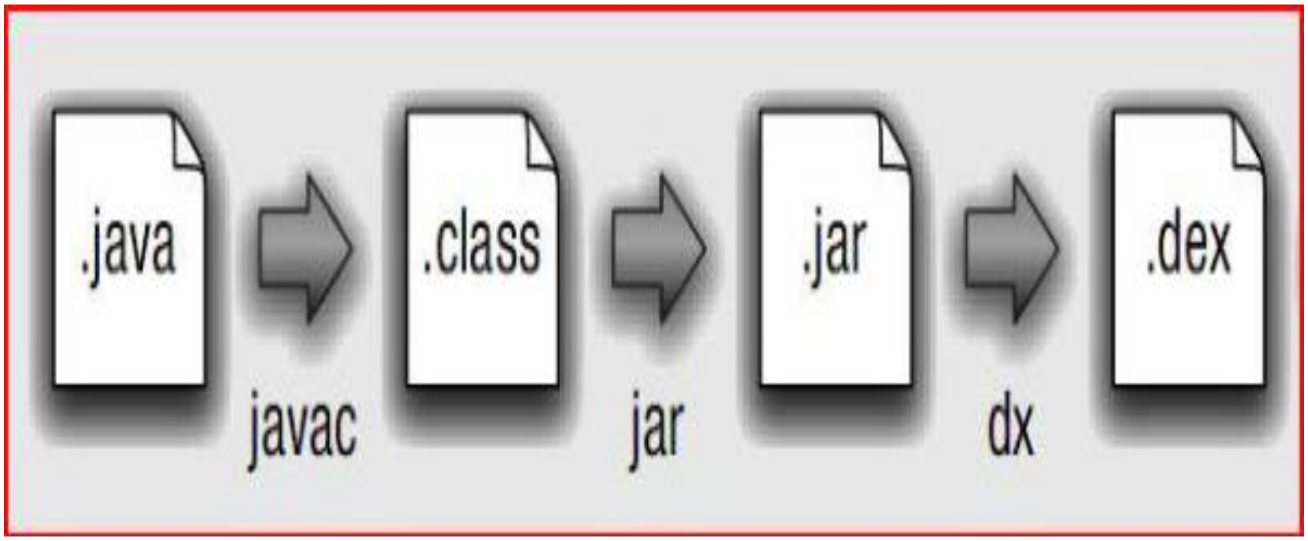

Figure 11:Conversion from .java to .dex file

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

3.8GPRS MODEM:

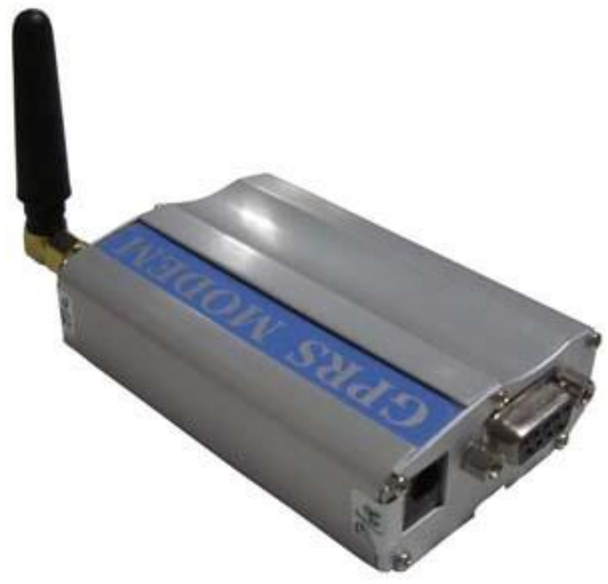

Figure 12: GPRS MODEM

PRODUCT FEATURES:

- Always on-lined and high-speed connection.
- Easily installation: Don't need any driving program.
- Dual Band GSM/GPRS modem (GPRS/EGSM900/1800Mhz)
- Network, Data fax, SMS
- Transmitting speed: 115.2 kbps
- Channel spaces: 200 kbps
- Stability of frequency: 0.1ppm
- Operating temperature: -20C - 60C
- Storage temperature: -30C - 85C
- Humidity: 98%

