CHAPTER-1

INTRODUCTION

In Current Technology, contamination or pollution is expanding quickly and in our environment, additionally dust particles are expanding because of these issues a large number of the general population harmed like breathing issue, asthma, lung growth, perpetual obstructive aspiratory disease (This sickness is described by expanding windedness.), and emphysema (A condition in which the air sacs of the lungs are harmed and augmented, causing shortness of breath.). In view of these reasons and issues, we are doing this programmed debilitate with tidy sensor task to lessen or stay away from these issues.

In the present market officially one device (Air XD continuous tidy estimation framework) is there identified with our undertaking yet this gadget just identify the tidy particles noticeable all around and it will show the clean levels on the screen/show and it will illuminate us by utilizing alert/ringer

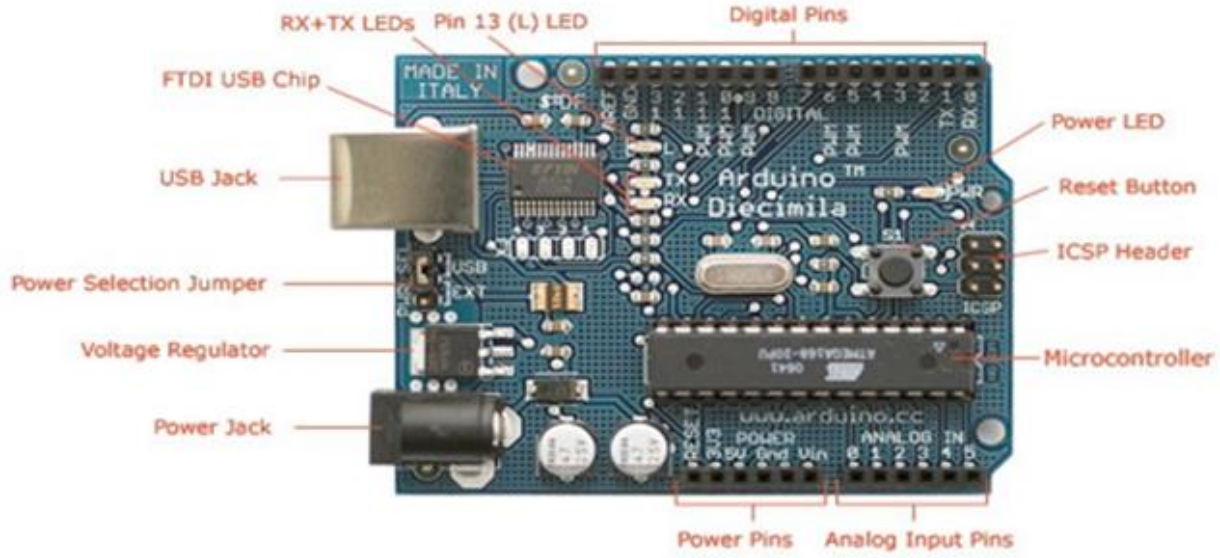
In this project, we are using dust sensor which detects the dust particles in the quality of the request of millimetres too. When any tidy molecule is discovered, it naturally turns on the fumes fan. Also, it will advise us by utilizing a signal and the outcomes are shown are on the LCD screen ie.dust content levels .The fan will keep running by utilizing the engine driver which is interfaced with Atmega328 microcontroller and control supply is given to this extend, for example, transformer, transfer, connect rectifier.

CHAPTER-2
HARDWARE REQUIREMENTS

2.1 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.



Arduino Board Description

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.

An important feature of the Arduino is that you can create a control program on the host PC, download it to the Arduino and it will run automatically. Remove the USB cable connection to the PC, and the program will still run from the top each time you push the reset button. Remove the battery and put the Arduino board in a closet for six months. When you reconnect the battery, the last program you stored will run. This means that you connect the board to the host PC to develop and debug your program, but once that is done, you no longer need the PC to run the program.

