

CAPSTONE PROJECT REPORT

(Project Term July-December, 2018)

TO MANAGE THE POWER USING GSM MODULE

Submitted by

Deepak	11507188
Krishna Gulati	11505658
Allu Keshav	11507582

Project Group Number: KC200

Course Code: CSE339

Under the Guidance of
(Ms. Devinder Kaur)

School of Computer Science and Engineering



DECLARATION

We hereby declare that the project work entitled “To Manage The Power Using GSM Module “ is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Ms. Devinder Kaur, during January to April 2018. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: K200

Name of Student 1: Deepak

Registration Number: 11507188

Name of Student 2: Krishan Gulati

Registration Number: 11505658

Name of Student 3: Allu Keshav

Registration Number: 11507582

(Signature of Student 1)

Date:

(Signature of Student 2)

Date:

(Signature of Student 3)

Date:

ACKNOWLEDGEMENT

We are sincerely thankful to all those people who have been giving us assistance in the making of the project. We would like to acknowledge with thanks the kind of patronage, loving inspiration and timely guidance, given by our course mentor Ms. Devinder Kaur. And we would also like to express our sincere thanks to all the faculties whose teachings gave us conceptual understanding and clarity of comprehension, which ultimately made the work of the project easier. Last but not the least we pay a deep sense of gratitude to our family members and friends who have been constant source of inspiration and support and their valuable advice for the preparation of this project.

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Mentor's Signature

Name: Devinder Kaur

Designation: Assistant Professor

School of Computer and Science Engineering,

Lovely Professional University,

Phagwara, Punjab.

Date:

TABLE OF CONTENTS

1. INTRODUCTION	1
2. PROFILE OF THE PROBLEM.	2
3. EXISTING SYSTEM	2
3.1 INTRODUCTION	3
3.2 EXISTING SOFTWARE	3
3.3 BLOCK DIAGRAM.....	6
3.4 WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED	7
4. HARDWARE REQUIREMENT.....	5
5. SOFTWARE REQUIREMENT ANALYSIS	22
5.1 INTRODUCTION	22
5.2 GENERAL DESCRIPTION	25
5.3 SPECIFIC REQUIREMENTS	27
6.HARDWARE TESTING.....	27
7. IMPLEMENTATION.....	28
7.1 IMPLEMENTATION OF THE PROJECT.....	28
8.USER MANUAL.....	29
9. SOURCE CODE.....	30
10. BIBLIOGRAPHY.....	62

1.INTRODUCTION

It is an advanced metering technology involving intelligent meters to read, process and analyse the data to customers. It measures energy consumption, remotely by switches the supply to customers and remotely monitors and controls the maximum electricity consumption. Smart metering system uses the advanced metering infrastructure technology for better performance.

These are capable of communicating in both directions. They can transmit the data to the utilities like energy consumption, parameter values, etc and also can receive information from utilities such as automatic meter reading system, reconnect/disconnect instructions and other important messages. These meters reduce the need to visit while taking or reading monthly bill. Modems are used in these smart meters to facilitate communication systems such as telephone, wireless, power line communications.

2. PROFILE OF THE PROBLEM

The demand for power has magnified exponentially over the last century.

One avenue through which today's energy issues are often addressed is through the reduction of energy usage in households.

This has increased the emphasis on the necessity for correct and economic ways of power measurements.

In the present billing system the distribution companies are unable to keep track of the changing maximum demand of consumers. The consumer is facing problems like receiving due bills for bills that have already been paid as well as poor reliability of electricity supply and quality even if bills are paid regularly. The remedy for all these problems is to keep track of the customers load on timely basis, which will help to assure accurate billing, track maximum demand and to detect threshold value. These are all the features to be taken into account for coming up with economical energy system.

3. EXISTING SYSTEM

3.1 INTRODUCTION

In the present system we are seeing that the person is come from the electricity board and check our electric meter and give the bills to the consumer. This is the just the meter reading. According to the meter we have to pay the bills. The main drawback is that the person have to move from one place to another and to read the meter reading of every house. Many times extra bill amount type of errors are coming. To overcome from this system we have come with an idea of smart meter which also eliminate the third party from consumer and the provider.

3.2 EXISTING SOFTWARE

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc. Lower end embedded systems - Generally 8,16 Bit Controllers used with an minimal

operating systems and hardware layout designed for the specific purpose. Examples Small controllers and devices in our everyday life like Washing Machine, Microwave Ovens, where they are embedded in.

EMBEDDED SYSTEM DESIGN CYCLE

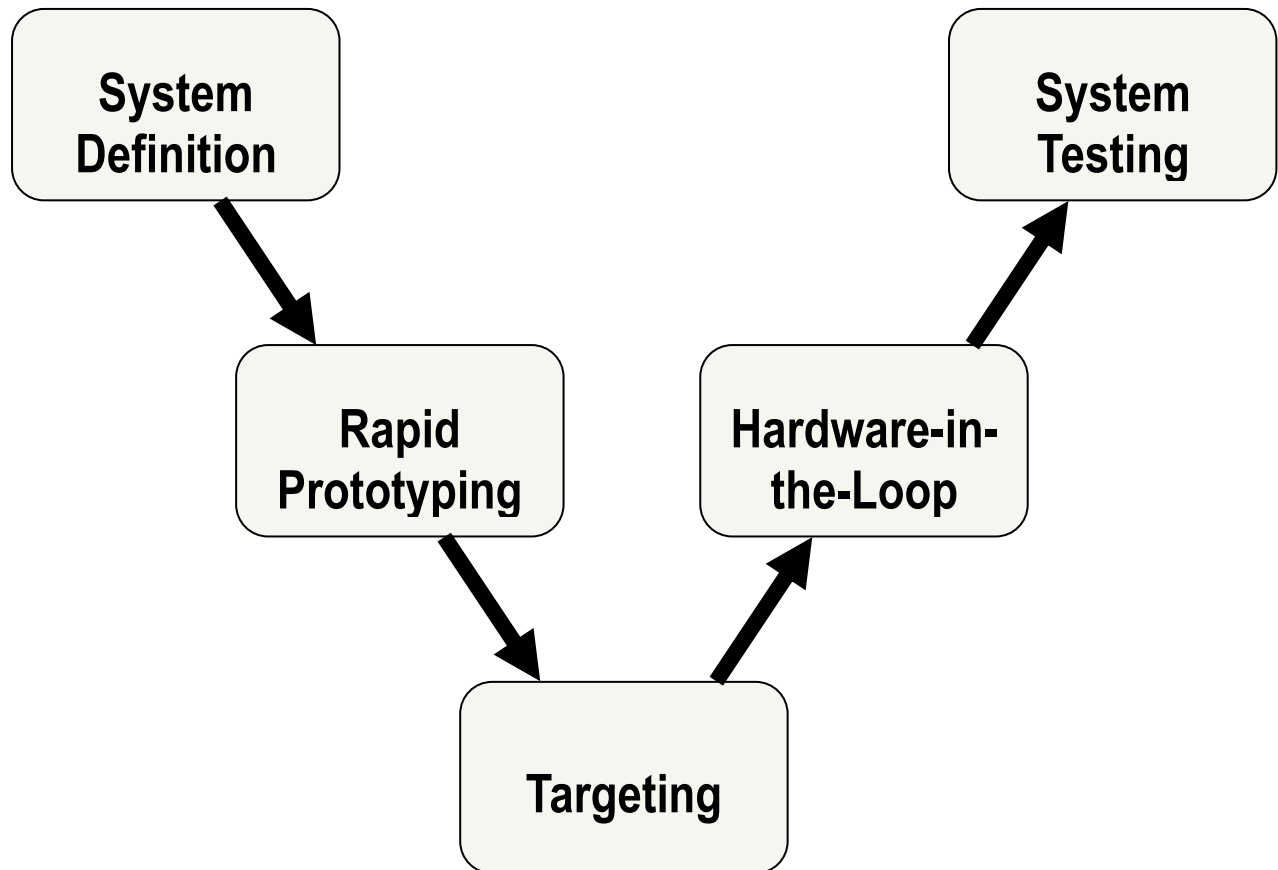


Figure 3.1 “V Diagram”

Characteristics of Embedded System

- An embedded system is any computer system hidden inside a product other than a computer.
- They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications.
 - Throughput – Our system may need to handle a lot of data in a short period of time.
 - Response–Our system may need to react to events quickly
 - Testability–Setting up equipment to test embedded software can be difficult
 - Debugability–Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem

- Reliability – embedded systems must be able to handle any situation without human intervention
 - Memory space – Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists
 - Program installation – you will need special tools to get your software into embedded systems
 - Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power
 - Processor hogs – computing that requires large amounts of CPU time can complicate the response problem
 - Cost – Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.
- Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.

APPLICATIONS

- 1) Military and aerospace embedded software applications
- 2) Communication Applications
- 3) Industrial automation and process control software
- 4) Mastering the complexity of applications.
- 5) Reduction of product design time.
- 6) Real time processing of ever increasing amounts of data.
- 7) Intelligent, autonomous sensors.

CLASSIFICATION

- Real Time Systems.
- RTS is one which has to respond to events within a specified deadline.
- A right answer after the dead line is a wrong answer

RTS CLASSIFICATION

- Hard Real Time Systems
- Soft Real Time System

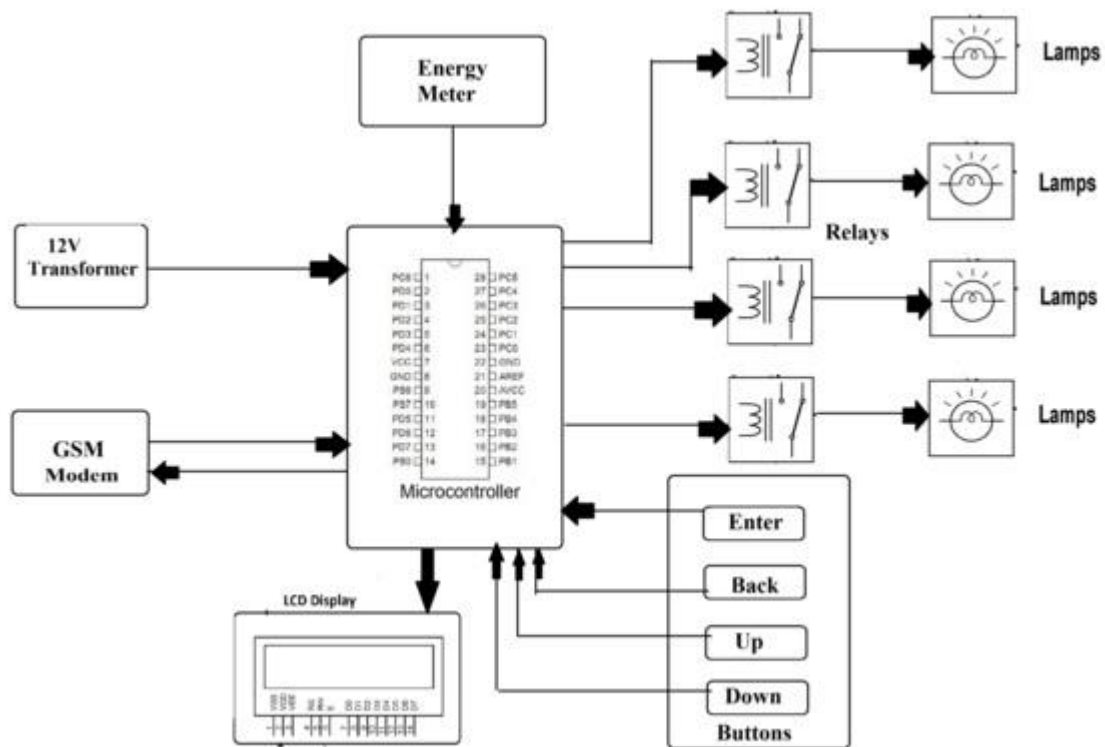
HARD REAL TIME SYSTEM

- "Hard" real-time systems have very narrow response time.
- Example: Nuclear power system, Cardiac pacemaker.

SOFT REAL TIME SYSTEM

- "Soft" real-time systems have reduced constraints on "lateness" but still must operate very quickly and repeatably.
- Example: Railway reservation system – takes a few extra seconds the data remains valid.