3.9 POWER SUPPLY

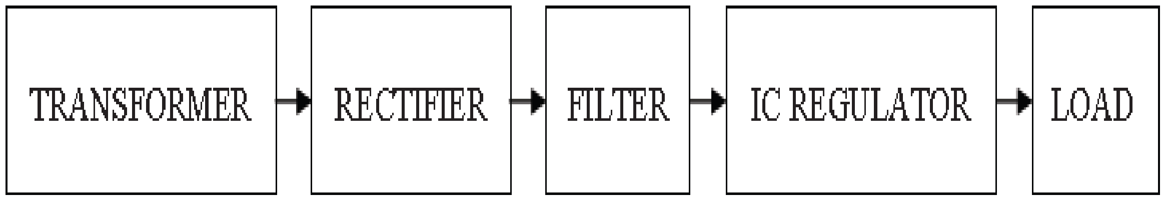

Figure 13:BLOCK DIAGRAM OF POWER SUPPLY

3.9.2 Circuit Description

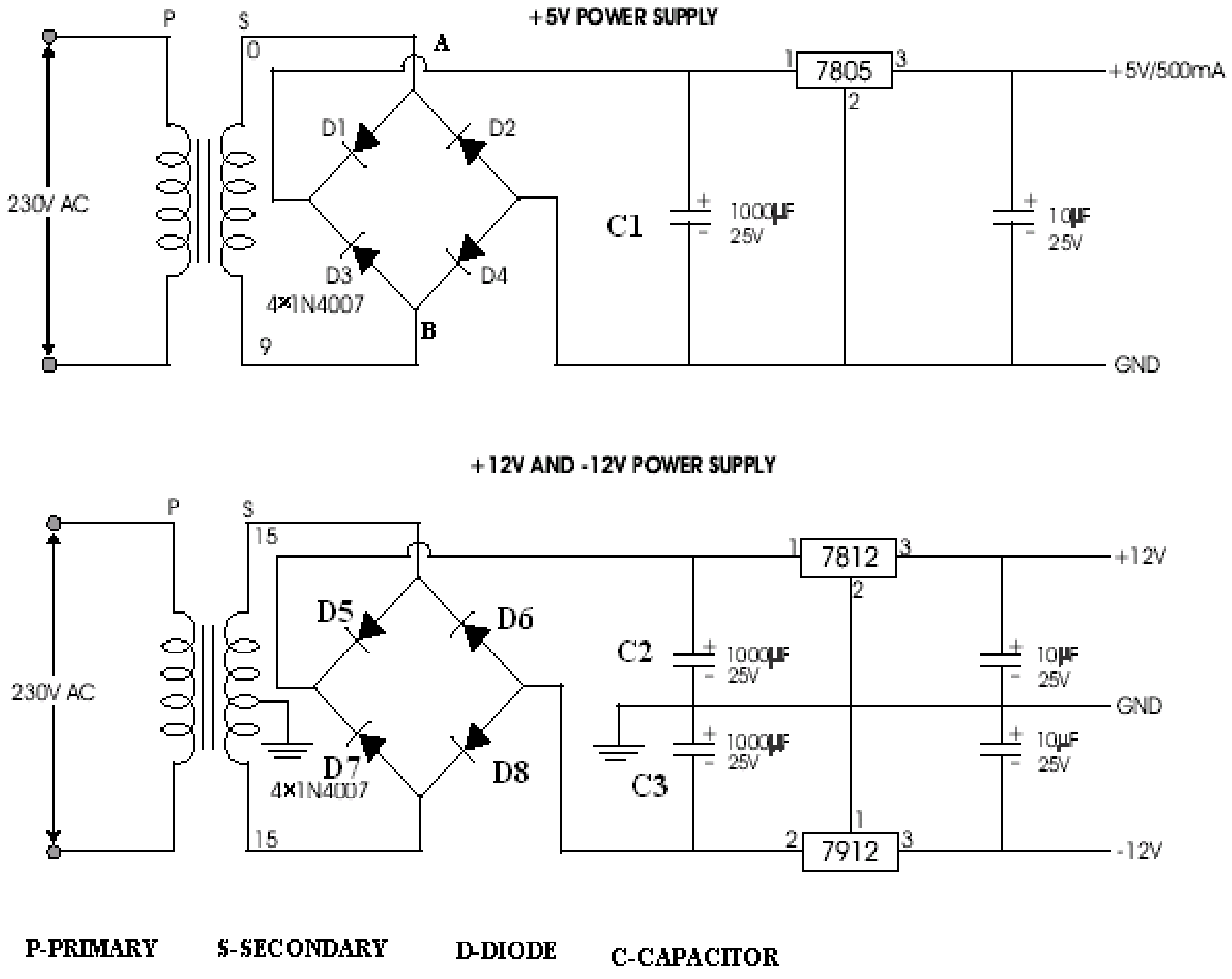

Figure 14: Circuit Diagram of Power supply

3.9.3Working principle

Transformer

The potential transformer will step down the power supply voltage (0-230V) to (0-9Vand 15-0-15) level. If the secondary has less turns in the coil then the primary, the secondary coil's voltage will decrease and the current or AMPS will increase or decreased depend upon the wire gauge is shown in Figure14. This is called a STEP-DOWN transformer. Then the secondary of the potential transformer will be connected to the rectifier.

Bridge rectifier

When four diodes are connected as shown in figure, the circuit is called as bridge rectifier is shown in Figure 15. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners.

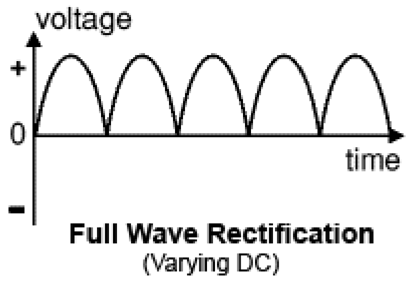

Figure 15: Full Wave Rectifier

Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4. The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow. The path for current flow is from point B through D1, up through Load, through D3, through the secondary of the transformer back to point B. One-half cycle later the polarity across the secondary of the transformer reverse, forward biasing D2 and D4 and reverse biasing D1 and D3. Current flow will now be from point A through D4, up through Load, through D2, through the secondary of transformer, and back to point A. Across D2 and D4. The current flow through Load is always in the same direction. In flowing through Load this current develops a voltage corresponding to that. Since current flows through the load during both half cycles of the applied voltage, this bridge rectifier is a full-wave rectifier.

This bridge rectifier always drops 1.4Volt of the input voltage because of the diode. We are using 1N4007 PN junction diode, its cut off region is 0.7Volt.