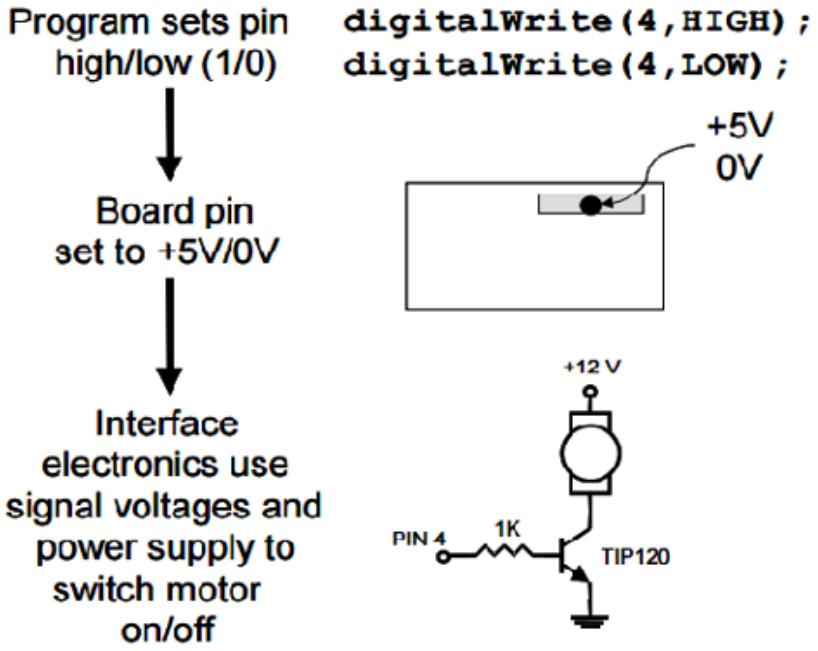
Arduino Hardware

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 labelled 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.

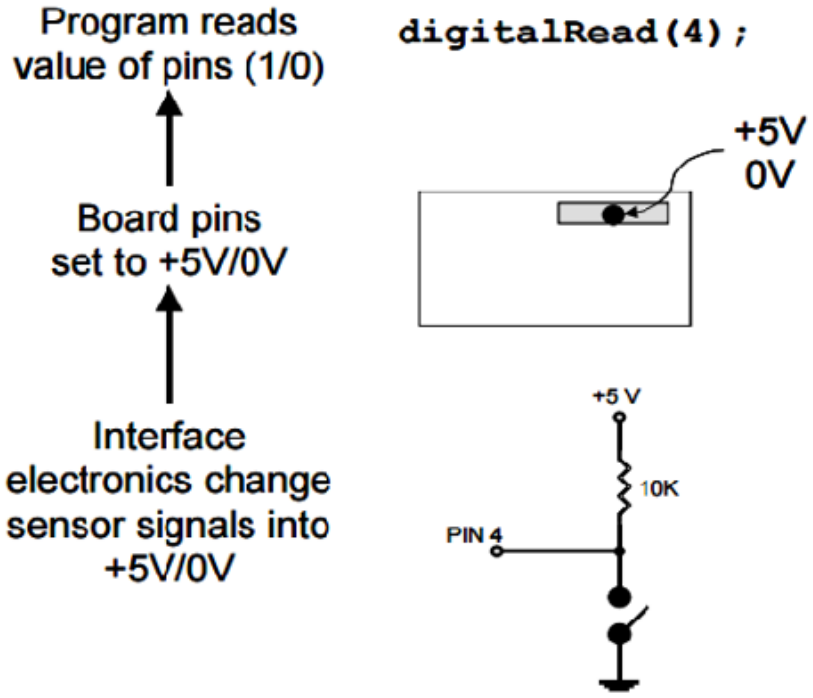
To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off.
The sequence of events is shown in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

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-unpredictable Memory Segments

– 4/8/16/32K Bytes of In-System Self-Programmable Flash program 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 maintenance: 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-direcct 10-bit ADC in TQFP and QFN/MLF bundle Temperature Measurement – 6-divert 10-bit ADC in PDIP Package Temperature Measurement

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Byte-arranged 2-wire Serial Interface (Philips I2 C perfect)

– 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-spare, Power-down, Standby, and Extended Standby

