**Phase Angle Measurement Circuit Or A Power Factor Measurement Circuit.**

**Roshan S. Jadhav**

**22220205**

**311066**

**ABSTRACT**

The Phase Angle Measurement Circuit and Power Factor Measurement Circuit are essential components in electrical engineering for analyzing the efficiency and quality of power systems. These circuits enable precise measurement of the phase angle between voltage and current waveforms, as well as the power factor, which are critical parameters for assessing the performance of electrical loads. By accurately quantifying the phase relationship between voltage and current, these circuits provide valuable insights into the reactive power consumption and overall efficiency of electrical equipment. Their applications range from industrial power systems to renewable energy installations, where optimizing power factor and phase angle is essential for minimizing energy losses and maximizing system reliability. Through their compact design and precise measurement capabilities, these circuits play a vital role in ensuring the efficient operation of electrical networks in various sectors.

Top of Form

.

**INTRODUCTION**

The Phase Angle Measurement Circuit and Power Factor Measurement Circuit are fundamental components in electrical engineering, crucial for assessing the performance and efficiency of power systems. These circuits play a pivotal role in analyzing the phase relationship between voltage and current waveforms, as well as quantifying the power factor of electrical loads. By providing accurate measurements of phase angle and power factor, these circuits enable engineers to optimize energy usage, improve system stability, and reduce wastage in diverse applications. Their importance spans across industries, including manufacturing, utilities, and renewable energy, where precise control of power factor is essential for enhancing overall system efficiency and reliability. This introduction highlights the significance of Phase Angle Measurement and Power Factor Measurement Circuits in modern electrical engineering practices.

**Embedded** **Introduction:**

An embedded system is one kind of a computer system mainly designed to perform several tasks like to access, process, and store and also control the data in various electronics-based systems. Embedded systems are a combination of hardware and software where software is usually known as firmware that is embedded into the hardware. One of its most important characteristics of these systems is, it gives the o/p within the time limits. Embedded systems support to make the work more perfect and convenient. So, we frequently use embedded systems in simple and complex devices too. The applications of embedded systems mainly involve in our real life for several devices like microwave, calculators, TV remote control, home security and neighbourhood traffic control systems, etc.

User interface

Embedded system

Software

Hardware

Inputs

Output

Link to other systems

Fig: Overview of embedded system

**Embedded system:**

Embedded system includes mainly two sections, they are

1. Hardware

2. Software

Input devices interfacing

and driver circuits

Memory

Timers

Serial communication ports

Processor

Power supply and oscillator circuits

Application specific circuits

Parallel ports

Interrupt controller

Output devices interfacing

Block diagram of embedded system

**Embedded System Hardware:**

As with any electronic system, an embedded system requires a hardware platform on which it performs the operation. Embedded system hardware is built with a microprocessor or microcontroller. The embedded system hardware has elements like input output (I/O) interfaces, user interface, memory and the display. Usually, an embedded system consists of:

* Power Supply
* Processor
* Memory
* Timers
* Serial communication ports
* Output/Output circuits
* System application specific circuits

Embedded systems use different processors for its desired operation. Some of the processors used are

1. Microprocessor

2. Microcontroller

3. Digital signal processor

**Microprocessor vs. Microcontroller**

**Microprocessor**

* **CPU** on a chip.
* We can attach required amount of ROM, RAM and I/O ports.
* Expensive due to external peripherals.
* Large in size
* general-purpose

**Microcontroller**

* **Computer** on a chip
* fixed amount of on-chip ROM, RAM, I/O ports
* Low cost.
* Compact in size.
* Specific –purpose

**Embedded System Software:**

The embedded system software is written to perform a specific function. It is typically written in a high level format and then compiled down to provide code that can be lodged within a non-volatile memory within the hardware. An embedded system software is designed to keep in view of the three limits:

* Availability of system memory
* Availability of processor’s speed
* When the system runs continuously, there is a need to limit power dissipation for events like stop, run and wake up.

**Bringing software and hardware together for embedded system:**

To make software to work with embedded systems we need to bring software and hardware together .for this purpose we need to burn our source code into microprocessor or microcontroller which is a hardware component and which takes care of all operations to be done by embedded system according to our code.

Generally we write source codes for embedded systems in assembly language, but the processors run only executable files.The process of converting the source code representation of your embedded software into an executable binary image involves three distinct steps:

1. Each of the source files must be compiled or assembled into an object file.
2. All of the object files that result from the first step must be linked together to produce a single object file, called the re-locatable program.
3. Physical memory addresses must be assigned to the relative offsets within the re-locatable program in a process called relocation.

The result of the final step is a file containing an executable binary image that is ready to run on the embedded system.

Source code

Linker

Locator

Assembler

Executable file

Processor

Flow of burning source code to processor

**Applications:**

Embedded systems have different applications. A few select [applications of embedded systems](https://www.elprocus.com/embedded-systems-real-time-applications/) are smart cards, telecommunications, satellites, missiles, digital consumer electronics, computer networking, etc.

[Embedded Systems in Automobiles](http://www.edgefx.in/importance-of-embedded-systems-in-automobiles-with-applications/)

* Motor Control System
* Engine or Body Safety
* [Robotics](http://www.edgefx.in/top-list-robotics-projects-for-engineering-beginners/) in Assembly Line
* Mobile and E-Com Access

Embedded systems in Telecommunications

* Mobile computing
* Networking
* [Wireless Communications](http://www.edgefx.in/multiple-input-and-multiple-output-mimo-wireless-communications/)

Embedded Systems in Smart Cards

* Banking
* Telephone
* [Security Systems](http://www.edgefx.in/microcontroller-based-projects-on-car-security-systems-using-gsm/)

**Implementation flow:**

**Stage 1:**

Considering the problems of existing methods and giving solution to that problem by considering the basic requirements for our proposed system

**Stage 2:**

Considering the hardware requirement for the proposed system

For this we need to select the below components:

1. Microcontroller

2. Inputs for the proposed system (ex: sensors, drivers etc.,)

3. Outputs (ex: relays, loads)

**Stage 3:**

After considering hardware requirements, now we need to check out the software requirements. Based on the microcontroller we select there exists different software for coding, compiling, debugging. we need to write source code for that proposed system based on our requirements and compile, debug the code in that software.

After completing all the requirements of software and hardware we need to bring both together to work our system. For this we need to burn our source code into microcontroller, after burning our source code to microcontroller then connect all input and output modules as per our requirement.

**LITERATURE REVIEW**

* In "Phase Angle Measurement Techniques: A Review" by Sharma et al. (2019), the authors discuss various methods and techniques used for phase angle measurement in power systems, including analog and digital approaches.
* "A Review on Phase Angle Measurement and Its Applications" by Gupta et al. (2020) provides an overview of the importance of phase angle measurement in power quality analysis and control, highlighting its applications in renewable energy systems and smart grids.
* Zhang et al. (2018) explore the advancements in phase angle measurement technology and its role in grid synchronization and stability analysis in their paper "Recent Advances in Phase Angle Measurement Techniques for Power Systems".
* "Comparison of Phase Angle Measurement Techniques for Power Systems" by Li and Wang (2017) compares different phase angle measurement techniques, including voltage and current synchronization methods, phase-locked loop (PLL) techniques, and digital signal processing (DSP) algorithms.
* Singh et al. (2019) review the advancements in power factor measurement techniques and their applications in energy management systems in their paper "Recent Trends in Power Factor Measurement Techniques".
* "A Review on Power Factor Measurement Techniques and Their Applications" by Das et al. (2020) provides insights into the various methods used for power factor measurement, including analog and digital approaches, and discusses their significance in power quality assessment.
* Kumar et al. (2018) discuss the challenges and opportunities in power factor measurement in their paper "Power Factor Measurement Techniques: Challenges and Future Trends".
* "Power Factor Measurement: A Review" by Mishra and Mishra (2016) offers a comprehensive review of power factor measurement techniques, emphasizing their importance in energy efficiency and power quality improvement initiatives.

**EXISTING METHOD**

Phase Angle Measurement Circuit: Utilizes analog or digital techniques such as phase shifters, zero crossing detection, or digital signal processing to measure the phase difference between voltage and current signals. Arduino-based solutions offer cost-effective and versatile options for phase angle measurement.

Power Factor Measurement Circuit: Implements methods like the wattmeter technique, reactive power compensation, digital power analyzers, or harmonic analysis to assess power factor. These circuits enable accurate evaluation of power factor, essential for optimizing energy efficiency and system stability in electrical networks.

**Drawbacks:**

* Manual references are required

**PROPOSED METHOD**

Arduino UNO is a versatile microcontroller board widely used in various electronic projects due to its ease of use, flexibility, and robust performance. Equipped with an Atmega328 microcontroller, Arduino UNO offers a wide range of input and output pins, analog and digital interfaces, and built-in communication protocols. Its simple programming environment, based on the Arduino IDE, allows users to quickly develop and deploy code for controlling sensors, actuators, and other peripherals. Arduino UNO's compatibility with a vast array of sensors, shields, and modules further enhances its versatility, making it suitable for prototyping, experimentation, and educational purposes. With its low cost and widespread availability, Arduino UNO serves as an accessible platform for electronics enthusiasts, hobbyists, students, and professionals alike, empowering them to bring their creative ideas to life and explore the world of embedded systems and IoT applications.

**BLOCK DIAGRAM**

Arduino

LCD

Power supply

Pzeo004t

Relay

Bulb

**Fig: block diagram of proposed method**

**ADVANTAGES AND APPLICATIONS**

**ADVANTAGES**

* Real time monitoring
* Remote Access
* Periodical collection of data

**APPLICATIONS**

* Energy production
* Substations

**HARDWARE AND SOFTWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS**

**ARDUINO**

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. This guide is for students in ME 2011, or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users, prowl the web; there are lots of resources.

This is what the Arduino board looks like.



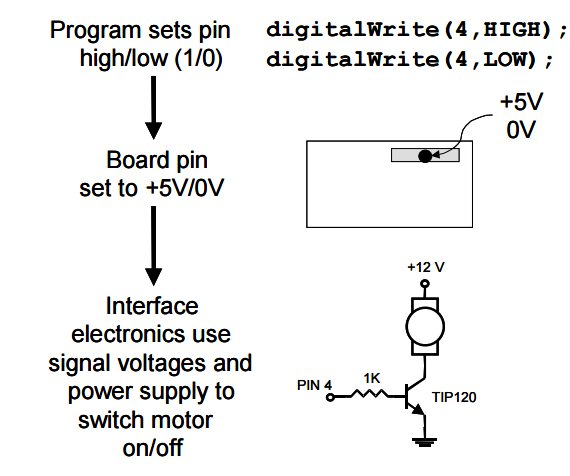
The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions.