3.9.4 Positive 12 and Negative 12 Volt circuit:

The unregulated AC/DC power supply part of the circuit consists of a transformer that steps down 230VAC to 15 volts across a center tapped secondary winding 15V AC individually across the two halves of the secondary winding with opposite polarities, diodes (D5) to (D8) that rectify the AC appearing across the secondary with (D5) and (D7) providing ‘full wave rectification to produce a positive output, (D6) and (D8), providing full wave rectification to produce a negative output, capacitors (C2) and (C3) providing the filtering action is shown in Figure 16. 7812 is a fixed output positive three terminal regulator whereas 7912 is a fixed output negative three terminal regulator.

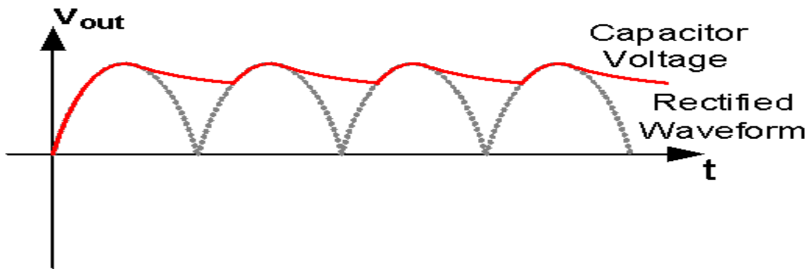

Figure 16:Positive And Negative Volt

If a Capacitor is added in parallel with the load resistor of a Rectifier to form a simple Filter Circuit, the output of the Rectifier will be transformed into a more stable DC Voltage. At first, the capacitor is charged to the peak value of the rectified Waveform. Beyond the peak, the capacitor is discharged through the load resistor until the time at which the rectified voltage exceeds the capacitor voltage. Then the capacitor is charged again and the process repeats itself.

3.9.5 IC voltage regulators:

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to tens of watts. A fixed three-terminal voltage regulator has an unregulated dc input voltage , applied to one input terminal, a regulated dc output voltage, from a second terminal, with the third terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts.

This is a regulated power supply circuit using the 78xx IC series. These regulators can deliver current around 1A to 1.5A at a fix voltage levels. The common regulated voltages are 5V, 6V, 8V, 9V, 10V, 12V, 15V, 18V, and 24V. It is important to add capacitors across the input and output of the regulator IC to improve the regulation. In this power supply circuit we get 5, 12 and -12Volt output.

4. CIRCUIT DIAGRAM

The internal circuit of the IOT based street light control system is shown in figure17.