3.3 BLOCK DIAGRAM



3.4 What's new in the system to be developed

The main aim of the Programmable energy meter project is to estimate energy consumed by load with the help of an Atmega 328 and calculate the cost. Energy meter gives pulses which will be counted and displayed on LCD. These pulses occur depending on energy consumed. GSM to send SMS on amount of energy consumed. 12V Transformer to give power supply to the system. The user needs to make call which will be received and displayed on LCD and stored. The system then comes to setup mode from where the user can switch to start or setting option, setting to set unit cost. Start to know the unit consumed and the cost. The system switches the loads ON/OFF Depending on user commands received through SMS. The system also includes a feature in which user can specify the number of days for which the user wants to retrieve the consumed and the estimated cost for the specified days.

4.HARDWARE REQUIREMENTS

1. VOLTAGE REGULATOR (LM 7805)
2. RECTIFIER
3. FILTER
4. MICROCONTROLLER (ATmega328)
5. GSM COMMUNICATION
6. GSM MODEM
7. ULN2003
8. RELAY
9. LED
10. LCD DISPLAY
11. MAX 232
12. DB9 CONNECTOR
13. ENERGY METER
14. OPTO COUPLER
15. 1N4007
16. RESISTOR
17. CAPACITOR

4.1 VOLTAGE REGULATOR 7805

Features

- Output Current up to 1A.
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V.
- Thermal Overload Protection.
- Short Circuit Protection.
- Output Transistor Safe Operating Area Protection.

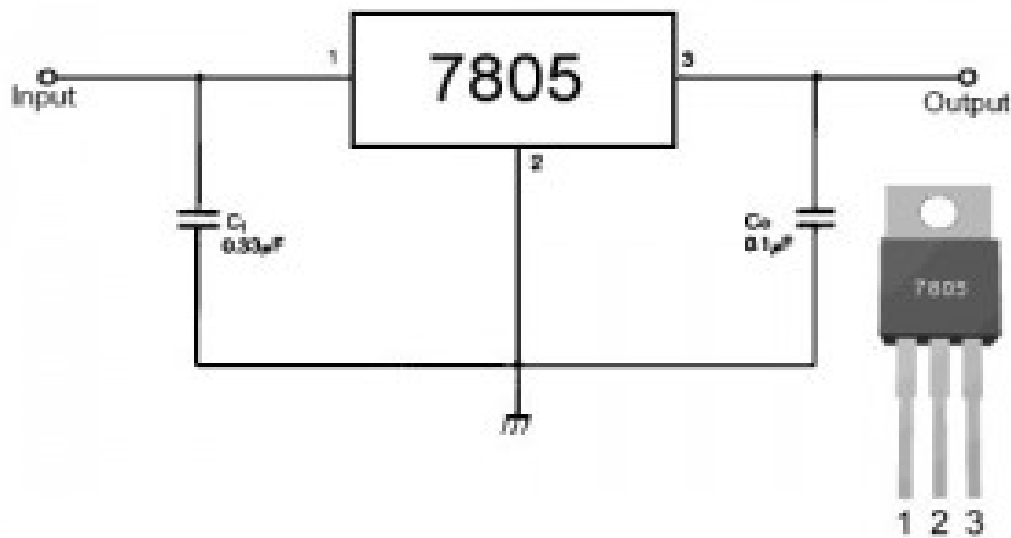


FIG:4.1 VOLTAGE REGULATOR(LM7805)

Description

The LM78XX/LM78XXA series of three-terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shutdown and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output Current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

4.2 RECTIFIER

A rectifier is an electrical device that converts alternating current (AC), which periodically reverses direction, to direct current (DC), current that flows in only one direction, a process

known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid state diodes, vacuum tube diodes, mercury arc valves, and other components. The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification. In positive half cycle only two diodes (1 set of parallel diodes) will conduct, in negative half cycle remaining two diodes will conduct and they will conduct only in forward bias only.

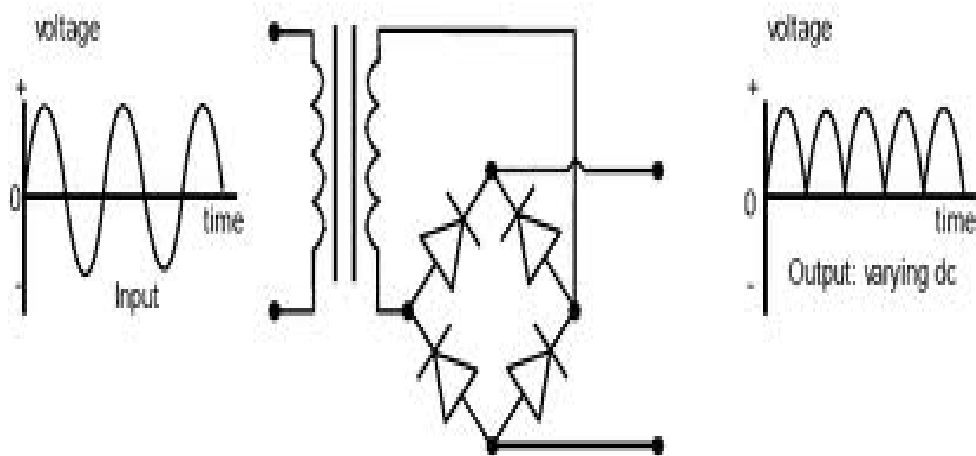


FIG:4.2 RECTIFIER

4.3 FILTER

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.

The simple capacitor filter is the most basic type of power supply filter. The use of this filter is very limited. It is sometimes used on extremely high-voltage, low-current power supplies for cathode-ray and similar electron tubes that require very little load current from the supply. This filter is also used in circuits where the power-supply ripple frequency is not critical and can be relatively high. Below figure can show how the capacitor charges and discharges.

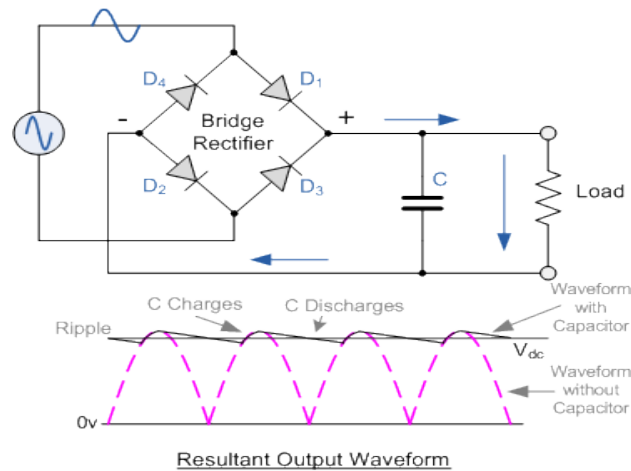


FIG:4.3 FILTER

4.4 ATMEGA 328

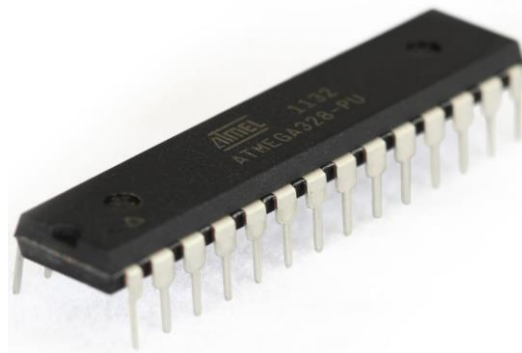


FIG:4.4ATMEGA 328

ATmega328 is an eight bit Microcontroller. It can handle the data sized of up to eight bits. It is an AVR based miniaturized scale controller. Its built in interior memory is around 32KB. It works extending from 3.3V to 5V. It has a capacity to store the information notwithstanding when the electrical supply is expelled from its biasing terminals. Its brilliant highlights incorporate the cost effectiveness, low power dispersal, programming lock for security purposes, genuine clock counter with isolated oscillator. It's ordinarily utilized as a part of Embedded Systems applications.

The computer on one hand is intended to play out all the universally useful errands on a solitary machine like you can utilize a computer to run a product to perform estimations or you can utilize a computer to store some media record or to get to web through the program, while the microcontrollers are intended to perform just the particular undertakings, for e.g., turning the AC off naturally when room temperature drops to a specific characterized confine and again turning it ON when temperature transcends as far as possible.

There are number of mainstream groups of microcontrollers which are utilized as a part of various applications according to their capacity and attainability to play out the coveted errand, most normal of these are 8051, AVR and PIC microcontrollers. In this we will present you with AVR group of microcontroller.

There are number of well known groups of microcontrollers which are utilized as a part of various applications according to their capacity and achievability to play out the coveted undertaking, most normal of these are 8051, AVR and PIC microcontrollers. In this we will present you with AVR group of microcontrollers.

4.5 GSM COMMUNICATION

GSM for mobile system is increasingly popular and established throughout the world. The term GSM usually means the GSM standard and protocols in the frequency spectrum around 900MHz. There is also DCS1800 - GSM protocols but at different air frequencies around 1800 MHz - and in the United States, where spectrum for Personal Communication Services (PCS) was auctioned at around 1900MHz. As a result of this, the original and most widely-used GSM frequency implementation is known as GSM900, and DCS1800 is also known as GSM1800. Though the physical frequencies used are differed, the protocols and architecture remain the same. The following sections describe about the functional entities, the radio interface signaling protocol, the logical and physical channel structure and the TDMA structure based on GSM.

System architecture

The figure below shows the GSM system architecture, which consists of the switching system, the base station system and the user equipment. Functional entities are briefly explained as follows.

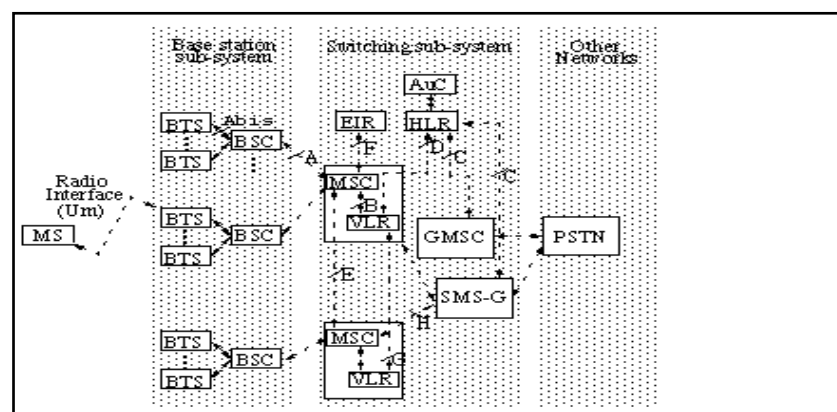


FIG:4.5(a) DETAILED ARCHITECTURE OF GSM

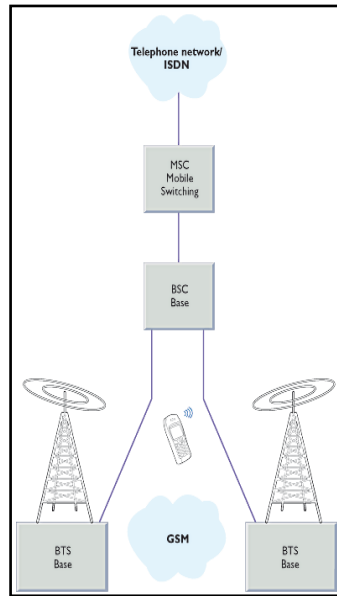


FIG:4.5(b) BASIC GSM NETWORK

4.6 GSM MODEM

A **GSM modem** is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone.

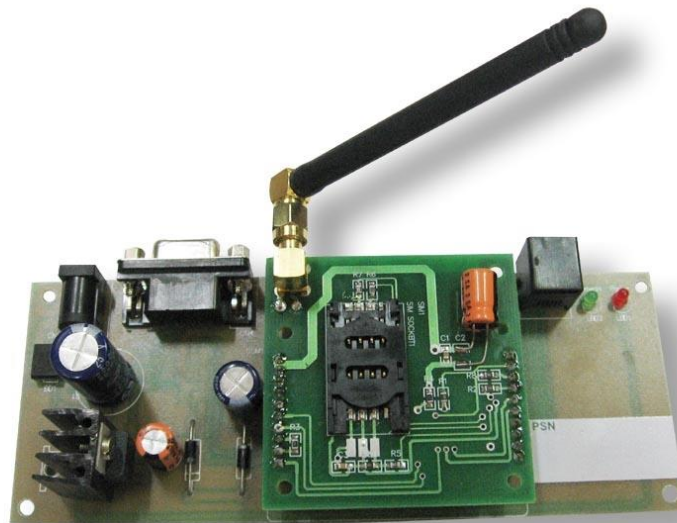


FIG:4.6 GSM MODEM

4.7 ULN2003

RELAY DRIVER:

ULN2003 is a high voltage and high current Darlington transistor array.