**Arduino Hardware**

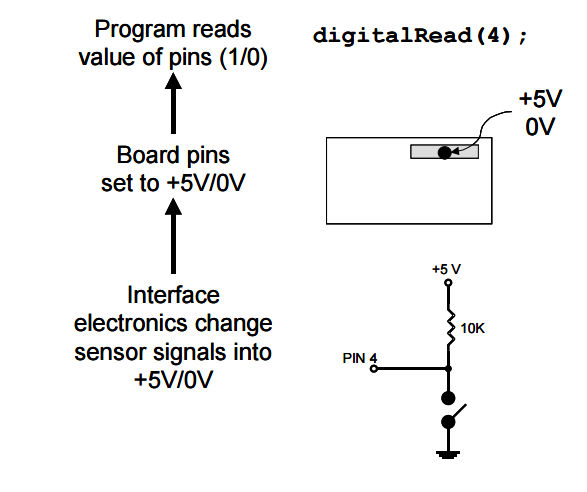
The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/O pins labeled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches.

Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices, but does mean that interface circuits are needed to control devices other than simple LED's. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the pin drive an interface circuit that in turn drives the motor. A later section of this document shows how to interface to a small motor.

To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off. The sequence of events is shown in this figure.



To determine the state of switches and other sensors, the Arduino is able to read the voltage value applied to its pins as a binary number. The interface circuitry translates the sensor signal into a 0 or +5 V signal applied to the digital I/O pin. Through a program command, the Ardiomp interrogates the state of the pin. If the pin is at 0 V, the program will read it as a 0 or LOW. If it is at +5 V, the program will read it as a 1 or HIGH. If more than +5 V is applied, you may blow out your board, so be careful. The sequence of events to read a pin is shown in this figure.



Interacting with the world has two sides. First, the designer must create electronic interface circuits that allow motors and other devices to be controlled by a low (1-10 mA) current signal that switches between 0 and 5 V, and other circuits that convert sensor readings into a switched 0 or 5 V signal. Second, the designer must write a program using the set of Arduino commands that set and read the I/O pins. Examples of both can be found in the Arduino resources section of the ME2011 web site.

**Atmega328p features:**

* High Performance, Low Power AVR® 8-Bit Microcontroller
* Advanced RISC Architecture

– 131 Powerful Instructions

– Most Single Clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 20 MIPS Throughput at 20 MHz

– On-chip 2-cycle Multiplier

* High Endurance Non-volatile Memory Segments

– 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)

– 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)

– 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C(1)

– Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation

– Programming Lock for Software Security

* Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Six PWM Channels – 8-channel 10-bit ADC in TQFP and QFN/MLF package Temperature Measurement – 6-channel 10-bit ADC in PDIP Package Temperature Measurement

– Programmable Serial USART

 – Master/Slave SPI Serial Interface

– Byte-oriented 2-wire Serial Interface (Philips I2 C compatible)

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

– Interrupt and Wake-up on Pin Change

* Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

* I/O and Packages

– 23 Programmable I/O Lines

– 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

* Operating Voltage:

– 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P

* Temperature Range:

– -40°C to 85°C

* Speed Grade:

– 0 - 20 MHz @ 1.8 - 5.5V

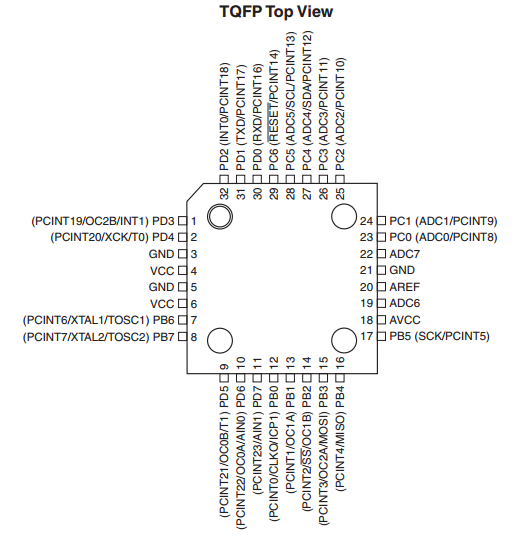
* Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:

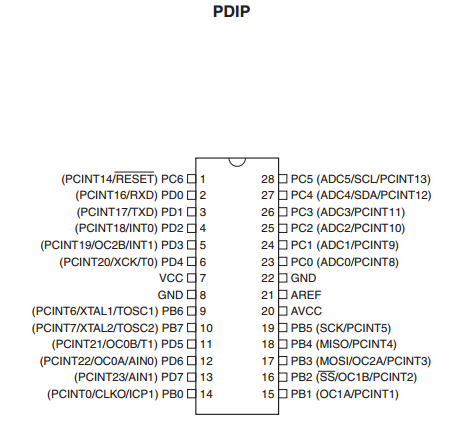
– Active Mode: 0.2 mA

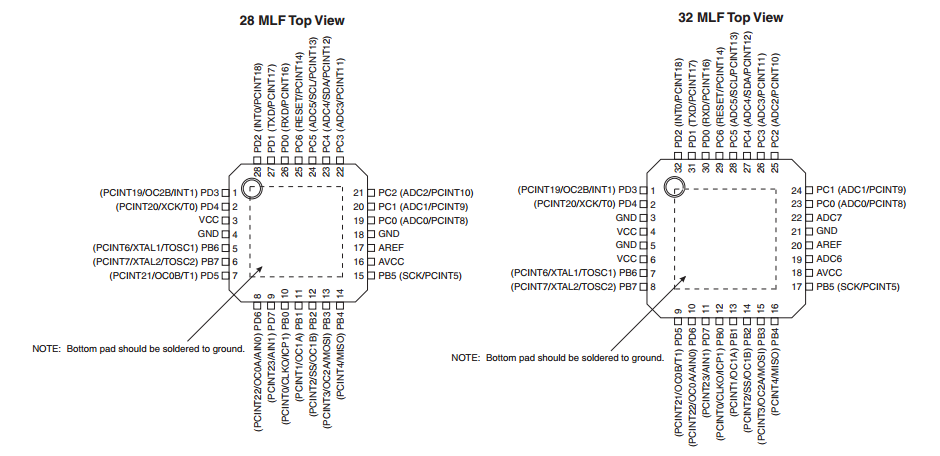
– Power-down Mode: 0.1 µA

– Power-save Mode: 0.75 µA (Including 32 kHz RTC)

**PIN CONFIGURATIONS**







**Pin Descriptions**

**VCC:**  Digital supply voltage.

**GND:** Ground.

**Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2:** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in ”Alternate Functions of Port B” on page 76 and ”System Clock and Clock Options” on page 26.

**Port C (PC5:0):** Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**PC6/RESET:** If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-3 on page 308. Shorter pulses are not guaranteed to generate a Reset. The various special features of Port C are elaborated in ”Alternate Functions of Port C” on page 79.

**Port D (PD7:0):** Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. The various special features of Port D are elaborated in “Alternate Functions of Port D” on page 82.

**AVCC:** AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

**AREF:** AREF is the analog reference pin for the A/D Converter

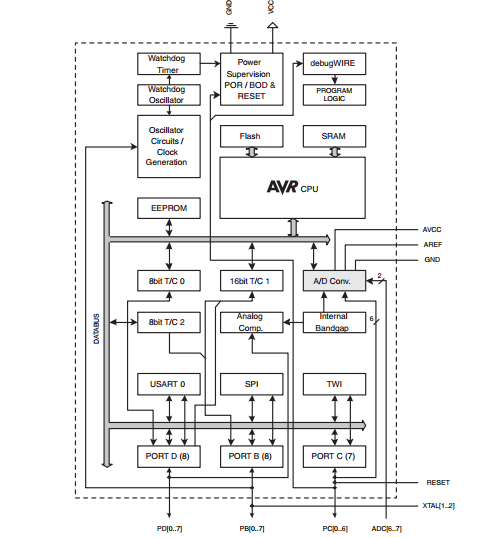
**ADC7:6 (TQFP and QFN/MLF Package Only):** In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

**OVERVIEW**

The ATmega48PA/88PA/168PA/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48PA/88PA/168PA/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

**BLOCK DIAGRAM**

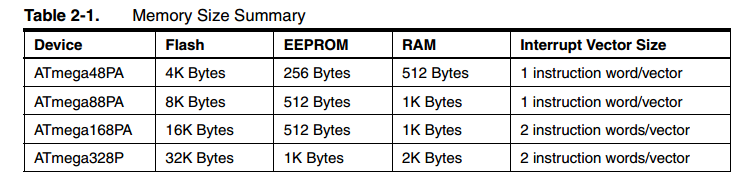
****

The ATmega48PA/88PA/168PA/328P provides the following features: 4/8/16/32K bytes of In System Programmable Flash with Read-While-Write capabilities, 256/512/512/1K bytes EEPROM, 512/1K/1K/2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel’s high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega48PA/88PA/168PA/328P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications. The ATmega48PA/88PA/168PA/328P AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

**Comparison Between ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P**

The ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P differ only in memory sizes, boot loader support, and interrupt vector sizes. Table 2-1 summarizes the different memory and interrupt vector sizes for the three devices.

****

ATmega88PA, ATmega168PA and ATmega328P support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega48PA, there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

**POWER:**

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The power pins are as follows:

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

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

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

**∙ GND.** Ground pins.

**Memory:**

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

**Input and Output**

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

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

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

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

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

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

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

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

There are a couple of other pins on the board:

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

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

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

**Communication**

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

**Programming**

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

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

∙ On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

**Automatic (Software) Reset**

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment.