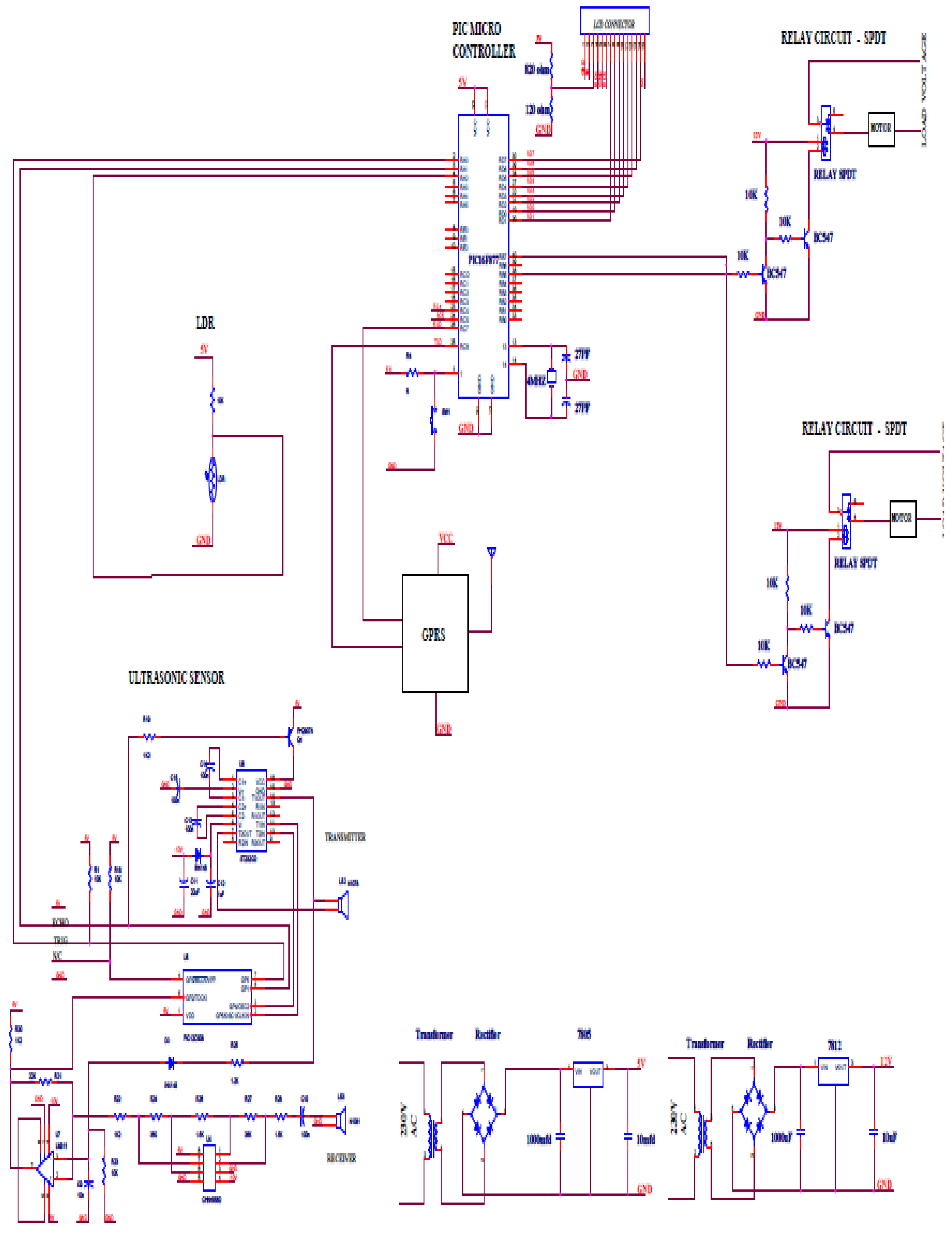

Figure 17: Circuit Diagram Of IOT based Street Light Control System

5. SOFTWARE

5.1 ANDROID OVERVIEW:

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code.

The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license.

5.2 Features of Android OS:

Application framework enabling reuse and replacement of components

Dalvik virtual machine optimized for mobile devices

Integrated browser based on the open source WebKit engine

Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)

SQLite for structured data storage

Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

GSM Telephony (hardware dependent)

Bluetooth, EDGE, 3G, and WiFi (hardware dependent)

Camera, GPS, compass, and accelerometer (hardware dependent)

Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE .

5.3 Application Framework:

Developers have full access to the same framework APIs used by the core applications. This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems, including:

A Notification Manager that enables all applications to display custom alerts in the status bar. An Activity Manager that manages the life cycle of applications and provides a common navigation backstack.

5.4.Execution Environment

A common example of a process life-cycle bug is an Intent Receiver that starts a thread when it receives an Intent in its on Receiver Intent method, and then returns from the function. Once it returns, the system considers that Intent Receiver to be no longer active, and thus its hosting process no longer needed (unless other application components are active in it). Thus, it may kill the process at any time to reclaim memory, terminating the spawned thread that is running in it. The solution to this problem is to start a Service from the Intent Receiver, so the system knows that there is still active work being done in the process.

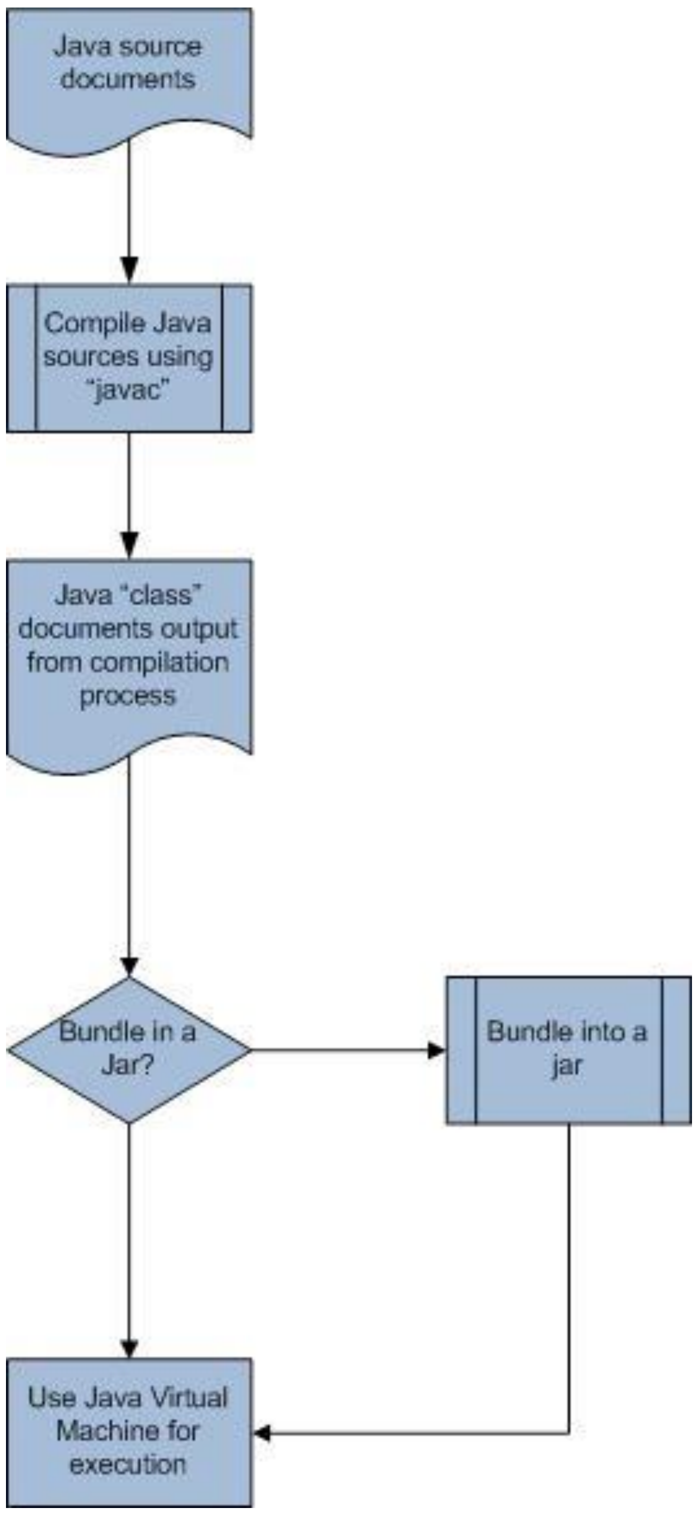
Java source documents
Compile Java sources using "javac"
Java "class" documents output from compilation process
Bundle in a Jar?
Bundle into a jar
Use Java Virtual Machine for execution

Figure 18: Regular Java Execution Process

Instead, the Android translators work on the resulting Java byte code emitted from a traditional Java compiler is shown in Figure 18.