DESCRIPTION:

The ULN2003 is a monolithic high voltage and high current Darlington transistor arrays. It consists of seven NPN Darlington pairs that feature high-voltage outputs with common-cathode Clamp diode for switching inductive loads. The collector-current rating of a single Darlington pair is 500mA. The Darlington pairs may be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED gas discharge), line drivers, and logic buffers.

The ULN2003 has a 2.7kW series base resistor for each Darlington pair for operation directly with TTL or 5V CMOS devices.

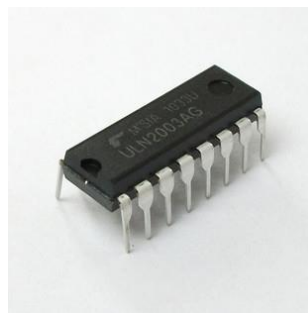


Fig 4.7(): ULN 2003

4.8 RELAY

A relay is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.



FIG:4.8 RELAY

4.9 LED

Light Emitting Diodes (LED) have recently become available that are white and bright, so bright that they seriously compete with incandescent lamps in lighting applications. They are still pretty expensive as compared to a GOW lamp but draw much less current and project a fairly well focused beam.

The diode in the photo came with a neat little reflector that tends to sharpen the beam a little but doesn't seem to add much to the overall intensity.

When run within their ratings, they are more reliable than lamps as well. Red LEDs are now being used in automotive and truck tail lights and in red traffic signal lights. You will be able to detect them because they look like an array of point sources and they go on and off instantly as compared to conventional incandescent lamps.

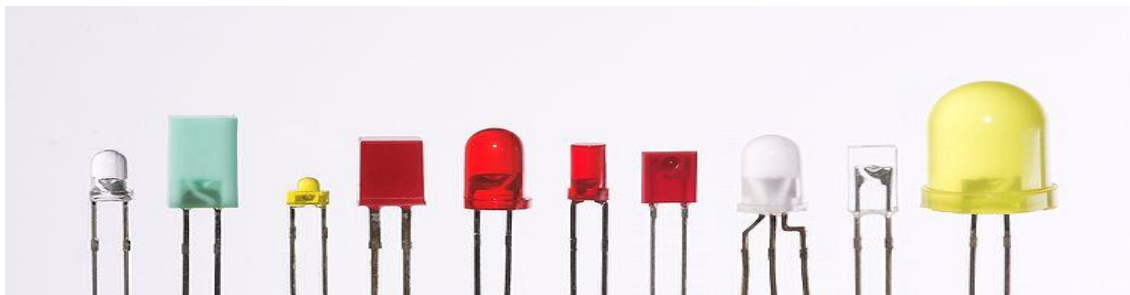


Fig 4.9(b) Different types of LED'S

4.10 LIQUID CRYSTAL DISPLAY (LCD)

Description:

This is the example for the Parallel Port. This example doesn't use the Bi-directional feature found on newer ports, thus it should work with most, if not all Parallel Ports. It however doesn't show the use of the Status Port as an input for a 16 Character x 2 Line LCD Module to the Parallel Port. These LCD Modules are very common these days, and are quite simple to work with, as all the logic required running them is on board.



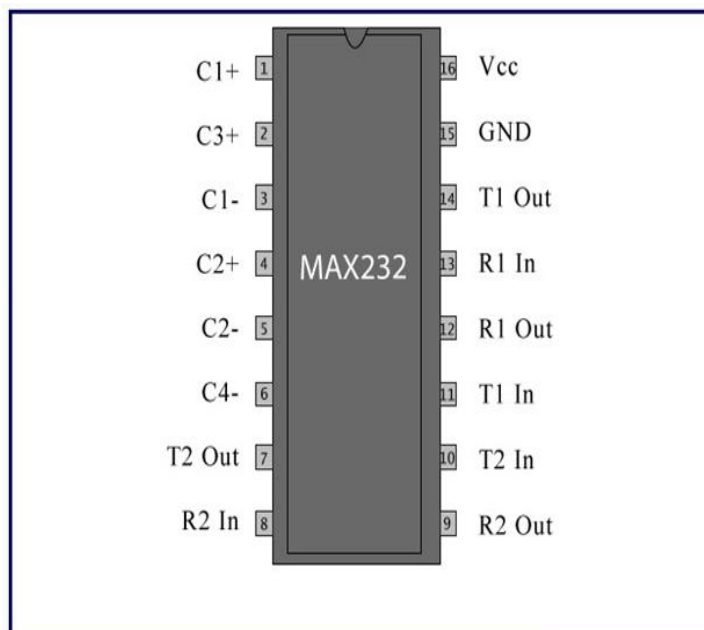
FIG 4.10: LCD

4.11 MAX 232

The MAX232 is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case. The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.

The later MAX232A is backwards compatible with the original MAX232 but may operate at higher baud rates and can use smaller external capacitors (0.1 μ F) in place of the 1.0 μ F capacitors used with the original device. The newer MAX3232 is also backwards compatible, but operates at a broader voltage range, from 3 to 5.5V.



4.12 DB9 CONNECTOR

The DB9 (originally DE-9) connector is an analog 9-pin plug of the D-Sub miniature connector family (D-Sub or Sub-D). The DB9 connector is mainly used for serial connections, allowing for the asynchronous transmission of data as provided for by standard RS-232 (RS-232C).

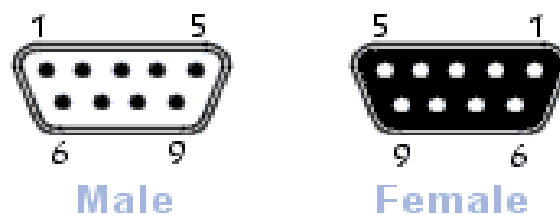


Fig 4.10: DB9 CONNECTOR

4.13 ENERGY METER

An energy or electric meter is a device that measures the amount of electrical energy consumed by a residence, business, or an electrically-powered device.



Fig 4.2.10: Energy Meter

Electric meters are typically calibrated in billing units, the most common one being the kilowatt hour. Periodic readings of electric meters establish billing cycles and energy used during a cycle. In settings when energy savings during certain periods are desired, meters may measure demand, the maximum use of power in some interval. In some areas, the electric rates are higher during certain times of day, to encourage reduction in use. Also, in some areas meters have relays to turn off nonessential equipment.

4.14 OPTOISOLATOR

Opto-isolators, or Opto-couplers, are made up of a light emitting device, and a light sensitive device, all wrapped up in one package, but with no electrical connection between the two, just a beam of light. The light emitter is nearly always an LED. The light sensitive device may be a photodiode, phototransistor, or more esoteric devices such as thyristors, triacse.t.c.

A lot of electronic equipment nowadays is using optocoupler in the circuit. An optocoupler or sometimes refer to as optoisolator allows two circuits to exchange signals yet remain electrically isolated. This is usually accomplished by using light to relay the signal. The standard optocoupler circuits design uses a LED shining on a phototransistor-usually it is anpntransistor and not PNP. The signal is applied to the LED, which then shines on the transistor in the IC. The light is proportional to the signal, so the signal is thus transferred to the phototransistor. Optocouplers may also comes in few module such as the SCR, photodiodes, TRIAC of other semiconductor switch as an output, and incandescent lamps, neon bulbs or other light source.

4.15 1N4007

Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must he kept in mind while using any type of diode.

- 1.Maximum forward current capacity
- 2.Maximum reverse voltage capacity
- 3.Maximum forward voltage capacity



Fig: 1N4007 diodes

4.16 RESISTORS

A resistor is a two-terminal electronic component designed to oppose an electric current by producing a voltage drop between its terminals in proportion to the current, that is, in accordance with Ohm's law:

$$V = IR$$

Resistors are used as part of electrical networks and electronic circuits. They are extremely commonplace in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).



FIG:4.15 RESISTOR

4.12 CAPACITORS

A capacitor or condenser is a passive electronic component consisting of a pair of conductors separated by a dielectric. When a voltage potential difference exists between the conductors, an electric field is present in the dielectric. This field stores energy and produces a mechanical force between the plates. The effect is greatest between wide, flat, parallel, narrowly separated conductors.

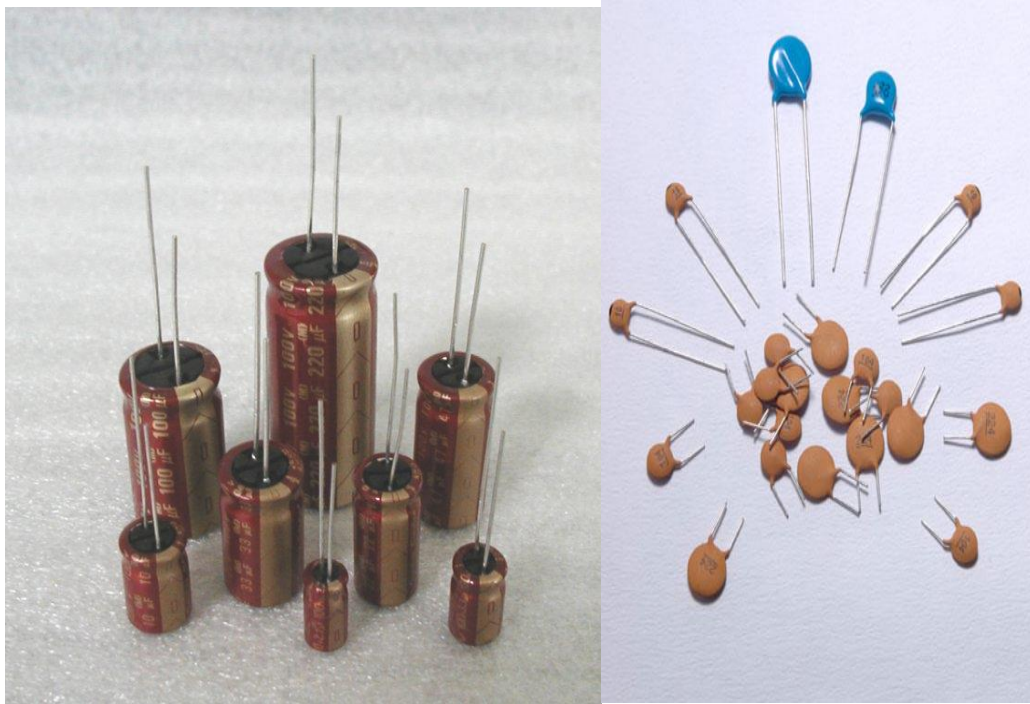


FIG:4.16 CAPACITOR

An ideal capacitor is characterized by a single constant value, capacitance, which is measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In practice, the dielectric between the plates passes a small amount of leakage current. The conductors and leads introduce an equivalent series resistance and the dielectric has an electric field strength limit resulting in a breakdown voltage.

The properties of capacitors in a circuit may determine the resonant frequency and quality factor of a resonant circuit, power dissipation and operating frequency in a digital logic circuit, energy capacity in a high-power system, and many other important aspects.

5. SOFTWARE REQUIREMENT ANALYSIS

5.1 INTRODUCTION

Arduino is an open-source prototyping stage which is based on simple to-utilize hardware and programming. Arduino sheets can read inputs - buzzer on a sensor, a finger on a button, or any message - and transform it into an output - initiating an engine, turning on a LED, publishing something on the web. You can guide your board by sending an arrangement of instructions to the microcontroller on the board. To do as such you need the Arduino programming language, and the Arduino Software (IDE), based on Processing.

Throughout the years Arduino has been used as the brain of thousands of projects, from regular items to complex logical instruments. An overall network of producers - students, specialists, artist, software engineers, and experts - has accumulated around this open-source stage, their commitments have signified a staggering measure of available information that can be of extraordinary help to newbies and specialists alike. Arduino was developed at the Ivrea Interaction Design Institute as a simple tool for quick prototyping, focussed on students without a background in electronics and programming. When it achieved a more extensive network i.e when demand for Arduino boards increased, the Arduino board began changing to adjust to new needs and difficulties, separating its offer from straightforward 8-bit sheets to items for IoT applications, wearable, 3D printing, and implanted conditions. All Arduino boards are totally open-source, enabling users to assemble them autonomously and in the long run, adjust them to their specific needs. The software, too, is open-source, and it is becoming favorite through the commitments of clients around the world.

The Arduino IDE supports the languages C and C++ using special rules of the code. The Arduino IDE supplies a product library from the Wiring venture, which gives numerous regular input and output procedures. The user-written code just requires two essential functions, for beginning the outline and the fundamental program loop, that are arranged and connected with a program stub principle() into an executable cyclic official program with the GNU toolchain, likewise included with the IDE dissemination. The Arduino IDE utilizes the program avr dude to change over the executable code into a content document in The user-write hexadecimal encoding that is stacked into the Arduino board by a loader program in the board's firmware.