- I/O and Packages

– 23 Programmable I/O Lines

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

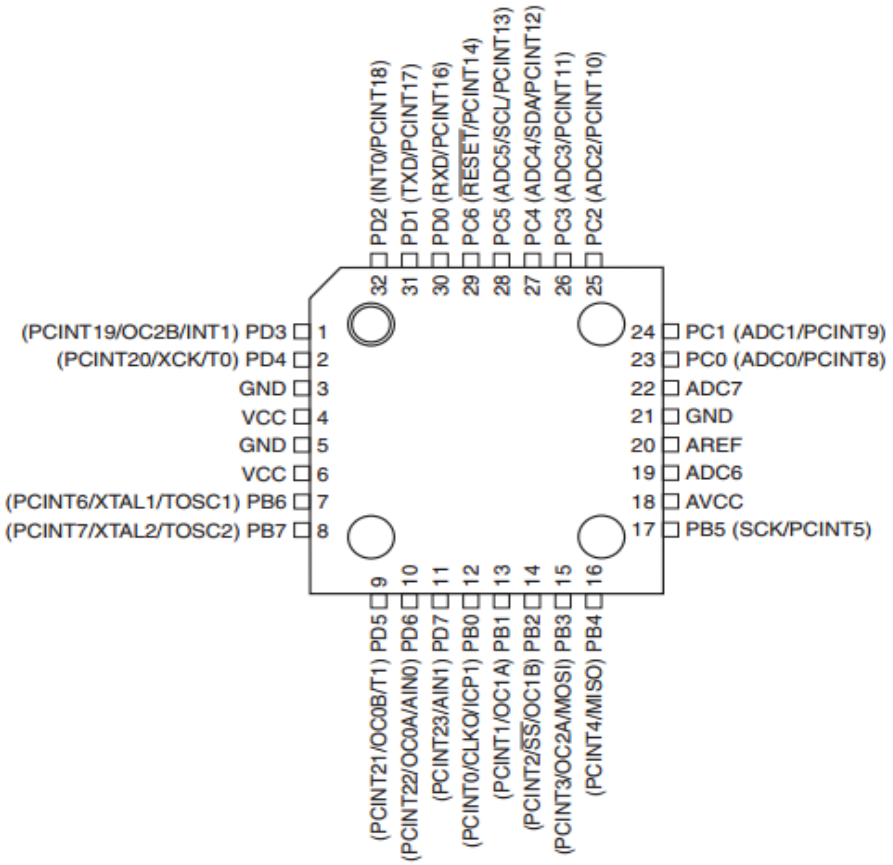
  – Active Mode: 0.2 mA

  – Power-down Mode: 0.1 µA

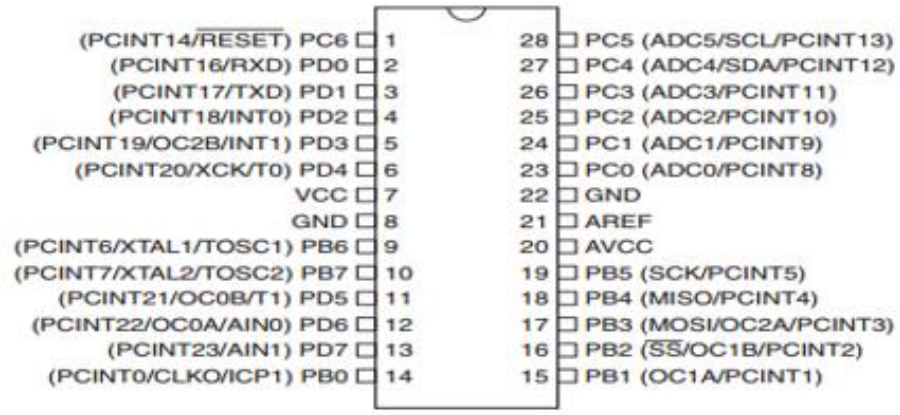
  – Power-spare Mode: 0.75 µA (Including 32 kHz RTC)