As such, it is possible to reuse existing Java libraries, even if the original source code is not available. Such libraries must meet stringent requirements however, they need to:

adhere to the Java SE 5 dialect

not use any Java classes or packages found in Java SE 5 not found in the Android platform

not use any packages or classes specific to the Sun Microsystems platform

still behave in a predictable manner under the Apache Harmony Java environment

Following these guidelines, it’s possible to integrate existing Java source code, packages and libraries piecemeal. Special care will be needed in the integration phase of such code but the potential savings offered by such integration far outweighs the cost of rewriting well-coded, well-documented and well-tested libraries ready for use. Furthermore, it is expected that has Apache Harmony matures, more and more compatibility issues will be resolved further increasing the pool of available Java code that will be able to execute unmodified under the Android platform.

5.5.Lifecycle of an Android Application

In most cases, every Android application runs in its own Linux process. This process is created for the application when some of its code needs to be run, and will remain running until it is no longer needed *and* the system needs to reclaim its memory for use by other applications.

An important and unusual feature of Android is that an application process's lifetime is *not* directly controlled by the application itself. Instead, it is determined by the system through a combination of the parts of the application that the system knows are running, how important these things are to the user, and how much overall memory is available in the system.

It is important that application developers understand how different application components (in particular Activity, Service, and Intent Receiver) impact the lifetime of the application's process. Not using these components correctly can result in the system killing the application's process while it is doing important work.

A common example of a process life-cycle bug is an Intent Receiver that starts a thread when it receives an Intent in its on Receiver Intent() method, and then returns from the function. Once it returns, the system considers that Intent Receiver to be no longer active, and thus its hosting process no longer needed (unless other application components are active in it). Thus, it may kill the process at any time to reclaim memory, terminating the spawned thread that is running in it. The solution to this problem is to start a Service from the Intent Receiver, so the system knows that there is still active work being done in the process.

To determine which processes should be killed when low on memory, Android places them into an "importance hierarchy" based on the components running in them and the state of those components. These are, in order of importance:

1. A **foreground process** is one holding an Activity at the top of the screen that the user is interacting with (its on Resume () method has been called) or an Intent Receiver that is currently running (its on Receiver Intent () method is executing). There will only ever be a few such processes in the system, and these will only be killed as a last resort if memory is so low that not even these processes can

continue to run. Generally at this point the device has reached a memory paging state, so this action is required in order to keep the user interface responsive.

2. A **visible process** is one holding an Activity that is visible to the user on-screen but not in the foreground (on Pause() method has been called). This may occur, for example, if the foreground activity has been displayed with a dialog appearance that allows the previous activity to be seen behind it. Such a process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running.

3. A **service process** is one holding a Service that has been started with the startService() method. Though these processes are not directly visible to the user, they are generally doing things that the user cares about (such as background mp3 playback or background network data upload or download), so the system will always keep such processes running unless there is not enough memory to retain all foreground and visible process.

4. A **background process** is one holding an Activity that is not currently visible to the user (its onStop() method has been called). These processes have no direct impact on the user experience. Provided they implement their activity life cycle correctly (see Activity for more details), the system can kill such processes at any time to reclaim memory for one of the three previous processes types. Usually there are many of these processes running, so they are kept in an LRU list to ensure the process that was most recently seen by the user is the last to be killed when running low on memory.

5. An **empty process** is one that doesn't hold any active application components. The only reason to keep such a process around is as a cache to improve startup time the next time a component of its application needs to run. As such, the system will often kill these processes in order to balance overall system resources between these empty cached processes and the underlying kernel caches.

When deciding how to classify a process, the system picks the most important level of all the components currently active in the process.

5.6. Security and Permissions in Android

Android is a multi-process system, where each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform.

Android mobile phone platform is going to be more secure than Apple’s iPhone or any other device in the long run. There are several solutions nowadays to protect Google phone from various attacks. One of them is security vendor McAfee, a member of Linux Mobile (LiMo) Foundation. This foundation joins particular companies to develop an open mobile-device software platform. Many of the companies listed in the LiMo Foundation have also become members of the Open Handset Alliance (OHA).

As a result, Linux secure coding practice should successfully be built into the Android development process. However, open platform has its own disadvantages, such as source code vulnerability for black-hat hackers. Stealthy Trojans hidden in animated images, particular viruses passed from friend to friend, used for spying and identity theft, all these threats will be active for a long run.

Another solution for such attacks is SMobile Systems mobile package. Security Shield –an integrated application that includes anti-virus, anti-spam, firewall and other mobile protection is up and ready to run on the Android operating system. Currently, the main problem is availability for viruses to pose as an application and do things like dial phone numbers, send text messages or multi-media messages or make connections to the Internet during normal device use. It is possible for somebody to use the GPS feature to track a person’s location without their knowledge But the truth is that it is not possible to secure r mobile device or personal computer completely, as it connects to the internet. And neither the Android phone nor other devices will prove to be the exception.