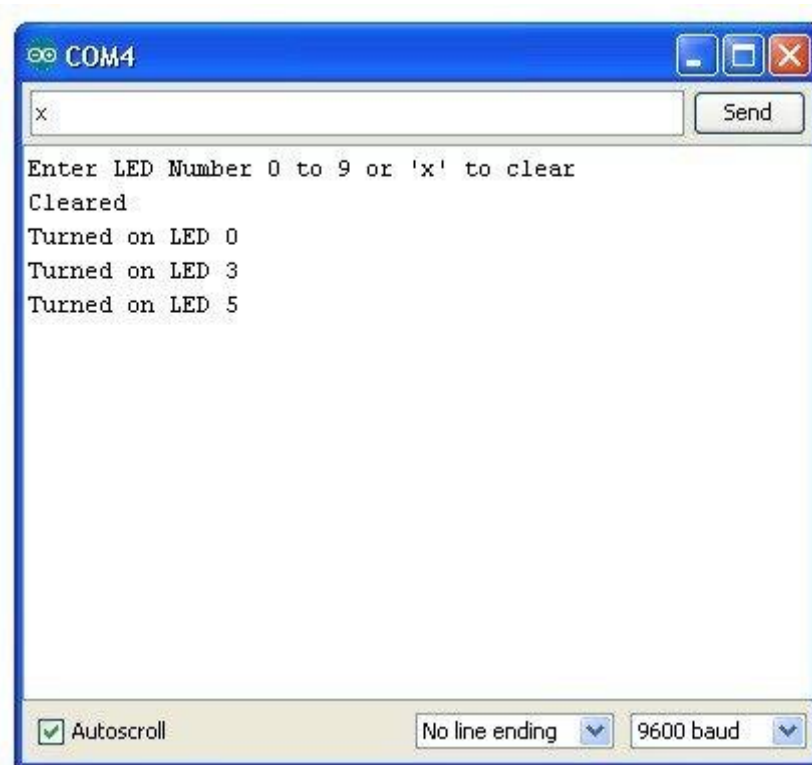
A Pluggable USB center (and Pluggable HID center) enables clients to make low level libraries that enable your board to exploit the MIDI, HID, or Mass Storage practices related with PC USB peripherals. So, your Arduino's USB association can all the more effectively imitate the conduct of a USB-associated MIDI instrument, mouse, PC console, or capacity gadget.

Below is the main screen of the Arduino where program is written and compiled.



A Serial Plotter work has been included, enabling you to locally chart serial information from your Arduino to your PC progressively. In case you're burnt out on observing your Arduino's simple sensor input information pour onto your screen like The Matrix, this resembles a prettier method to picture what's happening. As shown





5.2 GENERAL DESCRIPTION

Arduino

Because of its basic and open client encounter, Arduino has been utilized in a huge number of various tasks and applications. The Arduino programming is easy-to-use and flexible enough for advanced users. yet sufficiently adaptable for beginners.

It works on Mac, Windows, and Linux. Researches and students use it to fabricate minimal effort logical instruments, to demonstrate science and material science standards, or to begin with programming and robotics. Designers and engineers develop intelligent models, performers and specialists utilize it for establishments and to explore different avenues regarding new musical instruments. Creators, obviously, utilize it to manufacture a significant number of the activities shown at the Maker Faire, for instance. Arduino is a key instrument to learn new things. Anybody - youngsters, specialists, craftsmen, developers - can begin tinkering simply following the well-ordered directions of a unit, or sharing thoughts online with different individuals from the Arduino people group.

There are numerous different microcontrollers and microcontroller stages accessible for Physical Computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and numerous others offer comparative usefulness. These apparatuses take the complex

subtle elements of microcontroller programming and wrap it up in a simple to-utilize package. Arduino likewise streamlines the way toward working with microcontrollers, yet it offers some preferred standpoint for instructors, understudies, and intrigued novices over different frameworks:

- Cross-stage** - The Arduino Software (IDE) keeps running on Windows, Macintosh OSX, and Linux working frameworks. Most microcontroller frameworks are constrained to Windows.

- Simple, clear programming condition** - The Arduino Software (IDE) is anything but easy to-use for newbies, yet sufficiently adaptable for cutting edge clients to exploit too. For educators, it's advantageously in view of the Processing programming condition, so students figuring out how to program in that condition will be acquainted with how the Arduino IDE functions.

Open source and extensible programming - The Arduino programming is distributed as open source devices, accessible for augmentation by experienced developers. The dialect can be extended through C++ libraries, and individuals needing to comprehend the specialized points of interest can make the jump from Arduino to the AVR C programming dialect on which it's based. Correspondingly, you can include AVR-C code straightforwardly into your Arduino programs on the off chance that you need to.

Open source and extensible equipment - The designs of the Arduino sheets are distributed under a Creative Commons permit, so experienced circuit planners can make their own particular rendition of the module, broadening it and enhancing it. Indeed, even moderately unpracticed clients can construct the breadboard adaptation of the module with a specific end goal to see how it functions and spare cash.:

Some of the boards used on software:

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

Arduino Mega 2560 Rev. 3 Microcontroller Board is based on the Atmel ATmega2560 8-bit microcontroller (MCU). Arduino Mega 2560 features 54 digital input/output pins (15 of which can be used as PWM outputs) and 16 analog inputs. This Arduino MCU board also includes 4 UARTs (hardware serial ports), a 16MHz crystal oscillator, a USB connection, a power jack, an In-Circuit Serial Programming (ICSP) header, and a reset button.

Mega 2560 incorporates everything the user needs to support the MCU. The client can begin by interfacing the Mega 2560 to a PC with a USB link or by controlling it with an AC-to-DC

connector or battery. Arduino Mega 2560 board is good with most shields intended for the Uno and previous sheets Duemilanove or Diecimila. Mega 2560 is a refresh to the prior Arduino Mega board.

Arduino Mega ADK

This particular adaptation of the Arduino is essentially an Arduino Mega that has been particularly intended for interfacing with Android cell phones

Some of the key features of the Arduino Uno include:

- **An open source plan:** Its benefit being open source is that it has an extensive network of individuals utilizing and investigating it. This makes it simple to discover somebody to enable you to troubleshoot your undertakings.
- **An easy USB interface:** The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol.

6. HARDWARE TESTING

In hardware, a progression test is the checking of an electric circuit to check whether current streams (that it is in certainty a total circuit). A coherence test is performed by putting a little voltage (wired in arrangement with a LED or clamor creating part, for example, a piezoelectric speaker) over the picked way. In the event that electron stream is restrained by broken conductors, harmed parts, or inordinate obstruction, the circuit is "open".

Gadgets that can be utilized to perform progression tests incorporate multi meters which measure present and concentrated congruity analyzers which are less expensive, more fundamental gadgets, for the most part with a basic light that lights up when current streams.

An essential application is the congruity trial of a heap of wires in order to locate the two finishes having a place with a specific one of these wires; there will be a unimportant obstruction between the "right" closures, and just between the "right" finishes.

This test is the performed soon after the equipment patching and arrangement has been finished. This test goes for finding any electrical open ways in the circuit after the welding. Numerous a times, the electrical coherence in the circuit is lost because of ill-advised fastening, incorrectly and harsh treatment of the PCB, ill-advised use of the binding iron, part disappointments and nearness of bugs in the circuit graph. We utilize a multi meter to play out this test. We keep the multi meter in bell mode and interface the ground terminal of the multi meter to the ground. We associate both the terminals over the way that should be checked. On the off chance that there is continuation then you will hear the blare sound.

POWER ON TEST:

This test is performed to check whether the voltage at various terminals is as indicated by the prerequisite or not. We take a multi meter and place it in voltage mode. Keep in mind that this test is performed without microcontroller. Right off the bat, we check the yield of the transformer, regardless of whether we get the required 12 v AC voltage.

At that point we apply this voltage to the power supply circuit. Note that we do this test without microcontroller in light of the fact that if there is any inordinate voltage, this may prompt harming the controller. We check for the contribution to the voltage controller i.e., are we getting a contribution of 12v and a yield of 5v. This 5v yield is given to the microcontrollers' 40th stick. Consequently we check for the voltage level at 40th stick. So also, we check for alternate terminals for the required voltage. Along these lines we can guarantee that the voltage at all the terminals is according to the necessity.

8. IMPLEMENTATION

8.1 IMPLEMENTATION OF THE PROJECT

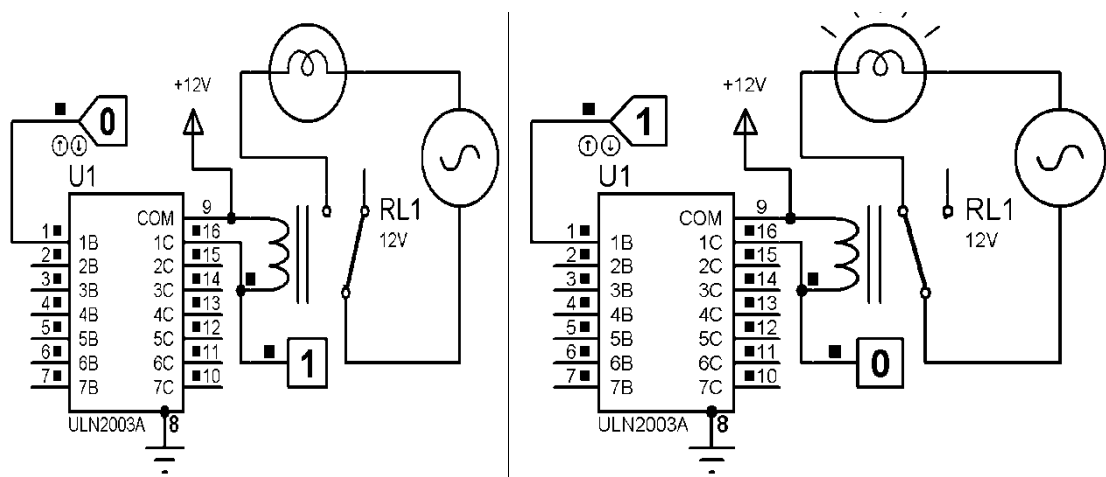
POWER SUPPLY

The circuit uses standard power supply comprising of a step-down transformer from 230V to 12V and 4 diodes forming a bridge rectifier that delivers pulsating dc which is then filtered by an electrolytic capacitor of about 470 μ F to 1000 μ F. The filtered dc being unregulated, IC LM7805 is used to get 5V DC constant at its pin no 3 irrespective of input DC varying from 7V to 15V. The input dc shall be varying in the event of input ac at 230volts section varies from 160V to 270V in the ratio of the transformer primary voltage V1 to secondary voltage V2 governed by the formula $V1/V2=N1/N2$. As $N1/N2$ i.e. no. of turns in the primary to the no. of turns in the secondary remains unchanged V2 is directly proportional to V1. Thus if the transformer delivers 12V at 220V input it will give 8.72V at 160V. Similarly at 270V it will give 14.72V. Thus the dc voltage at the input of the regulator changes from about 8V to 15V because of A.C voltage variation from 160V to 270V the regulator output will remain constant at 5V.

The regulated 5V DC is further filtered by a small electrolytic capacitor of 10 μ F for any noise so generated by the circuit. One LED is connected of this 5V point in series with a current limiting resistor of 330 Ω to the ground i.e., negative voltage to indicate 5V power supply availability. The unregulated 12V point is used for other applications as and when required.

ULN 2003 RELAY DRIVER IC

ULN2003 is an IC which is used to interface relay with the microcontroller since the output of the micro controller is maximum 5V with too little current delivery and is not practicable to operate a relay with that voltage. ULN2003 is a relay driver IC consisting of a set of Darlington transistors. If logic high is given to the IC as input then its output will be logic low but not the vice versa. Here in ULN2003 pin 1 to 7 are IC inputs and 10 to 16 are IC outputs. If logic 1 is given to its pin no 1 the corresponding pin 16 goes low. If a relay coil is connected from +ve to the output pin of the uln2003, (the relay driver) then the relay contacts change their position from normally open to close the circuit as shown below for the load on (say a lamp to start glowing). If logic 0 is given at the input the relay switches off. Similarly upto seven relays can be used for seven different loads to be switched on by the normally open (NO) contact or switched off by the normally closed contact (NC)



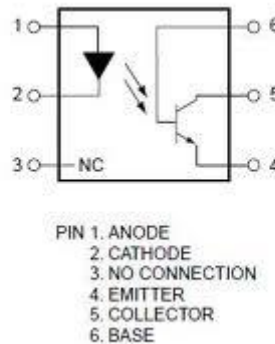
Load off Load on

OPTO COUPLER:

Opto-isolators, or Opto-couplers, are made up of a light emitting device, and a light sensitive device, all wrapped up in one package, but with no electrical connection between the two, just a beam of light. The light emitter is nearly always an LED. The light sensitive device is a phototransistor. When signal is given at pin 1 and 2 is grounded the transistor inside the opto isolator conducts between 5 and 4 which is used in the project by interfacing to the

microcontroller for change of logic state from the energy meter unit pulsing led in series with the opto led at pin no 1 and 2 for pulse counting purpose.