This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following halfsecond or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

**USB Overcurrent Protection**

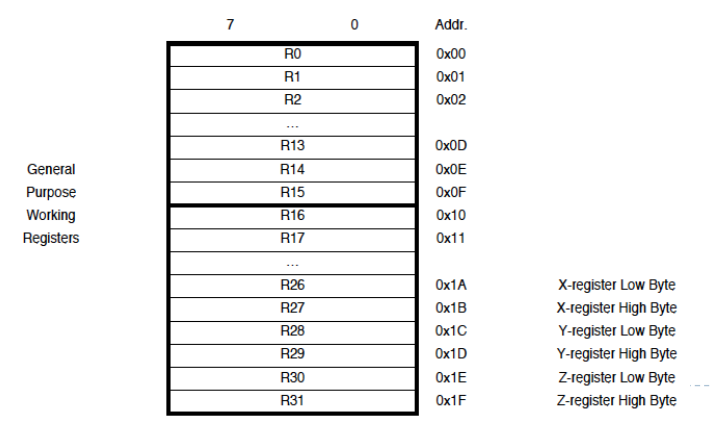
The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break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

**Physical Characteristics**

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

**Register File**

* 32 8-bit GP registers
* Part of SRAM memory space

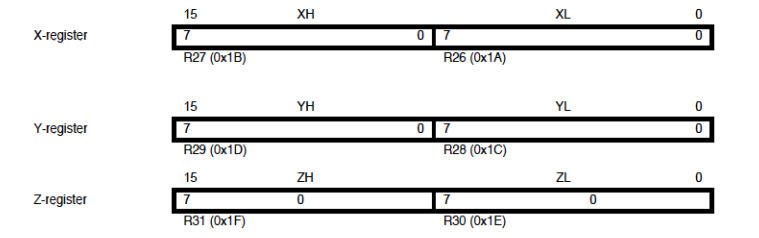
****

**Special Addressing Registers**

* X, Y and Z registers

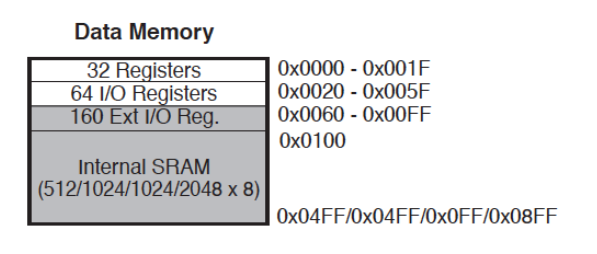
16-bit registers made using registers 26 – 31

* Support indirect addressing

****

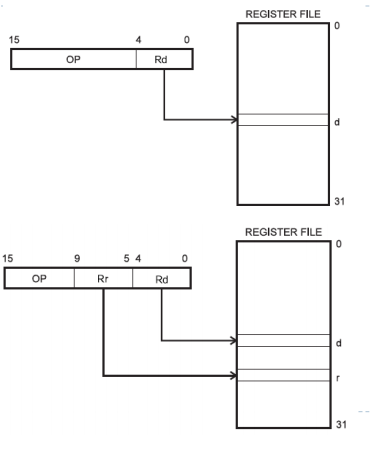
**AVR Memory**

* Program memory – Flash
* Data memory – SRAM

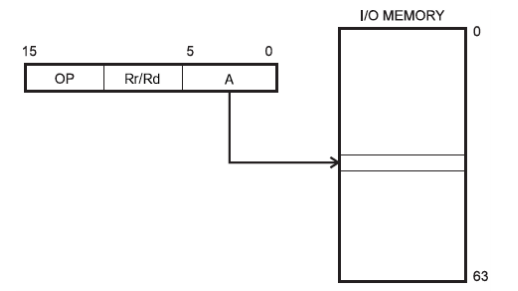


**Addressing Modes**

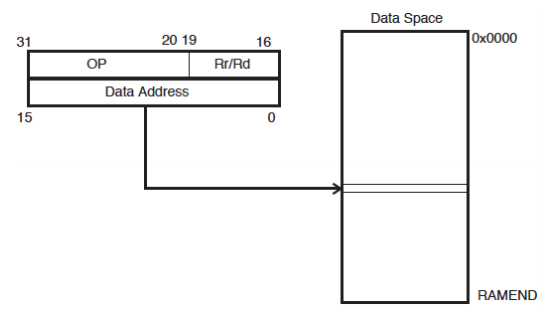
* Direct register addressing

****

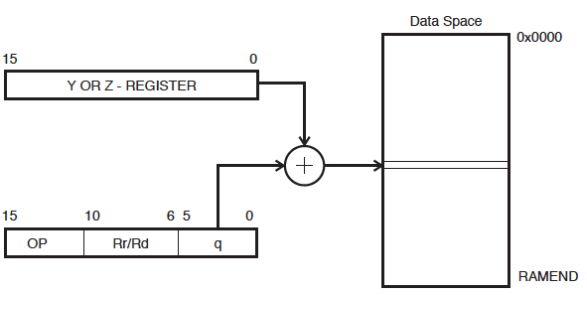
* Direct I/O addressing

****

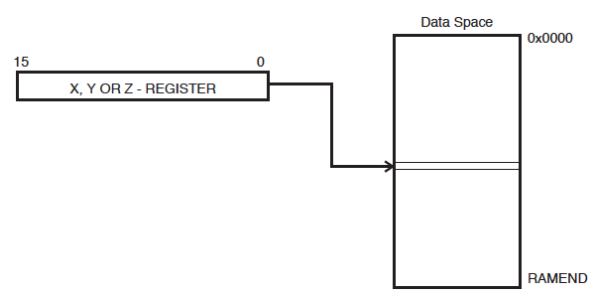
Direct data memory addressing

****

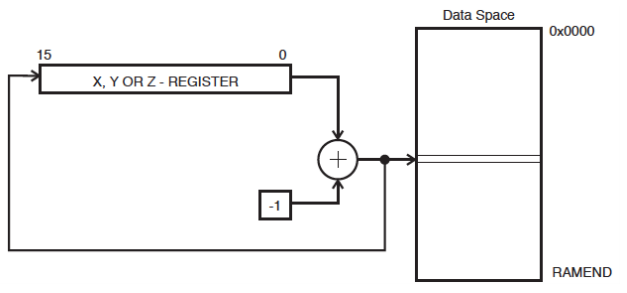
Direct data memory with displacement addressing

****

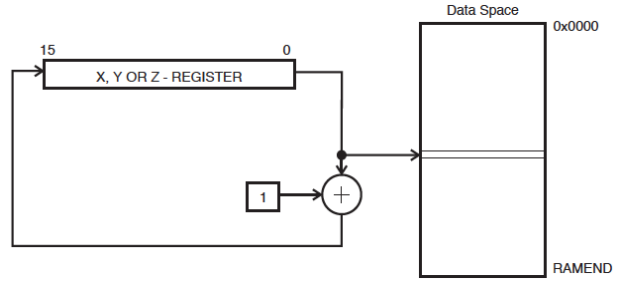
Indirect data memory addressing

****

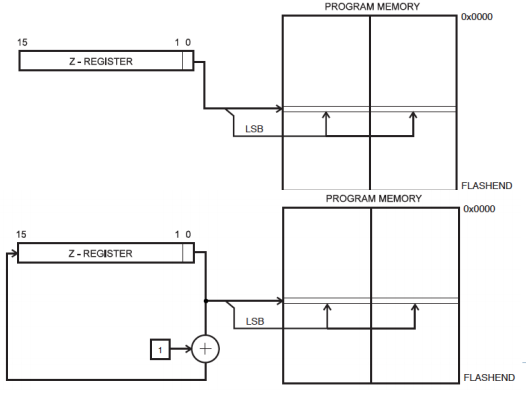
Indirect data memory addressing with pre-decrement

****

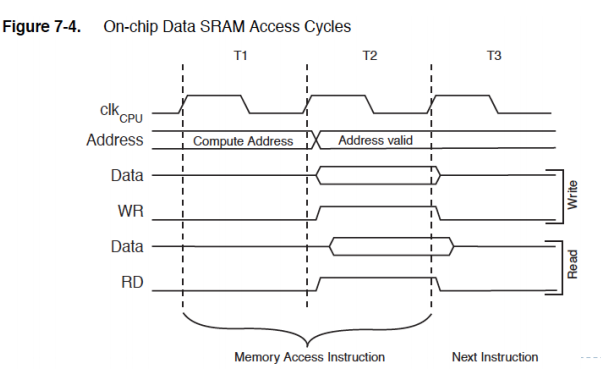
Indirect data memory addressing with post-increment

****

Program memory addressing (constant data)

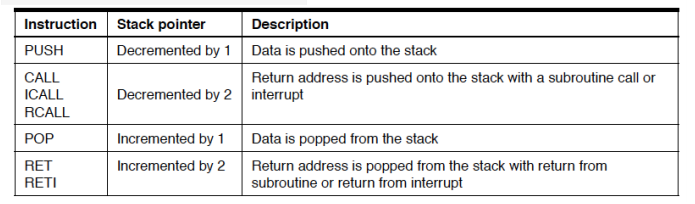
****

SRAM Read/Write Timing

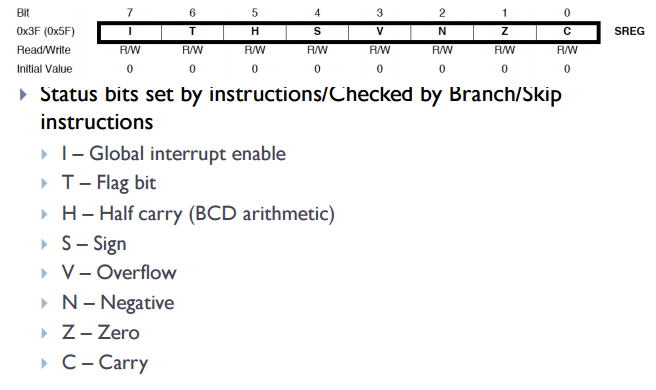
****

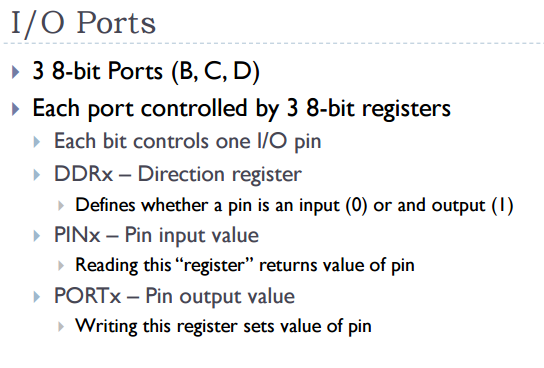
**Stack Pointer Register**

* Special register in I/O space [3E, 3D]
* Enough bits to address data space
* Initialized to RAMEND (address of highest memory address)
* Instructions that use the stack pointer

****

**Program Status Register (PSR)**

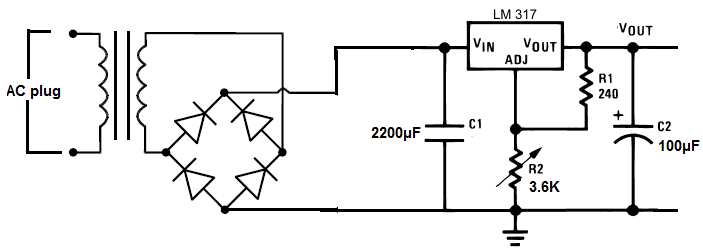
****

****

**Power supply:**

A power supply is a component that provides at least one electrical charge with power. It typically converts one type of electrical power to another, but it can also convert a different Energy form in electrical energy, such as solar, mechanical, or chemical.

A power supply provides electrical power to components. Usually the term refers to devices built into the powered component. Computer power supplies, for example, convert AC current to DC current and are generally located along with at least one fan at the back of the computer case.