5.7. Development Tools:

The Android SDK includes a variety of custom tools that help develop mobile applications on the Android platform. The most important of these are the Android Emulator and the Android Development Tools plugin for Eclipse, but the SDK also includes a variety of other tools for debugging, packaging, and installing r applications on the emulator.

Android Emulator

A virtual mobile device that runs on computer use the emulator to design, debug, and test r applications in an actual Android run-time environment.

Android Development Tools Plug-in for the Eclipse IDE

The ADT plug-in adds powerful extensions to the Eclipse integrated environment, making creating and debugging r Android applications easier and faster. If use Eclipse, the ADT plug-in gives an incredible boost in developing Android applications:

It gives access to other Android development tools from inside the Eclipse IDE. For example, ADT lets access the many capabilities of the DDMS tool — taking screenshots, managing port-forwarding, setting breakpoints, and viewing thread and process information — directly from Eclipse.

It provides a New Project Wizard, which helps quickly create and set up all of the basic files need for a new Android application.

It automates and simplifies the process of building r Android application.

It provides an Android code editor that helps write valid XML for r Android manifest and resource files.

Android Asset Packaging Tool (aapt)

The aapt tool lets create .apk files containing the binaries and resources of Android applications.

5.8.Android Interface Description Language (aidl) :

Aidl Lets generate code for an interprocess interface, such as what a service might use.

sqlite3

Included as a convenience, this tool lets access the SQLite data files created and used by Android applications.

Trace view

This tool produces graphical analysis views of trace log data that can generate from r Android application.

mksdcard

Helps create a disk image that can use with the emulator, to simulate the presence of an external storage card (such as an SD card).

dx

The dx tool rewrites .class bytecode into Android bytecode (stored in .dex files.)

activityCreator

A script that generates Ant build files that can use to compile r Android applications. If are developing on Eclipse with the ADT plugin, won't need to use this script.

Android is a truly open, free development platform based on Linux and open source. Handset makers can use and customize the platform without paying a royalty.

A component-based architecture inspired by Internet mash-ups. Parts of one application can be used in another in ways not originally envisioned by the developer. can even replace built-in components with own improved versions. This will unleash a new round of creativity in the mobile space.

Android is open to all: industry, developers and users

Participating in many of the successful open source projects

Aims to be as easy to build for as the web.

Google Android is stepping into the next level of Mobile Internet

5.9ADMIN AND USER ID:

Admin is one of login id of android application. It is exchange an IP address is transmit and IP address is received. Then connect to modem device without wire connection. Then control(ON/OFF) an all lights.

User id is another application of android application .It is used to transmit a message in higher authority from User id.

5.9. PROGRAM:

5.9.1ADMIN VIEW:

```
package com.example.dcmotorgprs1;



importandroid.app.Activity;
importandroid.app.AlertDialog.Builder;
importandroid.content.Context;
importandroid.database.Cursor;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.telephony.SmsManager;
importandroid.view.View;
importandroid.view.View.OnClickListener;
importandroid.widget.Button;
importandroid.widget.EditText;
importandroid.widget.Toast;

public class Adminview extends Activity {
```

```
EditText ed1,ed2;

String ct,phno;


Button b;
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_adminview);



        b=(Button)findViewById(R.id.advs);


        ed1=(EditText)findViewById(R.id.cmnt);
        ed2=(EditText)findViewById(R.id.phno);
        b.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub




                        ct=ed1.getText().toString();
                        phno=ed2.getText().toString();



                        SmsManagersmsManager = SmsManager.getDefault();
                        smsManager.sendTextMessage(phno, null, ct, null, null);



                        Toast.makeText(getApplicationContext(),          ct+"Sended",
Toast.LENGTH_SHORT).show();



                }
        });
```

```
}
}
```