Functional Block Diagram



MAX232

The MAX232 used in the project is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits like microcontroller. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

OPERATION

CONNECTIONS:

The output of power supply which is 5v is connected to 40th pin of microcontroller and Gnd is connected to 20th pin of microcontroller. Pin 2.0 to pin 2.7 of port 2 of MC is connected to data pins i.e., D₀ to D₇ of LCD display. Pins 4, 5, 6 i.e., RS, RW, EN. Of LCD are given to p 0.0 to p 0.2 of port of 0 of MC. Pin 3.0 and pin 3.1 of port 3 of MC are connected to pin 11 and pin 12 of Max232. Pin 3.3 of port 3 of MC is connected to pin 5 of Opto coupler. Pin 1 & 2 of Opto coupler are connected to energy meter. Pin 14 & 13 of Max232 are given to pins 2 & 3 of DB9 male connector. Pin 2 & pin 5 of DB9 Female Connector are given to GSM modem. The output of the power supply which is 5v is connected to the 40th pin of MC and GND is connected to its 20th pin. Pin 31 of MC is connected to 2nd pin of Max232. Pin 14 of Max232 is connected to pin 3 of DB9 Male connector. Pin 13 of Max232 is connected to pin 2 of DB9 connector. Pin 11 & 12 of Max232 is connected to port 3.0 and 3.1 of MC. Port 2.0 to 2.7 of MC are connected to pin D₀ – D₇ of LCD display. Port 0.0 to 0.2 of MC are connected to 4, 5, 6 pin's of LCD display. Port

0.3 to 0.6 of MC are connected to pin's (1, 2, 3, 4) pin's of ULN2003A. Pins (1, 2, 3, 4) C of ULN2003A are connected to relay's.

WORKING:

The project uses a commercial digital energy meter and derives positive pulses from the same by taking a connection from the pulsing LED. The pulsing LED pulses 3200 times for 1 unit of electrical energy i.e., 1 kilo watt hour. As it is not feasible to wait for consuming 1000 Wt Hr. the program assumes 10 pulses for unit, which is fed to the microcontroller to send 1 unit consumption through the GSM Modem duly interfaced to the microcontroller through Max232, Therefore the 1 unit so sent shown have to be read as 1/3200 unit.

The project uses a GSM modem. Upon a missed a call to the project board GSM modem ,the caller's number gets stored in the microcontroller for further communication to that number only. This gives the unique flexibility for changing number by the user at will without going through the cumbersome process of writing the number while burning the program on to the microcontroller .Thus in that case only that number is used for communication and the user has no option to change that . The 'AT' commands from which are received by the micro controller through level shifted IC Max232. The program while executed drives relays from the microcontroller through relay driver IC ULN2003. Loads are switched ON and switched OFF based on the corresponding command sent from the GSM modem. As per the program, an acknowledgement is received by ansms being sent from the controller to the user upon the status of the load depending on port 0 logic outputs responsible for operating the loads. The complete operation is displayed on the LCD screen.

9. USER MANUAL

The project uses a commercial digital energy meter and derives positive pulses from the same by taking a connection from the pulsing LED. The pulsing LED pulses 3200 times for 1 unit of electrical energy i.e., 1 kilo watt hour. As it is not feasible to wait for consuming 1000 Wt Hr. the program assumes 10 pulses for unit, which is fed to the microcontroller to send 1 unit consumption through the GSM Modem duly interfaced to the microcontroller through Max232, Therefore the 1 unit so sent shown have to be read as 1/3200 unit.

The project uses a GSM modem. Upon a missed a call to the project board GSM modem ,the caller's number gets stored in the microcontroller for further communication to that number only. This gives the unique flexibility for changing number by the user at will without going

through the cumbersome process of writing the number while burning the program on to the microcontroller .Thus in that case only that number is used for communication and the user has no option to change that . The ‘AT’ commands from which are received by the micro controller through level shifted IC Max232. The program while executed drives relays from the microcontroller through relay driver IC ULN2003. Loads are switched ON and switched OFF based on the corresponding command sent from the GSM modem. As per the program, an acknowledgement is received by ansms being sent from the controller to the user upon the status of the load depending on port 0 logic outputs responsible for operating the loads. The complete operation is displayed on the LCD screen.

Operating procedure:


1. The mobile number stored in the microcontroller will get constant messages regarding the unit consumed and the price.
2. To switch the load on and off, do the following operations from any other mobile number not from the phone whose number is stored and send the message to SIM in the modem:
 - i) Send 1 to switch ON load 1.
 - ii) Send 2 to switch OFF load 1.
 - iii) Send 3 to switch ON load 2.
 - iv) Send 4 to switch OFF load 2.
 - v) Send 5 to switch ON load 3.
 - vi) Send 6 to switch OFF load 3.
 - vii) Send 7 to switch ON load 4.
 - viii) Send 8 to switch OFF load 4.
2. To get the status of the loads send ‘S’ from any other mobile phone(whose number is not stored inside the controller) and send it to SIM inside the modem. Status will be sent to mobile whose number is stored on the controller.

10.CODING

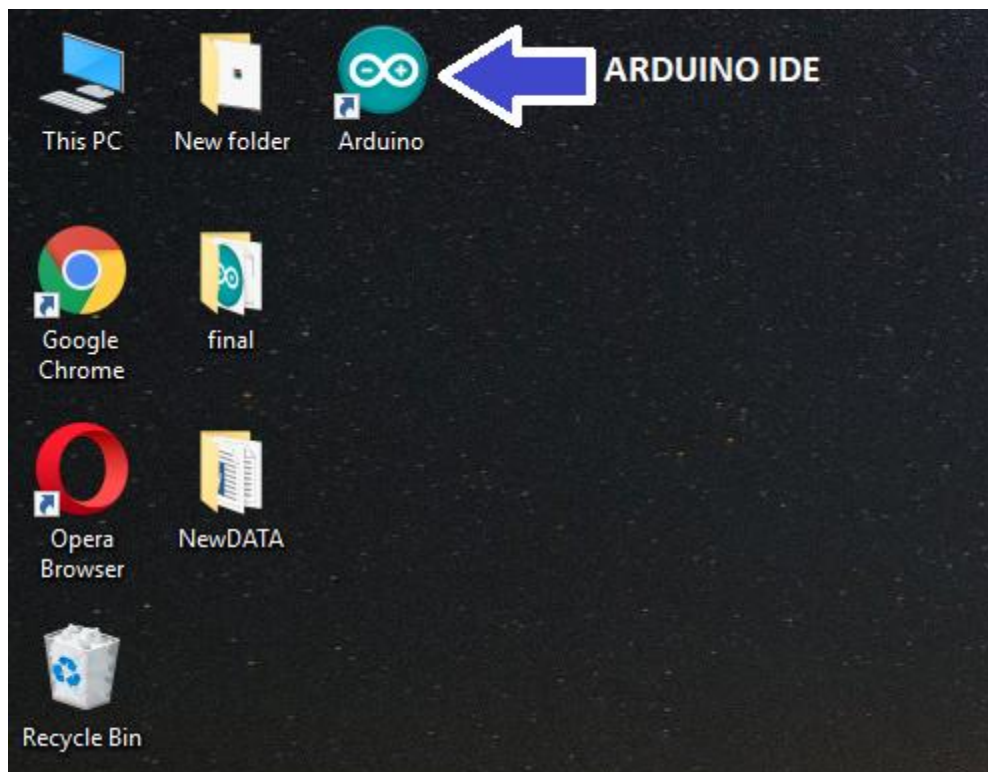
Arduino IDE

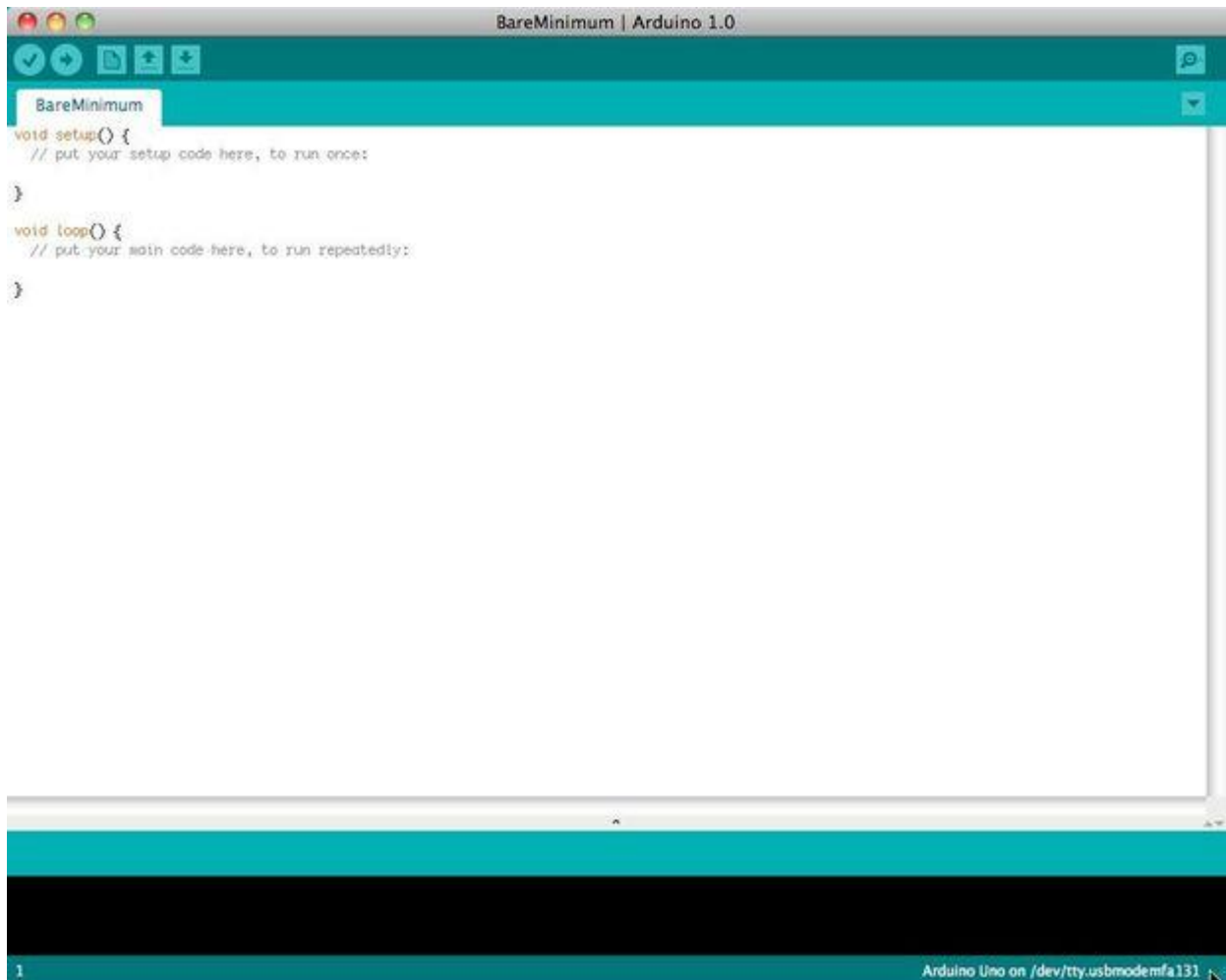
The Arduino site has loads of awesome data and is additionally where you download the product. While you can most likely do parts with the Arduino Web Editor, we propose introducing the Arduino IDE. IDE is easy for geek developers "Software engineers Editor" or all the more in fact "Integrated Development Environment". The Arduino IDE has all the usefulness required to alter Arduino code, examples from an extensive online library, access utility libraries, and more.

Before you can begin doing anything with the Arduino, you have to download and introduce the Arduino IDE (integrated development environment). Starting here on we will refer the Arduino IDE as the Arduino Programmer.