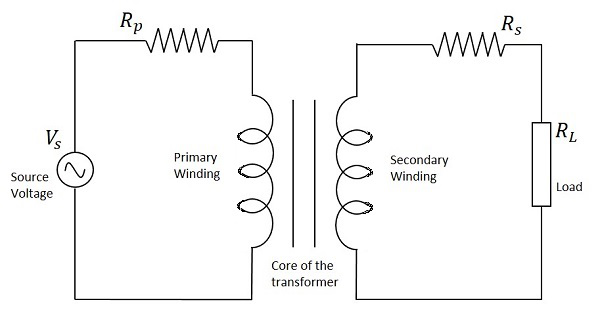
Most computer power supplies also have an input voltage switch that, depending on the geographic location, can be set to 110v/115v or 220v/240v. Due to the different power voltages supplied by power outlets in different countries, this switch position is crucial.

Some basic components used in the supply of power:

**Transformer:**

A transformer is a static electrical gadget that exchanges control between at least two circuits. A fluctuating current creates a changing attractive motion in one transformer curl, which thus actuates a differing electromotive power over a second loop twisted around a similar center.

Without a metallic association between the two circuits, electrical vitality can be exchanged between the two loops. The enlistment law of Faraday found in 1831 portrayed the impact of prompted voltage in any curl because of the changing attractive flux surrounded by the coil.



**Circuit of transformer**

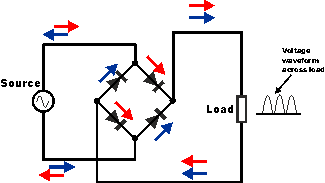
****

**Transformer**

**Rectifier:**

A **rectifier** is an electrical device that [converts](https://en.wikipedia.org/wiki/Electric_power_conversion) [alternating current](https://en.wikipedia.org/wiki/Alternating_current) (AC), which periodically reverses direction, to [direct current](https://en.wikipedia.org/wiki/Direct_current) (DC), which flows in only one direction.  The process is known as *rectification*, since it "straightens" the direction of current.

Rectifiers have many uses, but are often found to serve as components of DC power supplies and direct power transmission systems with high voltage. Rectification can be used in roles other than direct current generation for use as a power source.

****

**Circuit of rectifier**

****

**Rectifier**

**Capacitors:**

Capacitors are used to attain from the connector the immaculate and smoothest DC voltage in which the rectifier is used to obtain throbbing DC voltage which is used as part of the light of the present identity. Capacitors are used to acquire square DC from the current AC experience of the current channels so that they can be used as a touch of parallel yield.

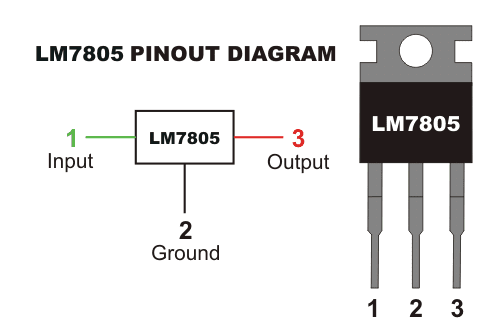


**Capacitor**

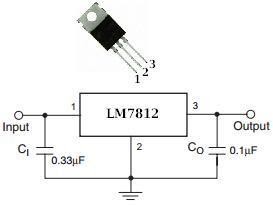
**Voltage regulators:**

The 78XX voltage controller is mainly used for voltage controllers as a whole. The XX speaks to the voltage delivered to the specific gadget by the voltage controller as the yield. 7805 will supply and control 5v yield voltage and 12v yield voltage will be created by 7812.

The voltage controllers are that their yield voltage as information requires no less than 2 volts. For example, 7805 as sources of information will require no less than 7V, and 7812, no less than 14 volts. This voltage is called Dropout Voltage, which should be given to voltage controllers.



**7805 voltage regulator with pinout**



**7812 voltage regulator with pinout**

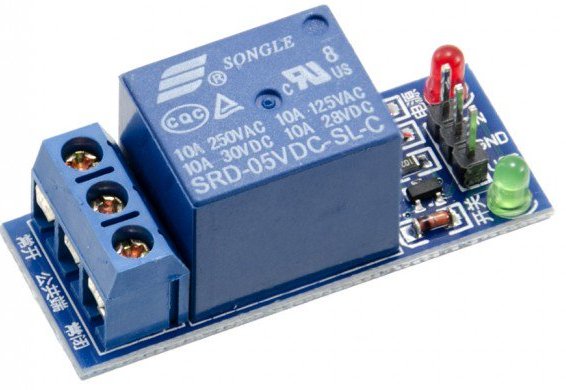
to set the bandwidth. Pin-7 is the analog output pin. Pin-8 is the power supply pin.

**Relay:**

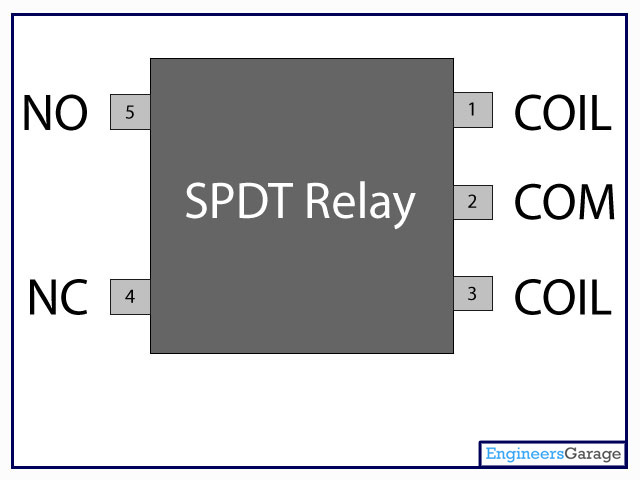
**What is a relay?**

A relay is an electromagnetic switch that is used to turn on and turn off a circuit by a low power signal, or where several circuits must be controlled by one signal.

Most of the high end industrial application devices have relays for their effective working. Relays are simple switches which are operated both electrically and mechanically. Relays consist of an electromagnet and also a set of contacts. The switching mechanism is carried out with the help of the electromagnet. There are also other operating principles for its working. But they differ according to their applications. Most of the devices have the application of relays.



### Pin Diagram:



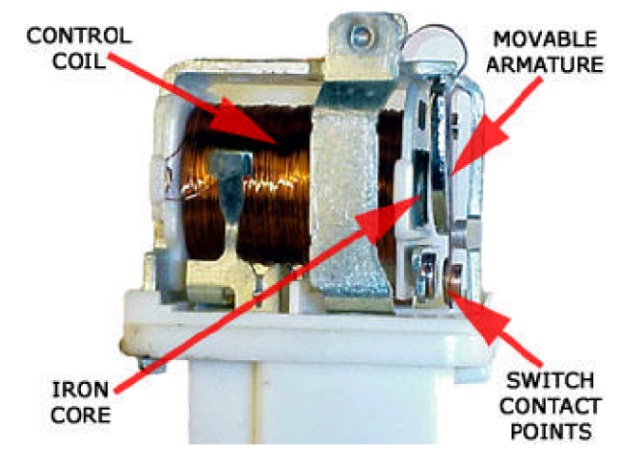
**Why is a relay used?**

The main operation of a relay comes in places where only a low-power signal can be used to control a circuit. It is also used in places where only one signal can be used to control a lot of circuits. The application of relays started during the invention of telephones. They played an important role in switching calls in telephone exchanges. They were also used in long distance telegraphy. They were used to switch the signal coming from one source to another destination. After the invention of computers they were also used to perform Boolean and other logical operations. The high end applications of relays require high power to be driven by electric motors and so on. Such relays are called contactors.

### Relay Design

* There are only four main parts in a relay. They are
* Electromagnet
* Movable Armature
* Switch point contacts
* Spring

The figures given below show the actual design of a simple relay.



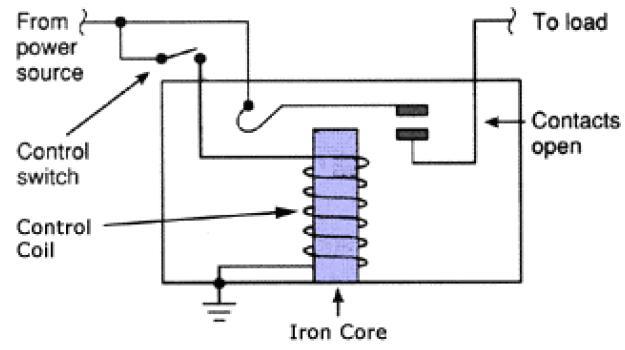
**Relay Construction**

It is an electro-magnetic relay with a wire coil, surrounded by an iron core. A path of very low reluctance for the magnetic flux is provided for the movable armature and also the switch point contacts.

The movable armature is connected to the yoke which is mechanically connected to the switch point contacts. These parts are safely held with the help of a spring. The spring is used so as to produce an air gap in the circuit when the relay becomes de-energized.

### How relay works?

The relay function can be better understood by explaining the following diagram given below.



**Relay Design**

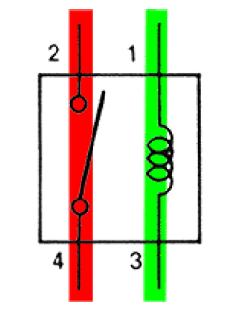
The diagram shows an inner section diagram of a relay. An iron core is surrounded by a control coil. As shown, the power source is given to the electromagnet through a control switch and through contacts to the load. When current starts flowing through the control coil, the electromagnet starts energizing and thus intensifies the magnetic field. Thus the upper contact arm starts to be attracted to the lower fixed arm and thus closes the contacts causing a short circuit for the power to the load. On the other hand, if the relay was already de-energized when the contacts were closed, then the contact move oppositely and make an open circuit.

As soon as the coil current is off, the movable armature will be returned by a force back to its initial position. This force will be almost equal to half the strength of the magnetic force. This force is mainly provided by two factors. They are the spring and also gravity.

Relays are mainly made for two basic operations. One is low voltage application and the other is high voltage. For low voltage applications, more preference will be given to reduce the noise of the whole circuit. For high voltage applications, they are mainly designed to reduce a phenomenon called arcing.

### Relay Basics

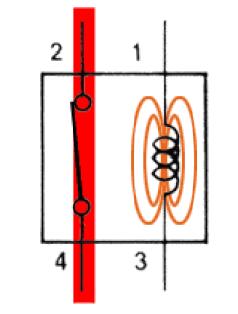
The basics for all the relays are the same. Take a look at a 4 pin relay shown below. There are two colors shown. The green color represents the control circuit and the red color represents the load circuit. A small control coil is connected onto the control circuit. A switch is connected to the load. This switch is controlled by the coil in the control circuit. Now let us take the different steps that occur in a relay.



**Relay operation**

* **Energized Relay (ON)**

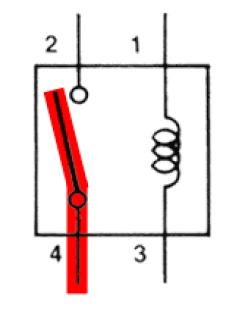
As shown in the circuit, the current flowing through the coils represented by pins 1 and 3 causes a magnetic field to be aroused. This magnetic field causes the closing of the pins 2 and 4. Thus the switch plays an important role in the relay working. As it is a part of the load circuit, it is used to control an electrical circuit that is connected to it. Thus, when the electrical relay in energized the current flow will be through the pins 2 and 4.