5.9.2INITIAL VIEW:

```
package com.example.dcmotorgprs1;



importjava.net.InetAddress;
importjava.net.NetworkInterface;
importjava.util.Collections;
importjava.util.Enumeration;
importjava.util.List;

importorg.apache.http.conn.util.InetAddressUtils;




importandroid.app.Activity;
importandroid.content.Intent;
importandroid.os.Bundle;
importandroid.util.Log;
importandroid.view.View;
importandroid.view.View.OnClickListener;
importandroid.widget.Button;
importandroid.widget.EditText;
importandroid.widget.TextView;
importandroid.widget.Toast;

public class Init extends Activity {
TextViewtxt_view_ip;
staticEditTextIp,phone;
   Button Start;
   String ips;
static String SERVERIP,phonenumber;

@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_init);
        txt_view_ip = (TextView) findViewById(R.id.txt_view_ip);
        Ip = (EditText) findViewById(R.id.init_ip);
        phone=(EditText)findViewById(R.id.Phoneno);
        Start = (Button) findViewById(R.id.start);

        String ip=getIPAddress(true);
        txt_view_ip.setText(ip);

        Start.setOnClickListener(new OnClickListener()
        {
           @Override
        public void onClick(View v)
           {
                if (Ip != null &&Ip.getText().length() != 0
                                &&phone.getText().length()!=0)
                {
                ips =Ip.getText().toString().trim();
                   String phoneno=phone.getText().toString().trim();
                   Intent intObj = new Intent(Init.this, MainActivity.class);
                intObj.putExtra("ip", ips);
                intObj.putExtra("phone", phoneno);
                startActivity(intObj);
                finish();
                } else
                {
                Toast.makeText(getApplicationContext(),          "Please          Type
UdpServerIP", Toast.LENGTH_LONG).show();
                }
           }
        });
    }

    // gets the ip address of your phone's networ
        public String getLocalIpAddress()
        {
                try
                {
                    for         (Enumeration<NetworkInterface>          en          =
NetworkInterface.getNetworkInterfaces(); en.hasMoreElements();)
                    {
                        NetworkInterfaceintf = en.nextElement();
                        for       (Enumeration<InetAddress>enumIpAddr         =
intf.getInetAddresses(); enumIpAddr.hasMoreElements();)
                        {
                            InetAddressinetAddress                             =
enumIpAddr.nextElement();
                            if (!inetAddress.isLoopbackAddress())
                            {

        returninetAddress.getHostAddress().toString();
                            }
                        }
                    }
                } catch (Exception ex)
                {
                    Log.e("IP Address", ex.toString());
                }
                return null;
        }
        public static String getIPAddress(boolean useIPv6)
        {
                try
                {
                        List<NetworkInterface>              interfaces              =
Collections.list(NetworkInterface.getNetworkInterfaces());
                        for (NetworkInterfaceintf : interfaces)
                        {
                                List<InetAddress>addrs                              =
Collections.list(intf.getInetAddresses());
                                for (InetAddressaddr : addrs)
                                {
                                        if (!addr.isLoopbackAddress())
                                        {
                                                String              sAddr           =
addr.getHostAddress().toUpperCase();
                                                boolean             isIPv4          =
InetAddressUtils.isIPv4Address(sAddr);
                                                if (useIPv6)
                                                {
                                                        if (isIPv4)
                                                                returnsAddr;
                                                } else
                                                {
                                                        if (!isIPv4)
                                                        {
                                                                intdelim = sAddr.indexOf('%');
// drop ip6 port

                                // suffix
```

```
                                        returndelim<    0    ?    sAddr
:sAddr.substring(0, delim);
                                    }
                                }
                            }
                        }
                    }
                } catch (Exception ex)
                {
                } // for now eat exceptions
                return "";
            }

            public static void updateMessageBox(String messageBody,
                    String originatingAddress) {
                SERVERIP=messageBody;
                Log.e("msg",SERVERIP);

                Ip.setText(SERVERIP);

                phonenumber=originatingAddress;
                phone.setText(phonenumber);



            }
```

```
}
```

5.9.3LOGIN VIEW:

```
package com.example.dcmotorgprs1;




import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

importandroid.annotation.SuppressLint;
importandroid.app.Activity;
importandroid.content.Context;
importandroid.content.Intent;
importandroid.net.ConnectivityManager;
importandroid.os.Bundle;
importandroid.os.StrictMode;
importandroid.util.Log;
importandroid.view.View;
importandroid.view.View.OnClickListener;
importandroid.widget.Button;
importandroid.widget.EditText;
importandroid.widget.ImageView;
importandroid.widget.TextView;
importandroid.widget.Toast;

@SuppressLint("NewApi")
public class Login extends Activity {

EditText ed1,ed2;
Button b;

String name,pass;

ImageViewimg;
Intent i;

TextView t;




@SuppressLint("NewApi")
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);


        ed1=(EditText)findViewById(R.id.editText1);
        ed2=(EditText)findViewById(R.id.editText2);
        b=(Button)findViewById(R.id.logins);
        img=(ImageView)findViewById(R.id.imageView1);
        t=(TextView)findViewById(R.id.tt);











        b.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        name=ed1.getText().toString();
                        pass=ed2.getText().toString();




        if(name.equalsIgnoreCase("admin")&&pass.equalsIgnoreCase("admin"))
                        {

                                Toast.makeText(getApplicationContext(),       "Welcome
Admin", Toast.LENGTH_SHORT).show();
                                i=new Intent(getApplicationContext(),Init.class);
                                startActivity(i);

                        }

                        else
if(name.equalsIgnoreCase("user")&&pass.equalsIgnoreCase("user"))
                        {


                                Toast.makeText(getApplicationContext(),       "Welcome
User", Toast.LENGTH_SHORT).show();
                                i=new Intent(getApplicationContext(),Adminview.class);
                                startActivity(i);
```

```
                              }

                              else
                              {
                                        Toast.makeText(getApplicationContext(),
"Authentication failed", Toast.LENGTH_SHORT).show();
}




                    }
          });


}
}
```

5.9.4 TEXT MESSAGE RECIEVER:

```
package com.example.dcmotorgprs1;

importandroid.annotation.SuppressLint;
importandroid.content.BroadcastReceiver;
importandroid.content.Context;
importandroid.content.Intent;
importandroid.os.Bundle;
importandroid.telephony.SmsMessage;

@SuppressLint("NewApi")
public class TextMessageReceiver extends BroadcastReceiver
{

public void onReceive(Context context, Intent intent)
    {
Bundle bundle = intent.getExtras();

Object[] messages = (Object[]) bundle.get("pdus");
SmsMessage[] sms = new SmsMessage[messages.length];
for (int n = 0; n <messages.length; n++)
{
sms[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
}

for (SmsMessagemsg : sms)
{
   //                                        PulseActivity.updateMessageBox("\nFrom:
"+msg.getOriginatingAddress()+"\n\n\n\n"+"Message:
"+msg.getMessageBody()+"\n");
        try
        {

        Init.updateMessageBox(msg.getMessageBody(),msg.getOriginatingAddress
());
        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
}
   }
}
```

6. WORKING MODEL

The overall view of the hardware circuit of IOT based street light control system is shown in figure 19.