Go to  <https://www.arduino.cc/en/Main/Software> and select the download for your computer. It will include not just the IDE but also device drivers.c

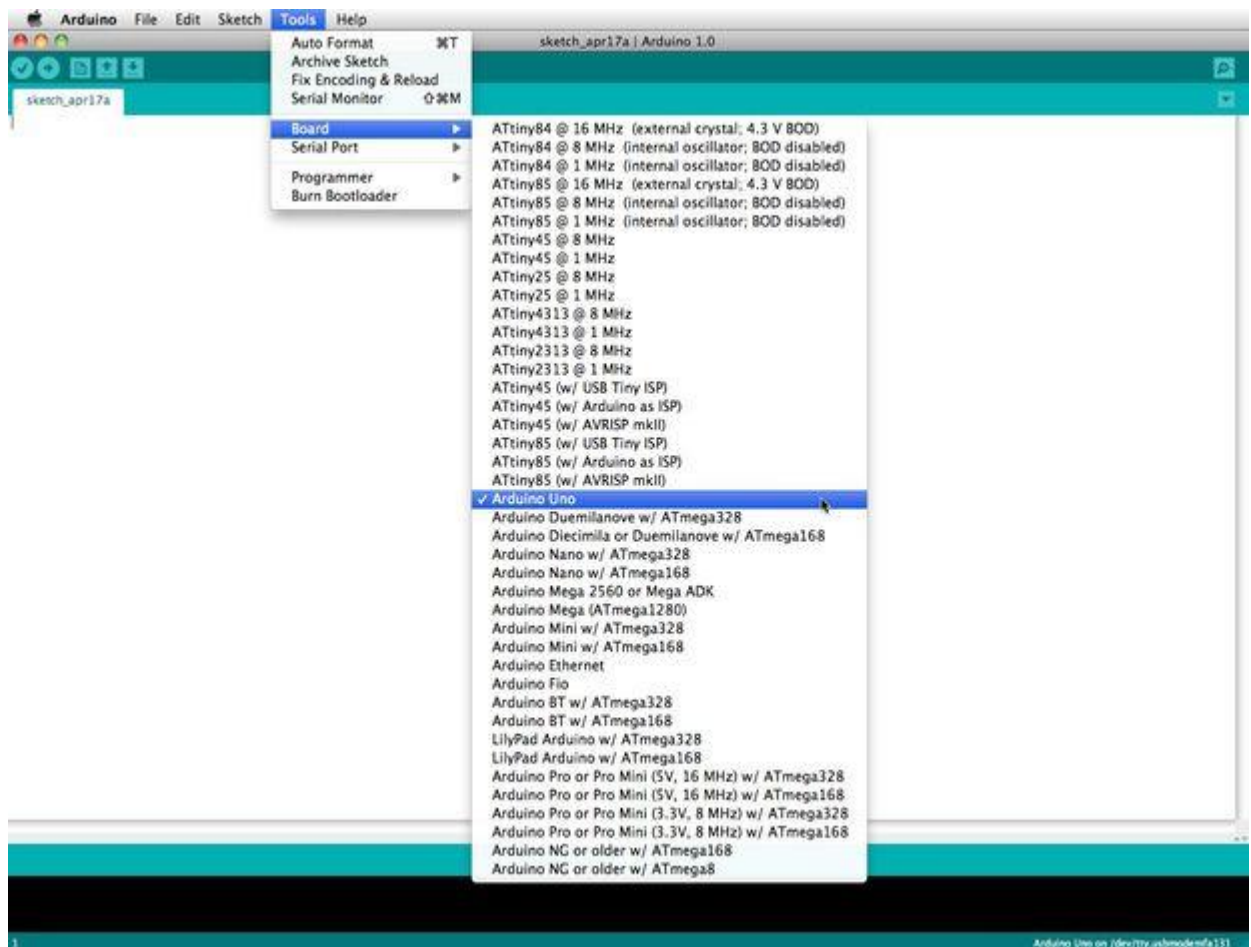
Once you've downloaded the software, install it. For Windows, you may need to install Drivers that are included in the Arduino IDE download.

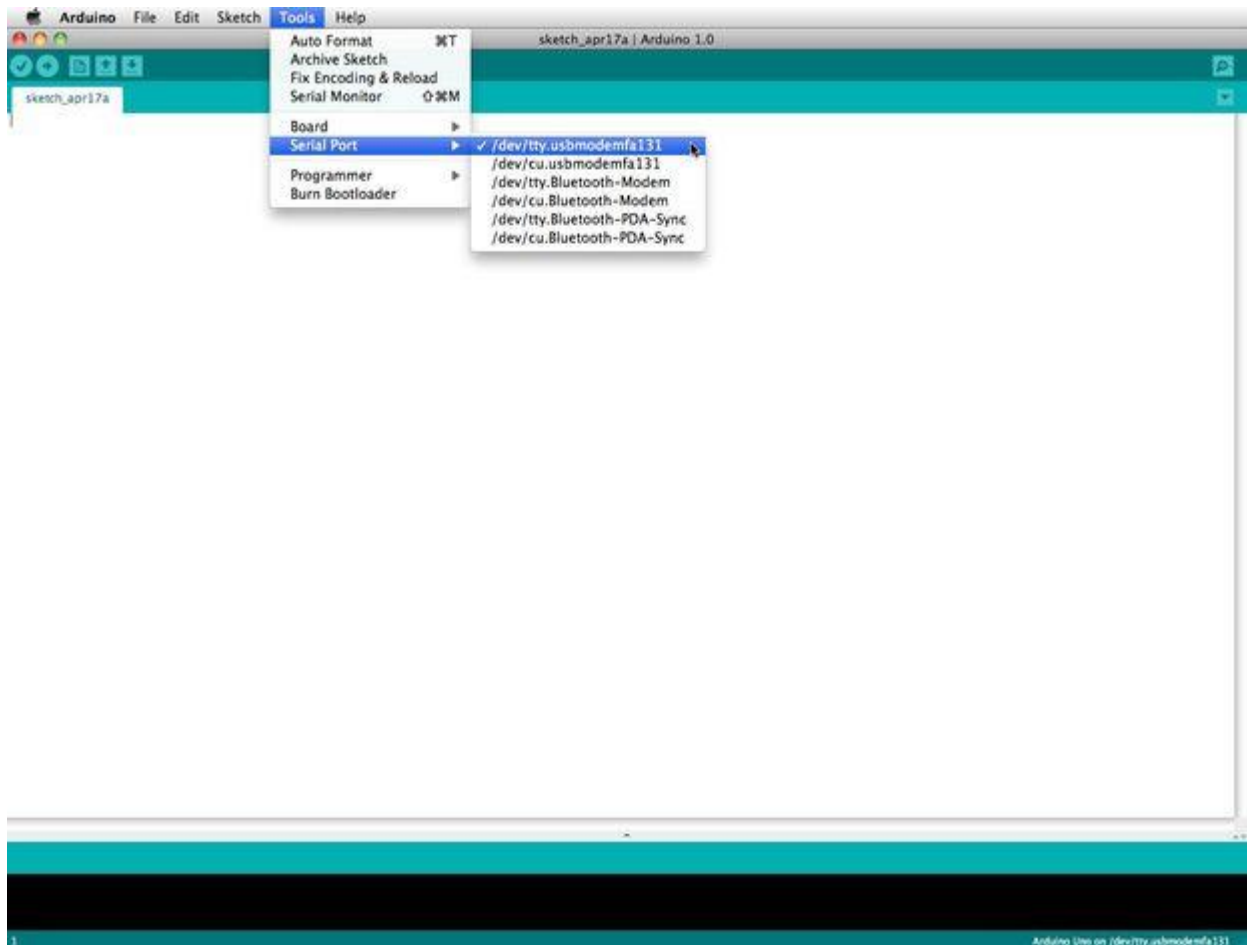




The Arduino Programmer depends on the Processing IDE and utilizes a variety of the C and C++ programming Language.

Settings





Before you can start doing anything in the Arduino programmer, you must set the board-type and serial port.

To set the board, go to the following:

Tools --> Boards

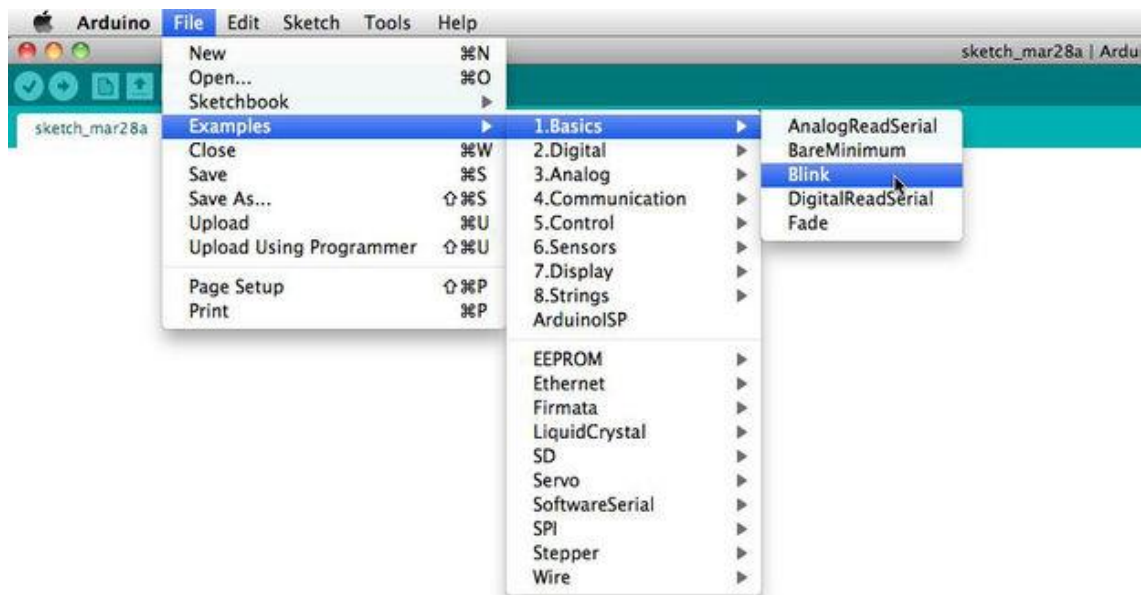
Select the version of board that you are using. Since I have an Arduino Uno plugged in, I obviously selected "Arduino Uno."

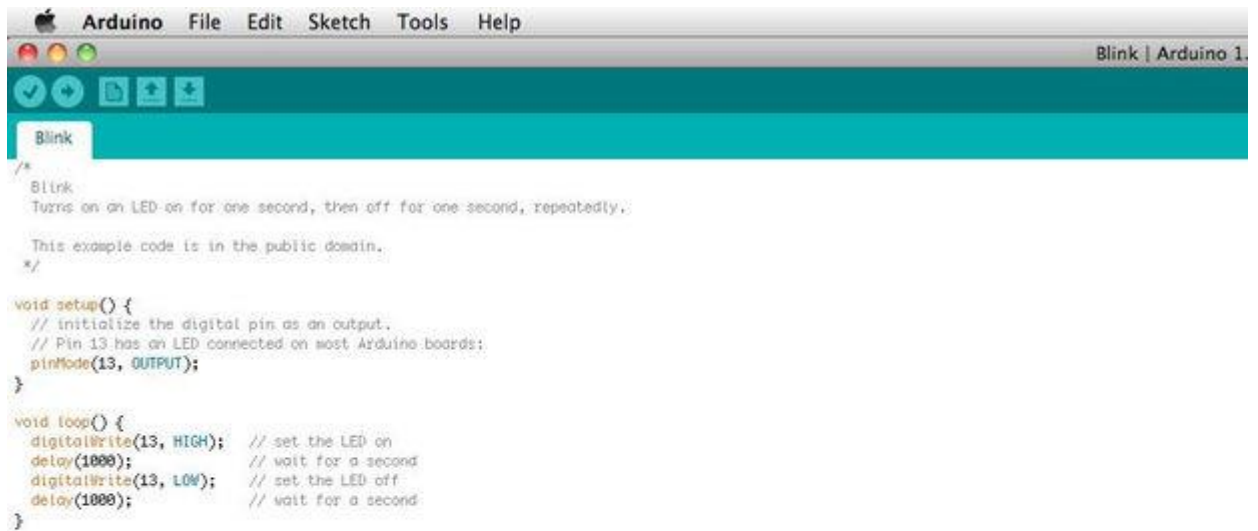
To set the serial port, go to the following:

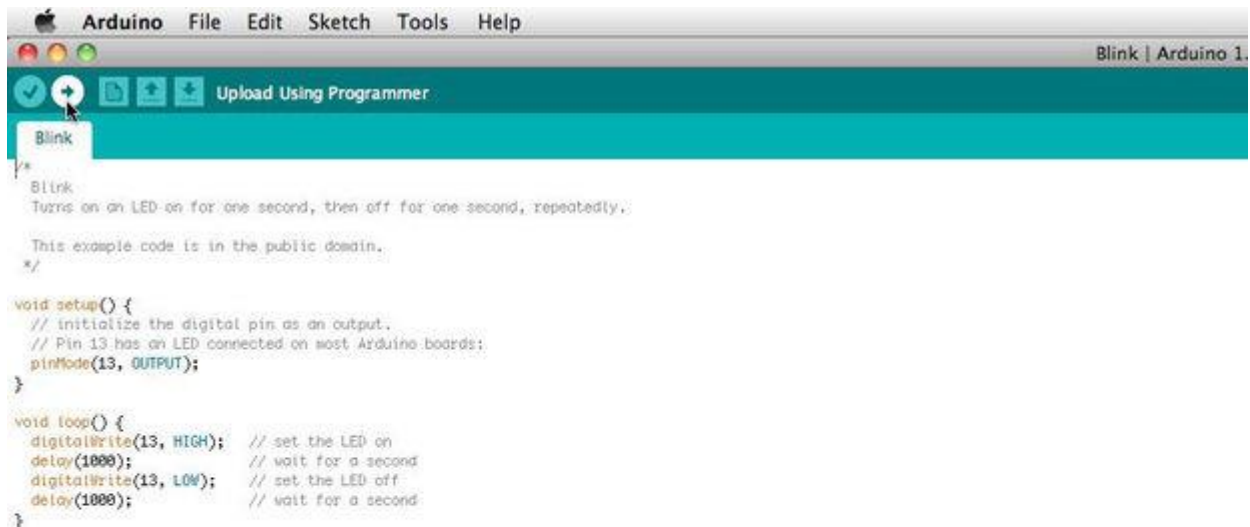
Tools --> Serial Port

Select the serial port that looks like:

/dev/tty.usbmodem [random numbers]







Arduino programs are called sketches. The Arduino software engineer accompanies a huge amount of example sketches preloaded. This is extraordinary on the grounds that regardless of whether you have never modified anything in your life, you can stack one of these portrayals and get the Arduino to accomplish something.

Load the Blink example

The Arduino IDE includes a Library Manager that's used to download extension libraries. Some of those libraries are written by the Arduino organization while others are written by the community. In the Sketch menu, under the Include Library choice is Manage Libraries. Click on that and a window pops up letting you search for and install libraries.

A large number of example programs are already made available to help you get started with the Arduino. In the File menu is the Examples choice which presents a variety of example programs. Which gives us the cue for the next section. The Blink program is a useless example. It just blinks one of the LED's on the board. However, this application is a quick way to verify the hardware is setup correctly and to learn a couple things about the Arduino.

If there is an Angel statue nearby, however, don't run this program because whatever you do when near a Weeping Angel, Don't Blink

With that out of the way, and shooing away any wayward Angels, in the Examples menu select 01.Basic and Blink. The IDE will open a new editing window containing the sample application.

The programming language is more-or-less-don't-look-too-closely C, a venerable programming language. Learning to program the Arduino is a chance to learn C, more-or-less, if you don't already know that language.

```
// the setup function runs once when you press reset or power the board

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

The `setup` function in an Arduino sketch is run once when the program starts. In this case it sets up the named pin, `LED_BUILTIN`, as an output pin.

```
// the loop function runs over and over again forever

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

The `loop` function runs continuously and is meant to handle "events" and make behavior changes by either reading pin states or changing pin states. In this case the `LED_BUILTIN` pin is first set `HIGH`, then set `LOW`, with 1000 milliseconds between each.

In case it's not clear -- this will change the output state of one of the pins on the Arduino GPIO. This particular pin corresponds to an LED on the board, and therefore the LED will blink on and off repeatedly.

With your Arduino plugged in, there are two buttons to click in the IDE user interface. They are the Verify button, which looks like a checkmark, in the toolbar, followed by the Upload button, which looks like a right-ward-pointing-arrow. Upload causes the program to load into the Arduino.

CODE

```
#include <SoftwareSerial.h>

#include <LiquidCrystal.h>

#include <TimerOne.h>

SoftwareSerial gsm(3, 4);

LiquidCrystal lcd(5, 6, 7, 8, 9, 10);

float x = 9.9;

#define RECEIVINGCALL 1

#define IDLE 2

#define BUSY 3

#define NO_ANSWER 4

#define TALKING 5

#define ERROR 6


#define Pin "pin"

#define ok "OK"

#define ERROR_str "ERROR"

#define Connected "Connected"

#define Connecting "connecting"

#define CLCC "+CLCC:"

#define atCLCC "AT+CLCC"

#define CMGF "AT+CMGF=1"

#define CMGL "+CMGL:"

#define CMGS "AT+CMGS=\''"

#define CMGD "AT+CMGD="

#define UNREAD "AT+CMGL=\''REC UNREAD\''"

#define CMGDA "AT+CMGDA=\''DEL ALL\''"
```

```
#define CNMI "AT+CNMI=0,0,0,0"
```

```
boolean gsm_connect = false;
```

```
boolean notConnected = true;
```

```
int no_configured = 0;
```

```
unsigned long current_time ;
```

```
String numtel;
```

```
String msg = "";
```

```
String indata = "";
```

```
String inputString = "";
```

```
String msg_index;
```

```
int index1;
```

```
int index2;
```

```
String call_number, num;
```

```
String sms_num, sms; // = get_sender_number();
```

```
float tempC;
```

```
int reading;
```

```
long time1;
```

```
long time2;
```

```
int beats, last_rate;
```

```
int smssend = 0;
```

```
//byte oldSample, sample;
```

```
unsigned int oldSample, sample;
```

```
int count = 0;
```

```
//int x = 30;
```

```
int y = 120;  
unsigned long pulse;  
unsigned int i = 1;  
long measurementStartTime = 0;  
int e_chk1 = 0;  
int e_chk2 = 0;  
int menu = 1;  
int menu_sel = 1;  
const int entr = 11;  
const int bck = 12;  
const int up = 13;  
const int dwn = A0;  
const int meter_pulse = 2;  
const int load1 = A5;  
const int load2 = A4;  
const int load3 = A3;  
const int load4 = A2;  
  
double estimated_uses;  
double estimated_cost;  
volatile double cost;  
volatile double unit;  
volatile double current_use = 0;  
volatile double last_use = 0;  
volatile double total_use = 0;  
volatile int sec = 0, trigger = 0, pulse_time;
```

```
boolean valid_sms;
```

```
char c, char1[10], char2[10];
```

```
String str = "";
```

```
//String msg = "";
```

```
int load1_satus = 0;
```

```
int load2_satus = 0;
```

```
int load3_satus = 0;
```

```
int load4_satus = 0;
```

```
/*
```

```
int distress = 0;
```

```
int ret = 0;
```

```
unsigned long lasttime=0;
```

```
unsigned long interval=60000;
```

```
unsigned long currenttime=0;
```

```
float lat_int;
```

```
float lon_int;
```

```
String lat="latitude=";
```

```
String lon="\nlongitude=";
```

```
String site="\nhttp://maps.google.com/maps?f=q&q=";
```

```
String help1="Soldier need help\n";
```

```

const int buzzer = 4;

const int tempPin = A5;

const int HR_RX = 9;

const int HELP = 5;

const int entr=6;

const int bck=7;

const int up=8;

const int dwn=12; // the number of the pushbutton pin
*/

void setup()
{
  lcd.begin(16, 2);
  gsm.begin(9600);
  Serial.begin(9600);
  Serial.println("hii");
  pinMode(entr, INPUT);
  pinMode(bck, INPUT);
  pinMode (up, INPUT);
  pinMode (dwn, INPUT);
  pinMode (load1, OUTPUT);
  pinMode (load2, OUTPUT);
  pinMode (load3, OUTPUT);
  pinMode (load4, OUTPUT);
  digitalWrite(load1, LOW);
  digitalWrite(load2, LOW);
  digitalWrite(load3, LOW);
  digitalWrite(load4, LOW);

```

```

//lcd.clear();

lcd.clear();

lcd.print(F("connecting to "));

lcd.setCursor(0, 1);

lcd.print(F("gsm"));

while (1)
{
    if (connect_gsm())
    {
        lcd.clear();

        lcd.print(Connected);

        Serial.println(F("GSM connected"));

        break;
    }
}

delay(2000);

lcd.clear();

lcd.print(F("Waiting for call"));

while ((no_configured != 1))
{
    switch (call_status())
    {

        case IDLE: // Nothing is happening

            break;


        case RECEIVINGCALL: // Yes! Someone is calling us

            // Serial.println(RECEIVING_CALL);

```

```

    call_number = get_call_number();

    lcd.clear();

    lcd.print(F("Receiving call"));

    lcd.setCursor(0, 1);

    Serial.println(call_number);

    lcd.print(call_number);

    hangcall();

    delay(1500);

    numtel = call_number;

    lcd.clear();

    lcd.print(F("Sending SMS"));

    sendsms(F("Number configured."), numtel);

    lcd.clear();

    lcd.print(F("SMS sent.."));

    no_configured = 1;

    lcd.clear();

    lcd.print(F("Energymeterwith"));

    lcd.setCursor(0, 1);

    lcd.print(F(" Load Control "));

    delay(2000);
}

}

Timer1.initialize(1000000);           // Initialize TimerOne library for the freq we
need

Timer1.attachInterrupt(estimate, 1000000);

attachInterrupt(0, meterpulse_detect, FALLING);

```



```

    delay(1000);
}

void meterpulse_detect()
{
    cost = cost + x / 1600;
    unit = (unit + 0.000625);
    trigger++;
}

void estimate()
{
    // current_use=unit;
    // total_use=current_use-last_use;
    // sec++;
    if (trigger == 1)
    {
        // last_use=total_use;
        //total_use=0;
        sec++;
    }
    else if (trigger == 2)
    {
        trigger = 0;
        pulse_time = sec;
    }
}

void loop()

```

```

{
  while (1)
  {
    while (menu == 1)
    {
      if (menu_sel == 1)
      {
        lcd.clear();
        lcd.print(F("->>1.Start"));
        lcd.setCursor(0, 1);
        lcd.print(F(" 2.Setting"));
      }
      else if (menu_sel == 2)
      {
        lcd.clear();
        lcd.print(F(" 1.Start"));
        lcd.setCursor(0, 1);
        lcd.print(F("->>2.Setting"));
      }
      if (digitalRead(entr) == LOW)
        menu = 0;
      else if (digitalRead(up) == 0)
        menu_sel = 1;
      else if (digitalRead(dwn) == 0)
        menu_sel = 2;

      delay(300);
    }
  }
}

```

```

}

if (menu_sel == 1)
{
    // lcd.clear();

    // lcd.print(F("Initiallizing....."));

    // delay(500);

    // lasttime = millis();

    while (digitalRead(bck) == HIGH)
    {
        lcd.clear();

        lcd.print("UNITS:");

        lcd.setCursor(0, 1);

        lcd.print("COST: ");

        lcd.setCursor(6, 0);

        lcd.print(unit, 4);

        lcd.print("Kwh  ");

        lcd.setCursor(6, 1);

        lcd.print(cost, 4);

        if (sms_available(sms_num, msg))
        {
            lcd.clear();

            lcd.print(sms_num);

            lcd.setCursor(0, 1);

            lcd.print(msg);

            delay(2000);

            checksms(sms_num, msg);
        }
    }
}

```

```

    delay(400);
}
menu = 1;
menu_sel = 1;
//e_chk1 = 0;
//e_chk2 = 0;
delay(500);
}
if (menu_sel == 2)
{
    lcd.clear();
    lcd.print("Set unit cost");
    lcd.setCursor(5, 1);
    lcd.print(x);
    delay(1000);
    while (digitalRead(entr) == HIGH)
    {
        lcd.setCursor(5, 1);
        lcd.print(x);
        if (digitalRead(up) == LOW)
        {
            x = x + 0.1;
            delay(200);
        }
        else if ((digitalRead(dwn) == LOW) && (x > 0))
        {
            x = x - 0.1;;