**Energized Relay (ON)**

* **De – Energized Relay (OFF)**

As soon as the current flow stops through pins 1 and 3, the relay switch opens and thus the open circuit prevents the current flow through pins 2 and 4. Thus the relay becomes de-energized and thus in off position.



**De-Energized Relay (OFF)**

**In simple, when a voltage is applied to pin 1, the electromagnet activates, causing a magnetic field to be developed, which goes on to close the pins 2 and 4 causing a closed circuit. When there is no voltage on pin 1, there will be no electromagnetic force and thus no magnetic field. Thus the switches remain open.**

### Pole and Throw

Relays have the exact working of a switch. So, the same concept is also applied. A relay is said to switch one or more poles. Each pole has contacts that can be thrown in mainly three ways. They are

* **Normally Open Contact (NO):**  NO contact is also called a make contact. It closes the circuit when the relay is activated. It disconnects the circuit when the relay is inactive.
* **Normally Closed Contact (NC):**  NC contact is also known as break contact. This is opposite to the NO contact. When the relay is activated, the circuit disconnects. When the relay is deactivated, the circuit connects.
* **Change-over (CO) / Double-throw (DT) Contacts:**  This type of contacts are used to control two types of circuits. They are used to control a NO contact and also a NC contact with a common terminal. According to their type they are called by the names **break before make** and **make before break** contacts.

Relays can be used to control several circuits by just one signal. A relay switches one or more poles, each of whose contacts can be thrown by energizing the coil.

Relays are also named with designations like

* **Single Pole Single Throw (SPST)**: The SPST relay has a total of four terminals. Out of these two terminals can be connected or disconnected. The other two terminals are needed for the coil to be connected.
* **Single Pole Double Throw (SPDT):** The SPDT relay has a total of five terminals. Out of these two are the coil terminals. A common terminal is also included which connects to either of two others.
* **Double Pole Single Throw (DPST):**  The DPST relay has a total of six terminals. These terminals are further divided into two pairs. Thus they can act as two SPST which are actuated by a single coil. Out of the six terminals two of them are coil terminals.
* **Double Pole Double Throw (DPDT)**: The DPDT relay is the biggest of all. It has mainly eight relay terminals. Out of these two rows are designed to be change over terminals. They are designed to act as two SPDT relays which are actuated by a single coil.

### Relay Applications

* A relay circuit is used to realize logic functions. They play a very important role in providing safety critical logic.
* Relays are used to provide time delay functions. They are used to time the delay open and delay close of contacts.
* Relays are used to control high voltage circuits with the help of low voltage signals. Similarly they are used to control high current circuits with the help of low current signals.
* They are also used as protective relays. By this function all the faults during transmission and reception can be detected and isolated.

#### **Application of Overload Relay**

Overload relay is an electro-mechanical device that is used to safeguard motors from overloads and power failures. Overload relays are installed in motors to safeguard against sudden current spikes that may damage the motor. An overload relay switch works in characteristics with current over time and is different from circuit breakers and fuses, where a sudden trip is made to turn off the motor. The most widely used overload relay is the thermal overload relay where a bimetallic strip is used to turn off the motor. This strip is set to make contact with a contactor by bending itself with rising temperatures due to excess current flow. The contact between the strip and the contactor causes the contactor to de-energize and restricts the power to the motor, and thus turns it off.

Another type of overload motor is the electronic type which continuously watches the motor current, whereas the thermal overload relay shuts off the motor depending on the rise of temperature/heat of the strip.

All overload relays available to buy comes in different specifications, the most important of them being the current ranges and response time. Most of them are designed to automatically reset to work after the motor is turned back on.

### Relay Selection

You must note some factors while selecting a particular relay. They are

* Protection Different protections like contact protection and coil protection must be noted. Contact protection helps in reducing arcing in circuits using inductors. Â Coil protection helps in reducing surge voltage produced during switching.
* Look for a standard relay with all regulatory approvals.
* Switching time Ask for high speed switching relays if you want one.
* Ratings There are current as well as voltage ratings. The current ratings vary from a few amperes to about 3000 amperes. Â In case of voltage ratings, they vary from 300 Volt AC to 600 Volt AC. There are also high voltage relays of about 15,000 Volts.
* Type of contact used whether it is a NC or NO or closed contact.
* Select Make before Break or Break before Make contacts wisely.
* Isolation between coil circuit and contacts

**LCD:**

LCD (Liquid Crystal Display) is the innovation utilized in scratch pad shows and other littler PCs. Like innovation for light-producing diode (LED) and gas-plasma, LCDs permit presentations to be a lot more slender than innovation for cathode beam tube (CRT). LCDs expend considerably less power than LED shows and gas shows since they work as opposed to emanating it on the guideline of blocking light.

A LCD is either made with a uninvolved lattice or a showcase network for dynamic framework show. Likewise alluded to as a meager film transistor (TFT) show is the dynamic framework LCD. The uninvolved LCD lattice has a matrix of conductors at every crossing point of the network with pixels. Two conductors on the lattice send a current to control the light for any pixel. A functioning framework has a transistor situated at every pixel crossing point, requiring less current to control the luminance of a pixel.

Some aloof network LCD's have double filtering, which implies they examine the matrix twice with current in the meantime as the first innovation took one sweep. Dynamic lattice, be that as it may, is as yet a higher innovation.

A 16x2 LCD show is an essential module that is generally utilized in various gadgets and circuits. These modules more than seven sections and other multi fragment LEDs are liked. The reasons being: LCDs are affordable; effectively programmable; have no restriction of showing exceptional and even custom characters (not at all like in seven fragments), movements, etc.

A 16x2 LCD implies 16 characters can be shown per line and 2 such lines exist. Each character is shown in a lattice of 5x7 pixels in this LCD. There are two registers in this LCD, in particular Command and Data.

The directions given to the LCD are put away by the order register. An order is a direction given to LCD to play out a predefined assignment, for example, introducing it, clearing its screen, setting the situation of the cursor, controlling presentation, and so forth. The information register will store the information that will be shown on the LCD. The information is the character's ASCII incentive to show on the LCD.

**Data/Signals/Execution of LCD**

Now that was all about the signals and the hardware. Let us come to data, signals and execution.

Two types of signals are accepted by LCD, one is data and one is control. The LCD module recognizes these signals from the RS pin status. By pulling the R / W pin high, data can now also be read from the LCD display. Once the E pin has been pulsed, the LCD display reads and executes data at the falling edge of the pulse, the same for the transmission case.

It takes 39-43μS for the LCD display to place a character or execute a command. It takes 1.53ms to 1.64ms except for clearing display and searching for cursor to the home position.

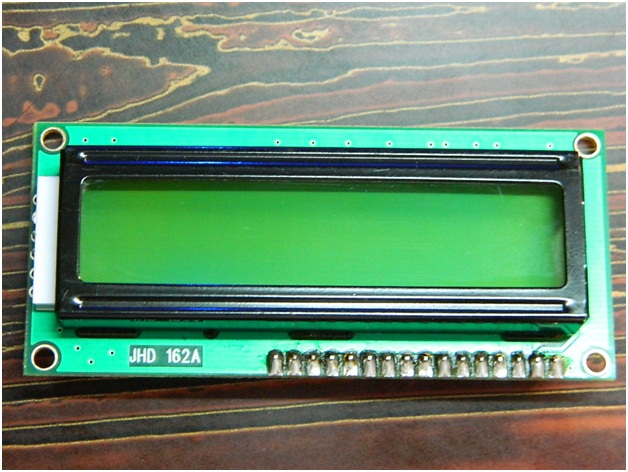
Any attempt to send data before this interval may result in failure in some devices to read data or execute the current data. Some devices compensate for the speed by storing some temporary registers with incoming data.

There are two RAMs for LCD displays, namely DDRAM and CGRAM. DDRAM registers the position in which the character would be displayed in the ASCII chart. Each DDRAM byte represents every single position on the display of the LCD.

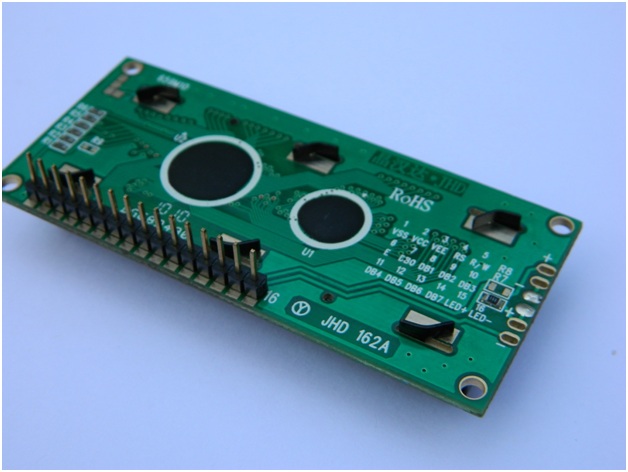
The DDRAM information is read by the LCD controller and displayed on the LCD screen. CGRAM enables users to define their personalized characters. Address space is reserved for users for the first 16 ASCII characters.

Users can easily display their custom characters on the LCD screen after CGRAM has been set up to display characters.

**Images of LCD Display:-**

[](http://www.circuitstoday.com/wp-content/uploads/2012/02/LCD-Display-Front-Side.jpg)

**LCD – Front View**

[](http://www.circuitstoday.com/wp-content/uploads/2012/02/lcd-display-back-side.jpg)

**LCD – Back View**

**Pin Diagram:**



**Pin Description:**

|  |  |  |
| --- | --- | --- |
| Pin No | Function | Name |
| 1 | Ground (0V) | Ground |
| 2 | Supply voltage; 5V (4.7V – 5.3V) | Vcc |
| 3 | Contrast adjustment; through a variable resistor | VEE |
| 4 | Selects command register when low; and data register when high | Register Select |
| 5 | Low to write to the register; High to read from the register | Read/write |
| 6 | Sends data to data pins when a high to low pulse is given | Enable |
| 7 | 8-bit data pins | DB0 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | Backlight VCC (5V) | Led+ |
| 16 | Backlight Ground (0V) | Led- |

**RS (Register select)**

A 16X2 LCD has two order and information registers. The determination of the register is utilized to change starting with one register then onto the next. RS=0 for the register of directions, while RS=1 for the register of information.

**Command Register**

The guidelines given to the LCD are put away by the direction register. An order is a direction given to LCD to play out a predefined assignment, for example, instating it, clearing its screen, setting the situation of the cursor, controlling showcase, and so on. Order preparing happens in the direction register.

**Data Register:**

The information register will store the information that will be shown on the LCD. The information is the character's ASCII incentive to show on the LCD. It goes to the information register and is prepared there when we send information to the LCD. While choosing RS=1, the information register.