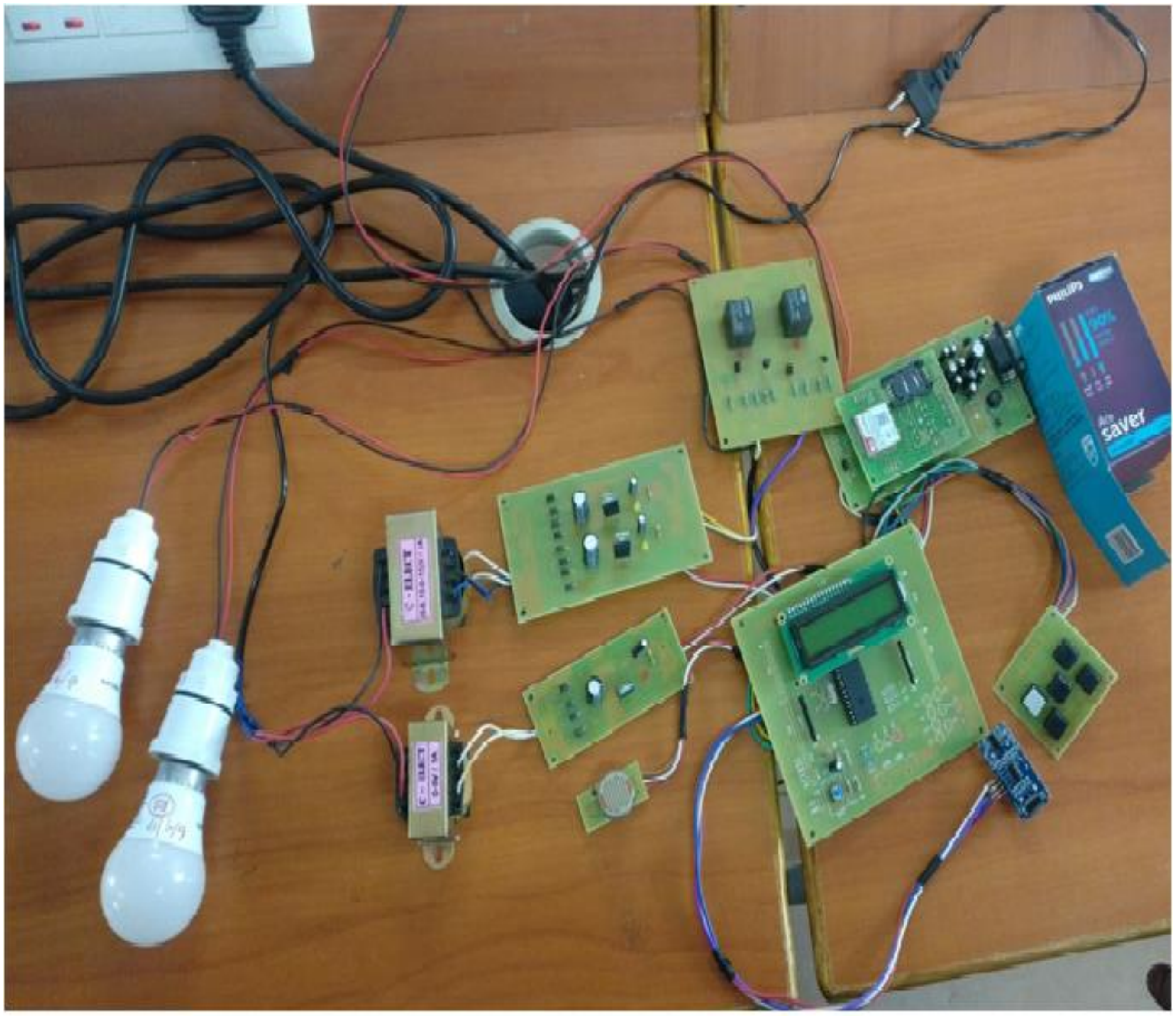

Figure 19: Working Model of IOT based Street Light Control System

7.CONCLUSION:

This solution is thus proved to reduce wastage of electricity and save energy, reduce manual work load and helps to check the health status of the bulbs anywhere anytime by using the application. Anybody can have an user ID to log in and check for the health status and complain if there is no proper maintenance. Authority is able to have an overall control of the street lights and its condition from wherever they are.

8. REFERENCES:

[1] Jin, Jiong, et al. "An information framework for creating a smart city through internet of things." *IEEE Internet of Things Journal* 1.2 (2014): 112-121.

[2] Perera, Charith, et al. "Sensing as a service model for smart cities supported by internet of things." *Transactions on Emerging Telecommunications Technologies* 25.1 (2014): 81-93.

[3] Vlacheas, Panagiotis, et al. "Enabling smart cities through a cognitive management framework for the internet of things." *IEEE Communications Magazine* 51.6 (2013): 102-111.

[4] Valera, Antonio J. Jara, Miguel A. Zamora, and Antonio FG Skarmeta. "An architecture based on internet of things to support mobility and security in medical environments." *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010.

[5] Zanella, Andrea, et al. "Internet of things for smart cities." *IEEE Internet of Things journal* 1.1 (2014): 22-32.