```

```

        delay(200);

    }

}

delay(500);

menu = 1;

}

}

}

/*

currenttime=millis();

if((currenttime-lasttime) >= interval)

{

    lasttime = currenttime;

    gps_track(0);

}

lcd.clear();

lcd.print(F("Beats/minute:"));

lcd.print(beats);

lcd.setCursor(0,1);

lcd.print(F("Temp:"));

lcd.print(tempC);

lcd.print(F("\337C"));

hbt();

if(beats < x)

{

    e_chk2 = 0;

    e_chk1++;

```

```

if(e_chk1 >= 5)
{
    tone(buzzer,50,5000);

    lcd.clear();

    lcd.print(F("heart beats are"));

    lcd.setCursor(0, 1);

    lcd.print("LOW");

    delay(2000);

    if((beats < x) && (smssend==0))
    {
        sendsms(F("heart beats are LOW"),numtel);

        smssend=1;
    }

    e_chk1 = 0;
}
}

else if(beats > y)
{
    e_chk1 = 0;

    e_chk2++;

    if(e_chk2 >= 5)
    {
        tone(buzzer,50,5000);

        lcd.clear();

        lcd.print(F("heart beats are"));

        lcd.setCursor(0, 1);

        lcd.print("HIGH");
    }
}

```

```

    delay(2000);
    if((beats > y) && (smssend==0))
    {
        sendsms(F("heart beats are HIGH"),numtel);
        smssend=1;
    }
    e_chk2 = 0;
}
}
else
{
    e_chk1 = 0;
    e_chk2 = 0;
    smssend=0;
}
if(digitalRead(HELP) == LOW)
{
    distress = 1;
    break;
}
}
menu = 1;
menu_sel = 1;
delay(500);
}

if(menu_sel == 2 && distress == 0)

```

```

{
lcd.clear();
lcd.print(F("Set lower limit"));
lcd.setCursor(8,1);
lcd.print(x);
delay(1000);
while(digitalRead(entr) == HIGH)
{
    lcd.setCursor(8,1);
    lcd.print(x);
    if (digitalRead(up) == LOW)
    {
        x = x+10;
        if(x>70)
        {
            x=30;
        }
        delay(200);
    }
    else if(digitalRead(dwn) == LOW)
    {
        x = x-10;
        if(x<30)
        {
            x=70;
        }
        delay(200);
    }
}

```



```

    }
}
delay(500);

lcd.clear();
lcd.print(F("Set higher limit"));
lcd.setCursor(8,1);
lcd.print(y);
lcd.print(" ");
while(digitalRead(entr) == HIGH)
{
    lcd.setCursor(8,1);
    lcd.print(y);
    lcd.print(" ");
    if (digitalRead(up)==LOW)
    {
        y = y+10;
        if(y>200)
        {
            y=70;
        }
        delay(500);
    }
    else if(digitalRead(dwn) == LOW)
    {
        y = y-10;
        if(y<70)

```

```

    {
        y=200;
    }
    delay(500);
}
}
delay(500);

menu = 1;
}

if(distress == 1 || digitalRead(HELP) == LOW)
{
    gps_track(1);
    lcd.clear();
    lcd.print(F("Distress Signal"));
    lcd.setCursor(0,1);
    lcd.print("Sent!");
    while(digitalRead(bck) == HIGH);
    menu = 1;
    distress = 0;
    while(digitalRead(bck) == HIGH);
}
}
}
*/
/*

```

```

void hbt()
{
int k=0;
long t_out1 = millis();
bool zero = false;
while(k<10)
{
if(digitalRead(HR_RX))
{
t_out1 = millis();
if(k==0)
time1=millis();
k++;
while(digitalRead(HR_RX));
}
if(millis() - t_out1 > 2000)
{
zero = true;
break;
}
}
if(!zero)
{
time2=millis();
beats=time2-time1;
beats=beats/10;
beats=60000/beats;

```

```

    if(beats >= 200)
    {
        beats = last_rate;
    }
    // lcd.clear();
}
else
{
    k=0;
    beats=0;
}
lcd.clear();
lcd.print(F("Beats/minute:"));
lcd.print(beats);
lcd.print(" ");
last_rate = beats;
k=0;
temp();
}

void temp()
{
    reading = analogRead(tempPin);
    tempC = reading / 9.31;
    lcd.setCursor(0,1);
    lcd.print(F("Temp:"));
    lcd.print(tempC);

```

```

lcd.print("\337C");
if(tempC>=40)
{
    lcd.clear();
    lcd.print("Temperature is ");
    lcd.setCursor(0, 1);
    lcd.print("HIGH");
    tone(buzzer,50,3000);
    delay(1000);
    sendsms(F("Temperature is HIGH"),numtel);
}
else if(tempC<=23)
{
    lcd.clear();
    lcd.print(F("Temperature is "));
    lcd.setCursor(0, 1);
    lcd.print("LOW");
    tone(buzzer,50,3000);
    delay(1000);
    sendsms(F("Temperature is LOW"),numtel);
}
delay(10);
}

void gps_track(int mode)
{
    lcd.clear();

```

```

lcd.print(F("tracking..."));

char char1[10],char2[10],char3[5],char4[7];

String slat,slon,msg,sbeat,stemp;

lat_int = 0;

lon_int = 0;

delay(1500);

indata = sendcmd("AT+CGNSINF",1);

if(indata.indexOf("+CGNSINF:")>=0)
{
    Serial.print("indata:" );
    Serial.println(indata);
    int index1 = indata.indexOf(',');
    String s2 = indata.substring(index1+1);
    index1 = s2.indexOf(',');
    s2 = s2.substring(index1+1);
    index1 = s2.indexOf(',');
    s2 = s2.substring(index1+1);
    Serial.println(s2);
    int index2 = s2.indexOf(',');
    slat = s2.substring(0,index2 - 1);
    Serial.print("lat = ");
    lat_int = slat.toFloat();
    Serial.println(lon_int);
    String s3 = s2.substring(index2+1);
    Serial.println(s3);
    int index3 = s3.indexOf(',');
    slon = s3.substring(0,index3 - 1);

```

```

lon_int = slon.toFloat();

Serial.print("lon = ");

Serial.println(lon_int);
}

if(!(lat_int>0) && !(lon_int>0))
{
    lcd.clear();

    lcd.print(F("gps not working.."));

    //msg="GPS not Working...";

    if(mode==0)

    mode=3;

    else if(mode==1)

    mode=4;

    goto down;
}

lcd.clear();

lcd.print("LAT=");

lcd.print(slat);

lcd.setCursor(0,1);

lcd.print("LON=");

lcd.print(sl原因);

down: dtostrf(beats, 0, 0, char3);

sbeat=String(char3);

dtostrf(tempC, 2, 2, char4);

stemp=String(char4);

if(mode==0)
{

```

```

    msg="HR="+sbeat+"\nTemp="+stemp+site+slat+","+slon+"&z=16";
}
else if(mode==1)
{
    msg="Soldier need help"+site+slat+","+slon+"&z=16";
}
else if(mode==3)
{
    msg="HR="+sbeat+"\nTemp="+stemp+"\ngps not working..";
}
else if(mode==4)
{
    msg=help1+"gps not working..";
}
delay(2000);
sendsms(msg,numtel);
}
*/

void checksms(String sms_number, String msg)
{
    if ((msg.startsWith(F("1")))) && (sms_number.indexOf(numtel) >= 0))
    {
        if (load1_satus == 0)
        {
            digitalWrite(load1, HIGH);

            load1_satus = 1;
        }
    }
}

```



```

else if (load1_satus == 1)
{
    digitalWrite(load1, LOW);
    load1_satus = 0;
}
// lcd.clear();
// lcd.print(F("sending status"));
// delay(1500);
// gps_track(0);
}
else if ((msg.startsWith(F("2"))) && (sms_number.indexOf(numtel) >= 0))
{
    if (load2_satus == 0)
    {
        digitalWrite(load2, HIGH);
        load2_satus = 1;
    }
    else if (load2_satus == 1)
    {
        digitalWrite(load2, LOW);
        load2_satus = 0;
    }
}
else if ((msg.startsWith(F("3"))) && (sms_number.indexOf(numtel) >= 0))
{
    if (load3_satus == 0)
    {

```

```

    digitalWrite(load3, HIGH);
    load3_satus = 1;
}
else if (load3_satus == 1)
{
    digitalWrite(load3, LOW);
    load3_satus = 0;
}
}
else if ((msg.startsWith(F("4"))) && (sms_number.indexOf(numtel) >= 0))
{
    if (load4_satus == 0)
    {
        digitalWrite(load4, HIGH);
        load4_satus = 1;
    }
    else if (load4_satus == 1)
    {
        digitalWrite(load4, LOW);
        load4_satus = 0;
    }
}

else if ((msg.startsWith(F("HEstimate"))) && (sms_number.indexOf(numtel) >= 0))
{
    Serial.println("HEstimate");
}

```

```

int index = msg.lastIndexOf('e');
int index3 = index + 3;
String MSS = msg.substring(index + 1, index3);
Serial.println("MSS: " + MSS);
int hours = MSS.toInt();
Serial.println(hours);

lcd.clear();
lcd.print(F("calculating "));
lcd.setCursor(0, 1);
lcd.print(F("estimate for "));
lcd.print(hours);
lcd.print(F("h"));
delay(2000);
estimated_uses = ((2.25 / pulse_time) * hours);
estimated_cost = (estimated_uses * x);
// Serial.println(estimated_uses);
// Serial.println(estimated_cost);
lcd.clear();
lcd.print(estimated_uses, 2);
lcd.print(F("kwh"));
lcd.setCursor(0, 1);
lcd.print(estimated_cost, 2);

String HESTIMATE_MSG = "Estimate use= " + String(estimated_uses, 2) +
"Estimate cost= " + String(estimated_cost, 2);
sendsms(HESTIMATE_MSG, numtel);

```

```

//    sms.beginSMS(numtel);
//    sms.print("Estimate use= ");
//    sms.println(estimated_uses, 2);
//    sms.print("Estimate cost= ");
//    sms.print(estimated_cost, 2);
//    sms.endSMS();

delay(100);

lcd.clear();

lcd.print(F("sms send"));

//    }
//    else
//    {
//    lcd.clear();
//    lcd.print(F("INVALID MESSAGE"));
//    }*/
}

```

```

else if ((msg.startsWith(F("DEstimate"))) && (sms_number.indexOf(numtel) >= 0))

```

```

{
    Serial.println("DEstimate");
    int index = msg.lastIndexOf('e');
    int index3 = index + 3;
    String MSS = msg.substring(index + 1, index3);
    Serial.println("MSS: " + MSS);
    int days = MSS.toInt();
    Serial.println(days);
}

```

```

// int index = indata.lastIndexOf('e');

/*for (int j = index + 1; j <= (indata - 1); j++)
{
    if (((int)indata[j] >= 48) && ((int)indata[j] <= 59))
        valid_sms = true;
    else
        valid_sms = false;
}

if ( valid_sms == true)
{
    valid_sms = false;

    int days = ((indata[index + 1] - '0') * 10) + ((indata[index + 2] - '0'));

    // delay(200);*/

lcd.clear();

lcd.print(F("calculating "));

lcd.setCursor(0, 1);

lcd.print(F("estimate for "));

lcd.print(days);

lcd.print(F("d"));

delay(2000);

estimated_uses = ((2.25 / pulse_time) * 24 * days);

estimated_cost = (estimated_uses * x);

//Serial.println(estimated_uses);

//Serial.println(estimated_cost);

lcd.clear();

lcd.print(estimated_uses);

lcd.print(F("kwh"));

```

```

    lcd.setCursor(0, 1);

    lcd.print(estimated_cost, 2);

    delay(2000);

    String DESTIMATE_MSG = "Estimate use= " + String(estimated_uses, 2) +
    "Estimate cost= " + String(estimated_cost, 2);

    sendsms(DESTIMATE_MSG, numtel);

    //    sms.beginSMS(numtel);

    //    sms.print("Estimate use= ");

    //    sms.println(estimated_uses, 2);

    //    sms.print("Estimate cost= ");

    //    sms.print(estimated_cost, 2);

    //    sms.endSMS();

    delay(100);

    lcd.clear();

    lcd.print(F("sms send"));

    delay(2000);

}

else

{

    lcd.clear();

    lcd.print(F("INVALID MESSAGE"));

}

}

```

12. BIBLIOGRAPHY

TEXT BOOKS REFERED

1. ATMEGA 328 Data Sheets.

WEBSITES

- www.atmel.com
- www.beyondlogic.org
- www.wikipedia.org
- www.howstuffworks.com
- www.alldatasheets.com