Read and Write Mode of LCD:

As stated, the LCD itself comprises of an interface IC. This interface IC can be perused or composed by the MCU. A large portion of the occasions we're simply going to keep in touch with the IC since perusing will make it increasingly perplexing and situations like that are exceptionally uncommon.Information such as cursor position, status completion interrupts, etc. can be read if necessary.

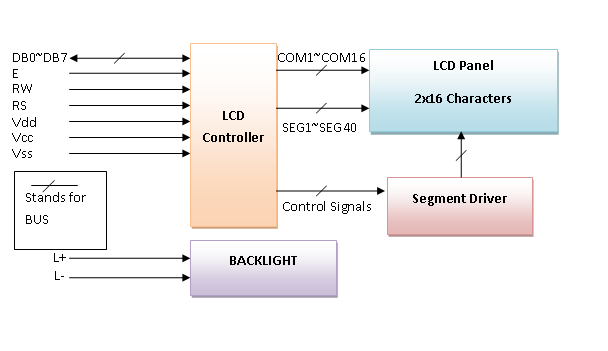
**LCD Commands:**

There are some preset commands in the LCD that we need to send to the LCD via some microcontroller. The following are some important command instructions:

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Hex Code** | **Command to LCD instruction Register** |
| 1 | 01 | Clear display screen |
| 2 | 02 | Return home |
| 3 | 04 | Decrement cursor (shift cursor to left) |
| 4 | 06 | Increment cursor (shift cursor to right) |
| 5 | 05 | Shift display right |
| 6 | 07 | Shift display left |
| 7 | 08 | Display off, cursor off |
| 8 | 0A | Display off, cursor on |
| 9 | 0C | Display on, cursor off |
| 10 | 0E | Display on, cursor blinking |
| 11 | 0F | Display on, cursor blinking |
| 12 | 10 | Shift cursor position to left |
| 13 | 14 | Shift cursor position to right |
| 14 | 18 | Shift the entire display to the left |
| 15 | 1C | Shift the entire display to the right |
| 16 | 80 | Force cursor to beginning ( 1st line) |
| 17 | C0 | Force cursor to beginning ( 2nd line) |
| 18 | 38 | 2 lines and 5×7 matrix |

## Command codes for LCD

**Block Diagram of LCD Display:-**

**[](http://www.circuitstoday.com/wp-content/uploads/2012/02/LCD-Display-Block-Diagram.png)**

**Control and display commands**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Instruction Code** | | | | | | | | | | **Instruction Code Description** | **Execution time** |
| RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| Read Data From RAM | 1 | 1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Read data from internal RAM | 1.53-1.64ms |
| Write data to RAM | 1 | 0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Write data into internal RAM (DDRAM/CGRAM) | 1.53-1.64ms |
| Busy flag & Address | 0 | 1 | BF | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | Busy flag (BF: 1→ LCD Busy) and contents of address counter in bits AC6-AC0. | 39 µs |
| Set DDRAM Address | 0 | 0 | 1 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | Set DDRAM address in address counter. | 39 µs |
| Set CGRAM Address | 0 | 0 | 0 | 1 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | Set CGRAM Address in address counter. | 39 µs |
| Function Set | 0 | 0 | 0 | 0 | 1 | DL | N | F | X | X | Set interface data length (DL: 4bit/8bit), Numbers of display line (N: 1-line/2-line) display font type (F:0→ 5×8 dots, F:1→ 5×11 dots) | 39 µs |
| Cursor or Display Shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | X | X | Set cursor moving and display shift control bit, and the direction without changing DDRAM data | 39 µs |
| Display & Cursor On/Off | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | Set Display(D),Cursor(C) and cursor blink(b) on/off control | 39 µs |
| Entry Mode Set | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | SH | Assign cursor moving direction and enable shift entire display. | 0µs |
| Return Home | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | Set DDRAM Address to “00H” from AC and return cursor to its original position if shifted. | 43µs |
| Clear Display | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Write “20H” to DDRAM and set DDRAM Address to “00H” from AC | 43µs |

4-bit and 8-bit Mode of LCD:

The LCD can work in two striking modes, the 4-bit mode and the 8-bit mode. We send the information snack through snack in 4 bit mode, first upper chomp, by then lower snack. For those of you who don't have the foggiest idea what a goody is: a chomp is a four-piece gathering, so a byte's lower four bits (D0-D3) are the lower snack, while a byte's upper four bits (D4-D7) are the higher snack. This enables us to send 8 bit data. This connects with us to send 8 bit data. Whereas in 8 bit mode we can send the 8-bit information truly in one stroke since we utilize all the 8 information lines. You need to get it now; yes 8-bit mode is quicker and immaculate than 4-bit mode. In any case, the fundamental shortcoming is that it needs 8 microcontroller-related information lines. This will result in our MCU coming up short on I/O pins, so 4-bit mode is extensively utilized. To set these modes, no control pins are used.

**BULB**

The electronic bulb is the simplest electrical lamp that was invented for illumination more than a century ago. It was the small and simplest light that brightened the dark space. The electronic bulb is also known as an incandescent lamp, incandescent light globe or incandescent light bulb. Bulb comes in different sizes and light output and operates with a voltage range from 1.5 Volts to about 300 Volts. Now let us study the parts and structure of the bulb in detail.

Structure Of The Electronic Bulb

An electronic bulb is a small and simple light source that uses a wire filament to glow on the application of electricity. The structure of incandescent light bulbs is shown in the figure below.

The light bulb consists of three key parts

* The filament
* The glass bulb
* The base of the light bulb

The filament, which is a coiled thin wire, is made of tungsten. Tungsten is chosen as filament sine it has a high melting point, which avoids the melting of the filament at high temperatures.

The filament is enclosed in a globe-shaped glass mount and is connected with copper and lead wires connected to the lamp’s base. The wires and the filament are enclosed in a glass bulb, which is filled with an inert gas like argon. Since argon is an inactive gas, it protects the filament from burning as well as increases the lifetime of the filament. Thin glass is used to manufacture the bulb, preventing the air from reaching the filament to protect it from burning.

When electric power is passed through the bulb, it reaches the filament through copper and lead wires. The base holds the bulb upright and connects to the electric circuit.

Copper and lead wires let the electricity pass from the base to the tungsten filament. It causes the filament to emit light and glow.

There are two types of bulb bases:

* Spiral base
* Two side nails base

The spiral base bulb features a piece of lead that directly connects the lamp to the electric circuit.

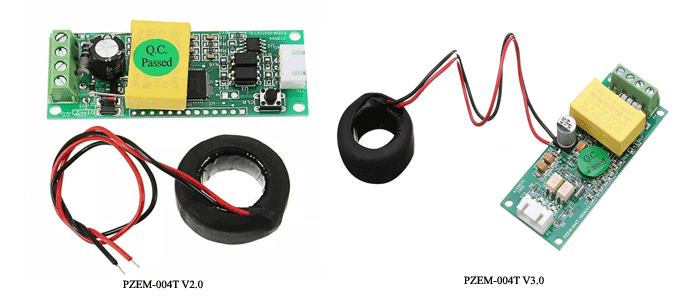


**PZEM004T sensor**

PZEM-004T is an electronic module that functions to measure: Voltage, Current, Power, Frequency, Energy and Power Factors. With the completeness of these functions / features, the PZEM-004T module is ideal for use as a project or experiment for measuring power on an electrical network such as a house or building.

The PZEM-004T module is produced by a company called Peacefair, there are 10 Ampere and 100 Ampere models. Please be careful because the wiring between the 10 Ampere models with 100 Amperes is different, if a short circuit or a short circuit can occur in the electrical network.

**The DifferenceBetween PZEM-004T V2.0 and PZEM-004T V3.0**

PZEM-004T V2.0 and PZEM-004T V3.0

* There is Reset Energi button on V2.0
* In V3.0 the function to Reset Energy uses software, so there is no push button for Reset Energy
* V3.0 is an upgraded version of V2.0 so that the level of accuracy is better
* The conversion / reading time on V3.0 is faster than V2.0
* The protocol used for data communication is different between the two

**PZEM-004T Current Transformer ( CT ) Split Core**

In addition there is also the PZEM-004T 100 Ampere which uses the Current Transformer split core model. Because it uses a split core, of course it has advantages in its ease of use because it can be directly installed on a power network cable that is already installed without having to remove the power cable.

Examples of PZEM-004T modules that use CT split core can be seen in the image below



PZEM-004T CT Split Core

**Specifications / Features PZEM-004T**

Although there are some differences between PZEM-004T V2.0 and PZEM-004T V3.0, functionally or featureally, both have similarities. Following are the features or specifications of the PZEM-004T module :

A. Function

* Measurement function (voltage, current, active power).
* Power button clear / reset Energy (PZEM-004T V2.0)
* Power-down data storage function (cumulative power down before saving)
* TTL Serial Communication
* Power Measurement : 0 ~ 9999kW
* Voltage Measurement : 80 ~ 260VAC
* Current Measurement : 0 ~ 100A

B. Spesifications

* Working voltage: 80 ~ 260VAC
* Rated power: 100A / 22000W
* Working Frequency: 45-65Hz
* Measurement accuracy: 1.0

**PZEM-004T programming**

PZEM-004T module is very easy to use in programming using various types of microcontroller boards such as Arduino, ESP8266, STM32, WeMos, NodeMCU, Raspberry Pi etc. because it uses TTL serial communication.

It should be noted that the protocols used in PZEM-004T V2.0 and PZEM-004T V3.0 are different, so the library and programming are also different. Many people think that the PZEM-004T module is corrupt or doesn’t work just because they don’t know and are using the wrong library.

**SOFTWARE REQUIREMENTS**

**Arduino IDE:**

**Arduino IDE**where IDE stands for Integrated Development Environment – An official software introduced by Arduino.cc, that is mainly used for writing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go.

**Introduction to Arduino IDE:**

* Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
* It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
* It is easily available for operating systems like MAC, Windows, and Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
* A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, [Arduino Micro](https://www.theengineeringprojects.com/2018/09/introduction-to-arduino-micro.html) and many more.
* Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
* The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
* The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
* This environment supports both C and C++ languages.

**How to install Arduino IDE:**

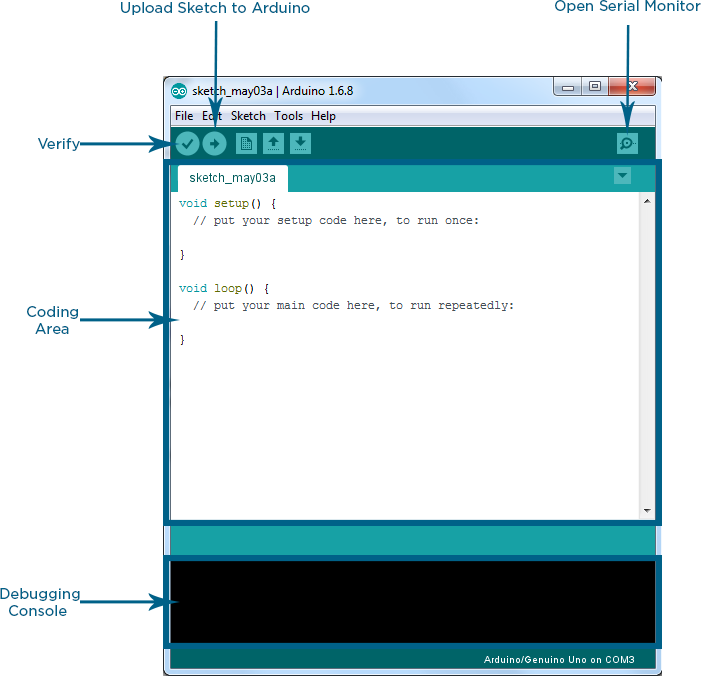
You can download the Software from [Arduino](https://www.arduino.cc/en/Main/Software) main website. As I said earlier, the software is available for common operating systems like Linux, Windows, and MAX, so make sure you are downloading the correct software version that is easily compatible with your operating system.

* If you aim to download Windows app version, make sure you have Windows 8.1 or Windows 10, as app version is not compatible with Windows 7 or older version of this operating system.

The IDE environment is mainly distributed into three sections

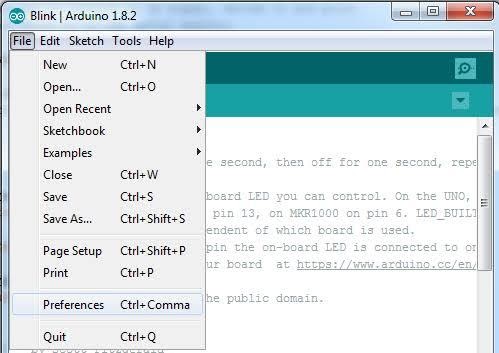
* **1. Menu Bar**
* **2. Text Editor**
* **3. Output Pane**

As you download and open the IDE software, it will appear like an image below.

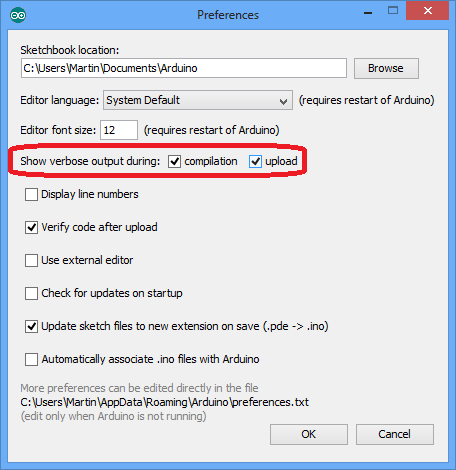


The bar appearing on the top is called **Menu Bar** that comes with five different options as follow

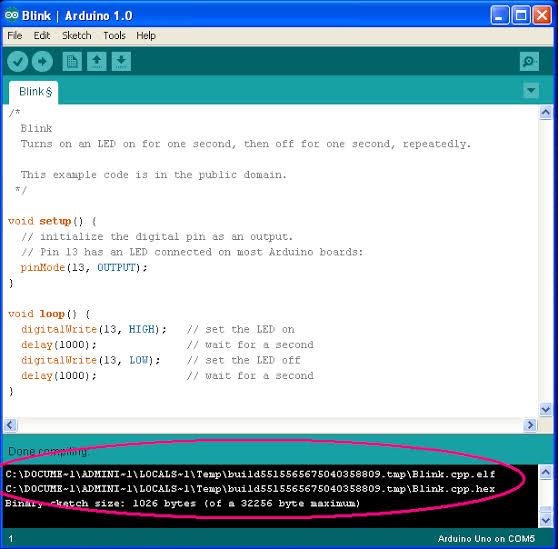
* **File** – You can open a new window for writing the code or open an existing one. Following table shows the number of further subdivisions the file option is categorized into.

****

As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button.

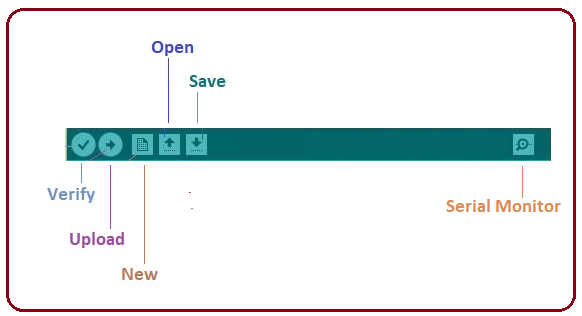


And at the end of compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.

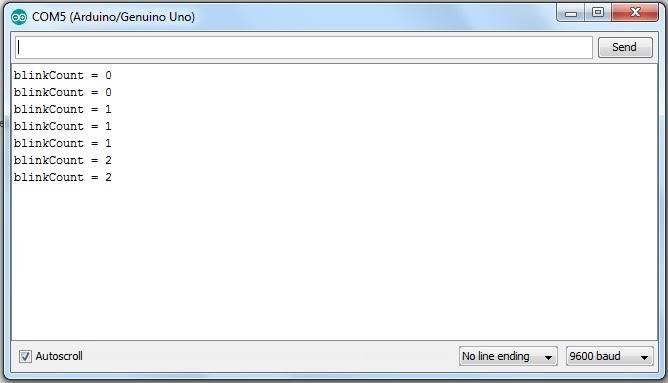
****

* **Edit** – Used for copying and pasting the code with further modification for font
* **Sketch** – For compiling and programming
* **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.
* **Help** – In case you are feeling skeptical about software, complete help is available from getting started to troubleshooting.

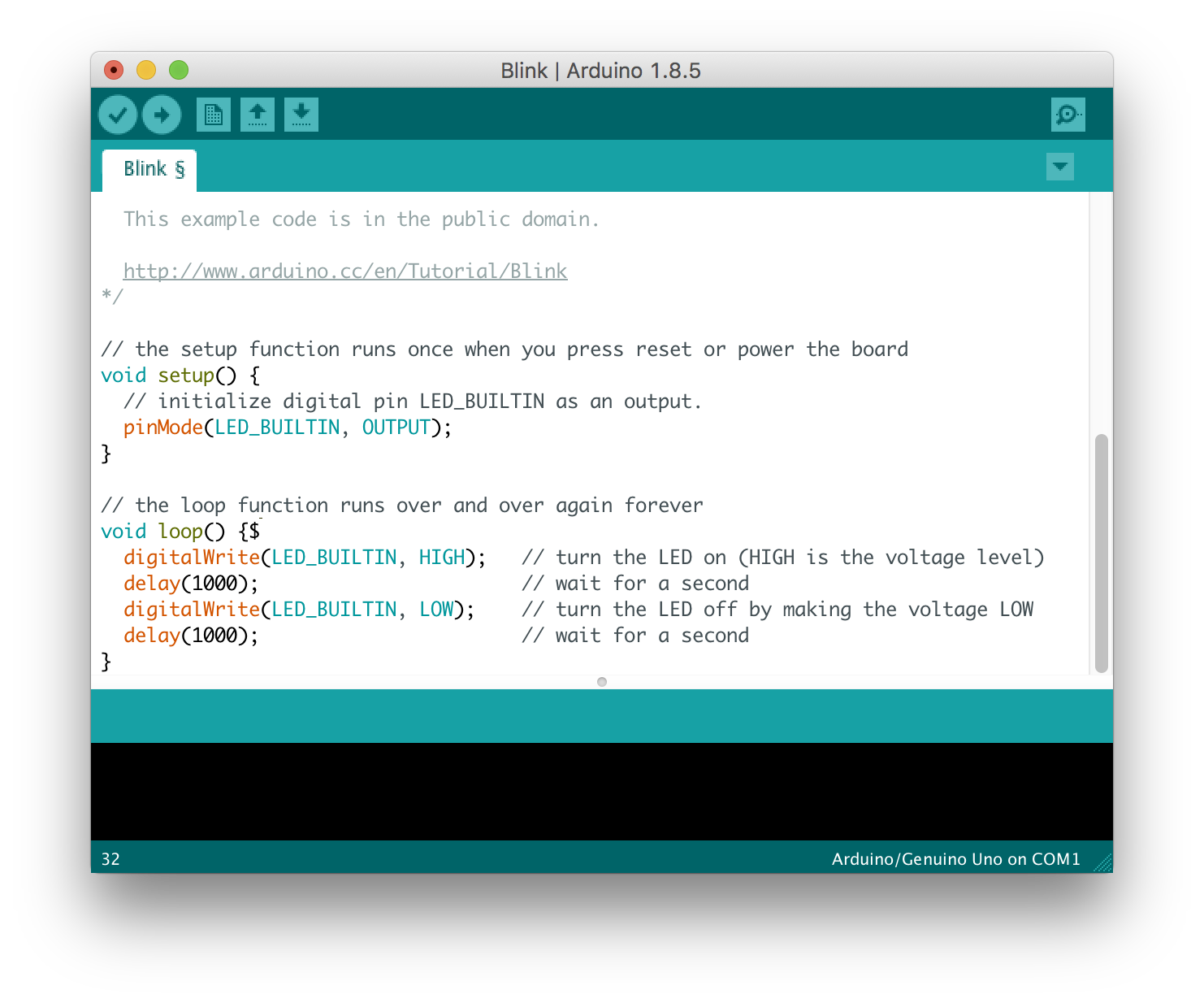
The **Six Buttons** appearing under the Menu tab are connected with the running program as follow.

****

* The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
* The arrow key will upload and transfer the required code to the Arduino board.
* The dotted paper is used for creating a new file.
* The upward arrow is reserved for opening an existing Arduino project.
* The downward arrow is used to save the current running code.
* The button appearing on the top right corner is a **Serial Monitor** – A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. You can also go to the Tools panel and select Serial Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.
* You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below.



The main screen below the Menu bard is known as a simple text editor used for writing the required code.



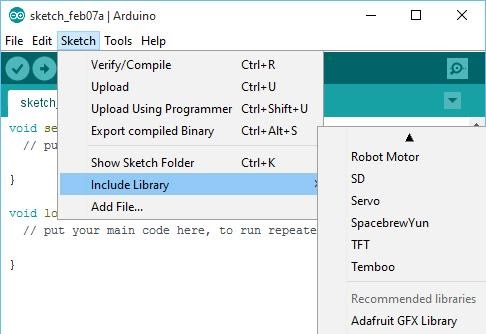
The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors occurred in the program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.



More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

**Libraries:**

Libraries are very useful for adding the extra functionality into the Arduino Module. There is a list of libraries you can add by clicking the Sketch button in the menu bar and going to Include Library.



As you click the Include Library and Add the respective library it will on the top of the sketch with a #include sign. Suppose, I Include the EEPROM library, it will appear on the text editor as

#include <EEPROM.h>.

Most of the libraries are preinstalled and come with the Arduino software. However, you can also download them from the external sources.

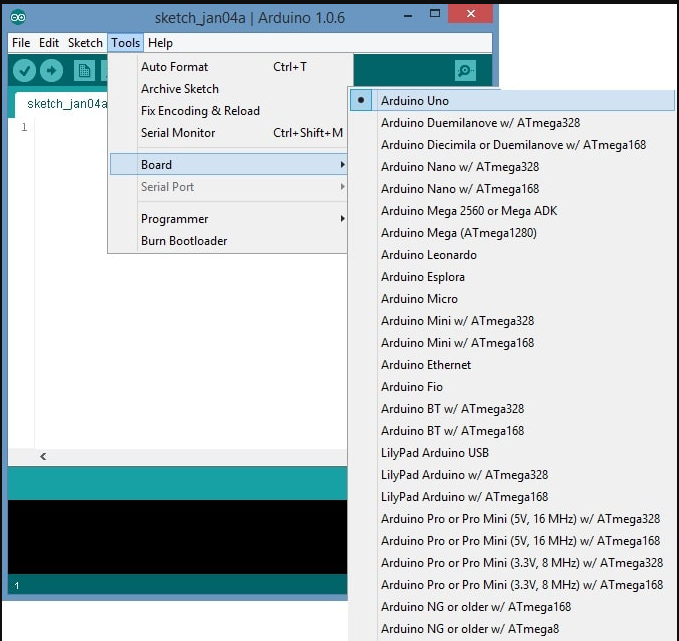
**Making pins Input and output:**

The digitalRead and [digitalWrite](https://www.theengineeringprojects.com/2018/09/how-to-use-digitalwrite-arduino-command.html) commands are used for addressing and making the Arduino pins as an input and output respectively.

These commands are text sensitive i.e. you need to write them down the exact way they are given like digitalWrite starting with small “d” and write with capital “W”. Writing it down with Digitalwrite or digitalwrite won’t be calling or addressing any function.

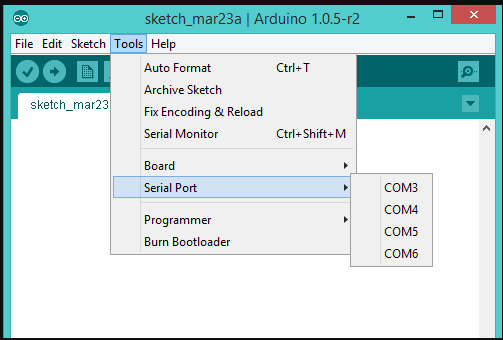
**How to select the board:**

In order to upload the sketch, you need to select the relevant board you are using and the ports for that operating system. As you click the Tools on the Menu, it will open like the figure below.



* Just go to the “Board” section and select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.

Following figure shows the COM4 that I have used for my project, indicating the Arduino Uno with COM4 port at the right bottom corner of the screen.



* After correct selection of both Board and Serial Port, click the verify and then upload button appearing in the upper left corner of the six button section or you can go to the Sketch section and press verify/compile and then upload.
* The sketch is written in the text editor and is then saved with the file extension .ino.

It is important to note that the recent Arduino Modules will reset automatically as you compile and press the upload button the IDE software, however, older version may require the physical reset on the board.

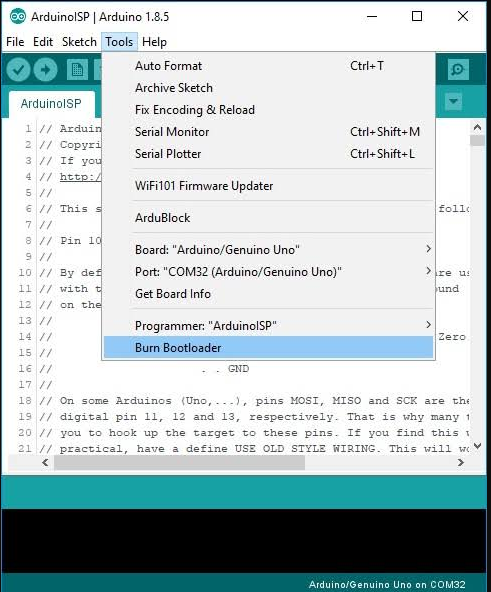
* Once you upload the code, TX and RX LEDs will blink on the board, indicating the desired program is running successfully.

**Note**: The port selection criteria mentioned above is dedicated for Windows operating system only, you can check this [Guide](https://www.arduino.cc/en/Guide/Environment) if you are using MAC or Linux.

* The amazing thing about this software is that no prior arrangement or bulk of mess is required to install this software, you will be writing your first program within 2 minutes after the installation of the IDE environment.

**BootLoader:**

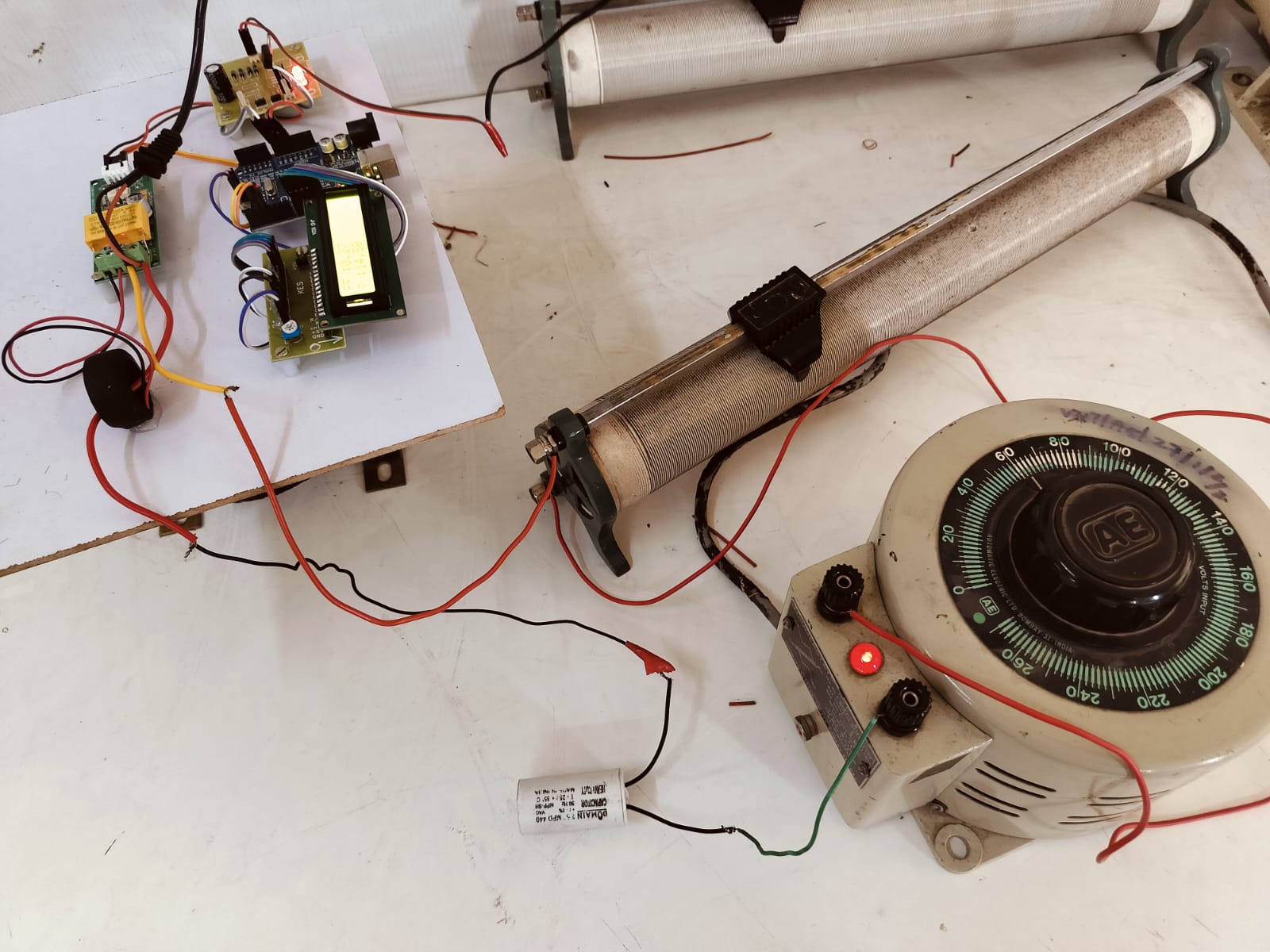
As you go to the Tools section, you will find a bootloader at the end. It is very helpful to burn the code directly into the controller, setting you free from buying the external burner to burn the required code.



When you buy the new Arduino Module, the bootloader is already installed inside the controller. However, if you intend to buy a controller and put in the Arduino module, you need to burn the bootloader again inside the controller by going to the Tools section and selecting the burn bootloader.

**CHAPTER 7**

**TESTING RESULTS**



**A close-up of a machine

Description automatically generated**

**CHAPTER 8**

**CONCLUSION**

In conclusion, both the Phase Angle Measurement Circuit and the Power Factor Measurement Circuit play crucial roles in assessing and optimizing the performance of electrical systems. The Phase Angle Measurement Circuit offers precise determination of the phase relationship between voltage and current waveforms, aiding in the analysis of system stability and efficiency. On the other hand, the Power Factor Measurement Circuit provides valuable insights into the utilization of electrical power, helping to identify opportunities for improving energy efficiency and reducing wastage. By integrating these circuits into power monitoring and control systems, engineers can enhance the reliability, efficiency, and sustainability of electrical networks across various industries. Together, these circuits contribute to the advancement of electrical engineering practices, enabling the development of more efficient and environmentally friendly power systems for the future.

.

**REFERENCES**

* Li, C., Zhao, Y., & Li, Q. (2017). Phase Angle Measurement of Voltage and Current Signals in Power System Based on Arduino. 2017 IEEE International Conference on Smart Grid and Clean Energy Technologies (ICSGCE). doi: 10.1109/icsgce.2017.241
* Hu, L., Liu, W., Liu, Y., & Chen, W. (2018). Design of a Portable Phase Angle Measurement System Based on Arduino. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). doi: 10.1109/iaeac.2018.8546823
* Hong, W., & Zhang, J. (2020). Research on Power Factor Measurement Circuit of Smart Meter Based on Improved Design. 2020 International Conference on Electronics Technology (ICET). doi: 10.1109/icet48669.2020.9182821
* Rovira, S., Ordonez, R., & Rodriguez, P. (2015). A Novel Power Factor Measurement Circuit Using Low-Cost Microcontrollers. IEEE Transactions on Instrumentation and Measurement, 64(12), 3441-3450. doi: 10.1109/tim.2015.2418604