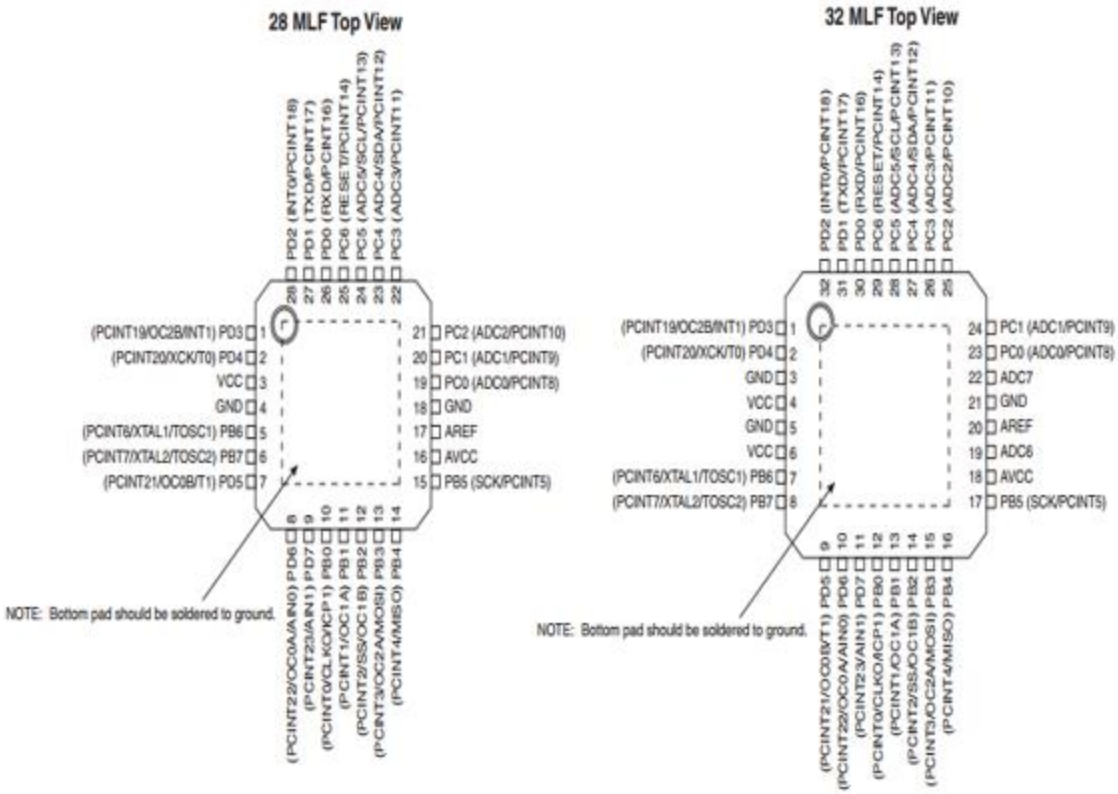
PIN CONFIGURATIONS

TQFP Top View



PDIP





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" and "System Clock and Clock Options" .

.

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 un programmed, 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. Shorter pulses are not guaranteed to generate a Reset. The various special features of Port C are elaborated in "Alternate Functions of Port C".

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".

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

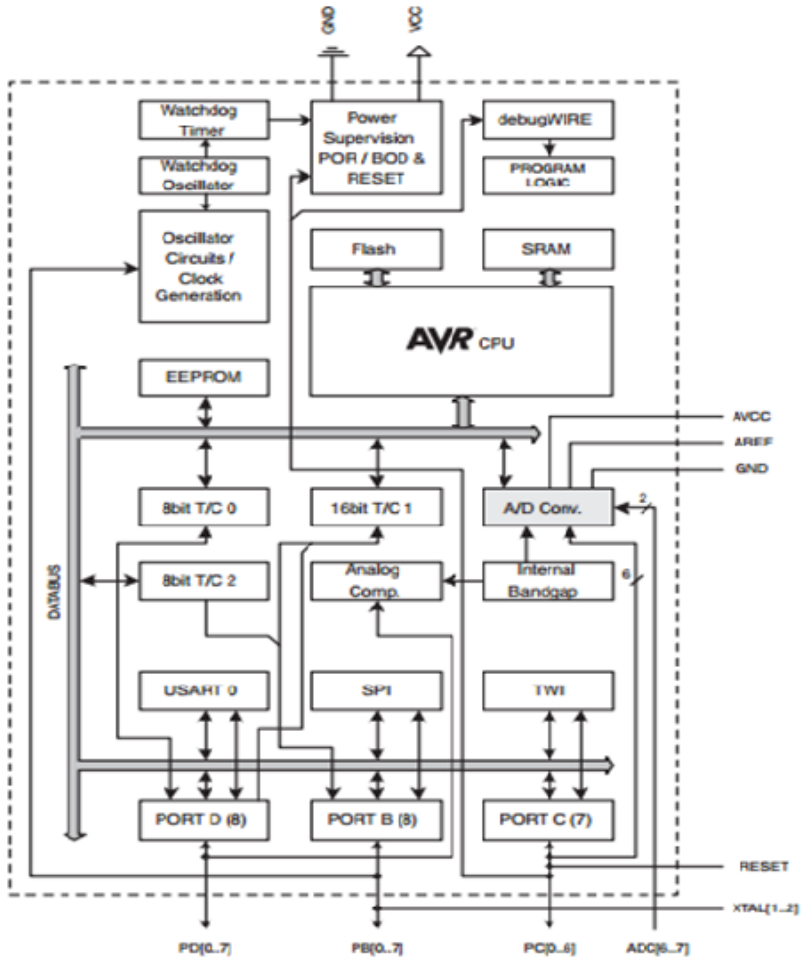
ADC7:6 (TQFP and QFN/MLF Package Only): In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

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



Architecture of ATMEGA328P

The ATmega48PA/88PA/168PA/328P provides the following features: 4/8/16/32K bytes of InSystem 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.

Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector

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 boot loader). 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 K.Ohms. 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, labelled 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 Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; 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.

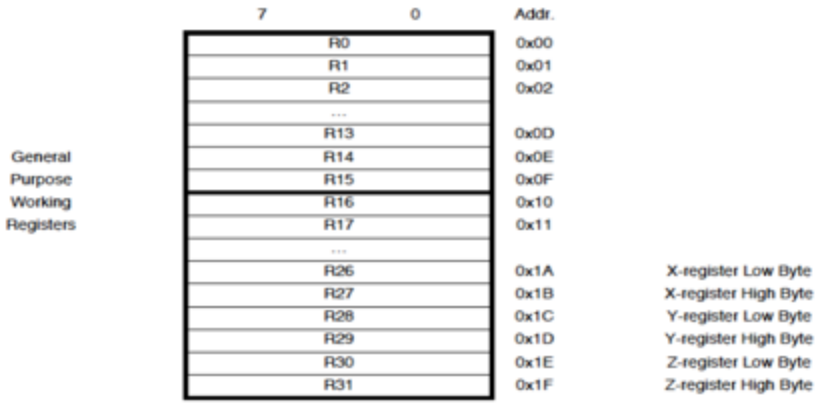
Physical Characteristics

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

Register Memory

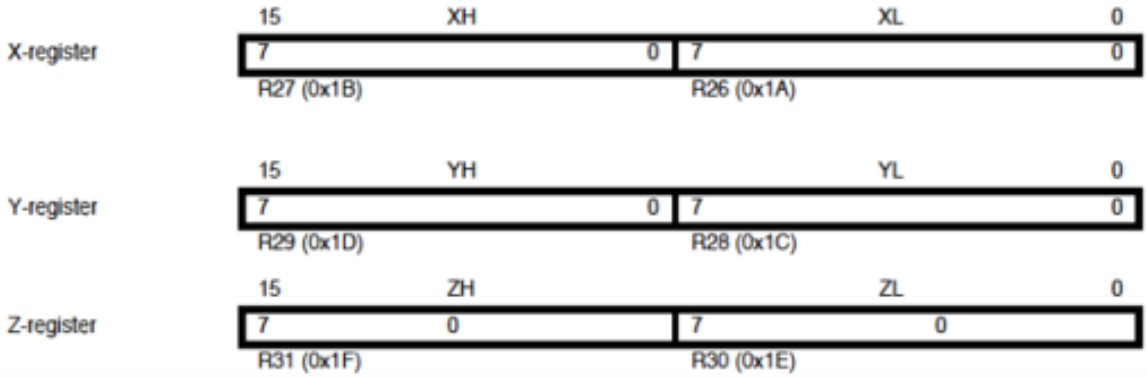


Special Addressing Registers

- X, Y and Z registers

  16-bit registers made using registers 26 – 31
- Support indirect addressing

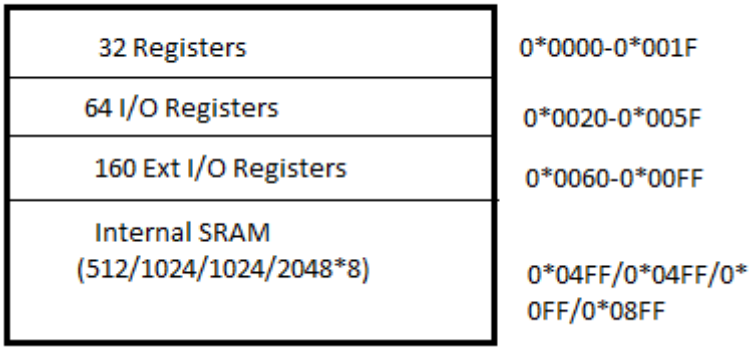
Addressing Register Memory



AVR Memory

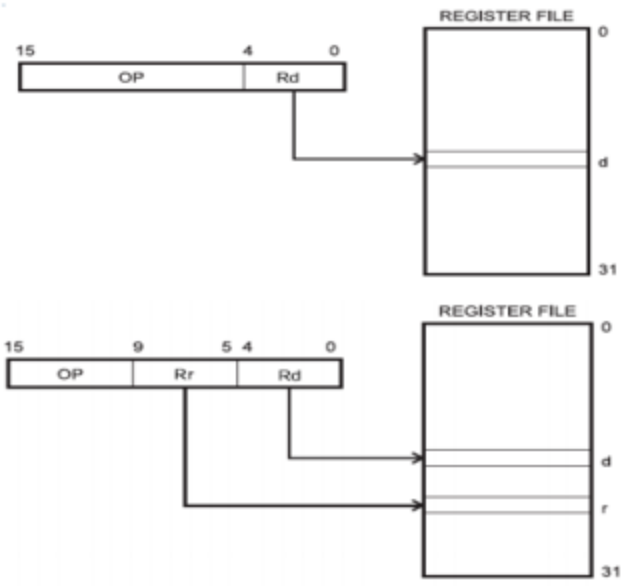
- Program memory – Flash
- Data memory – SRAM

Data Memory

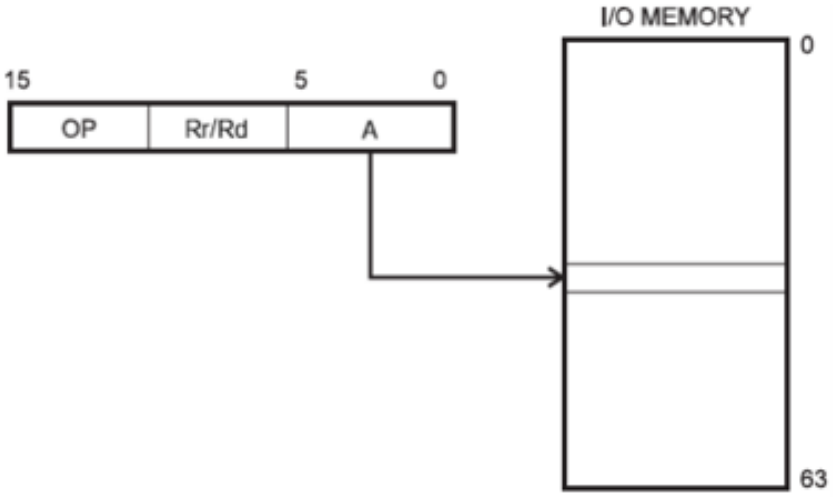


Addressing Modes

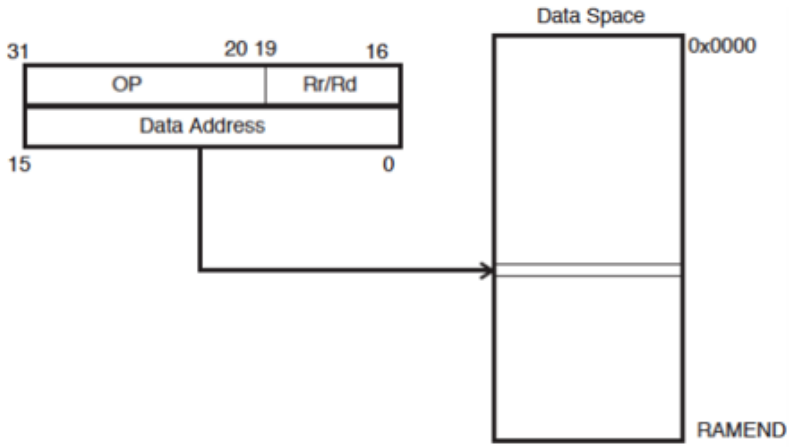
- Direct register addressing



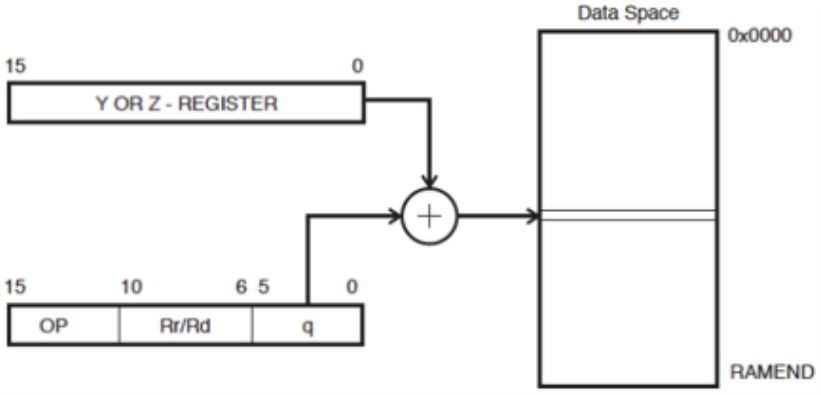
- Direct I/O addressing



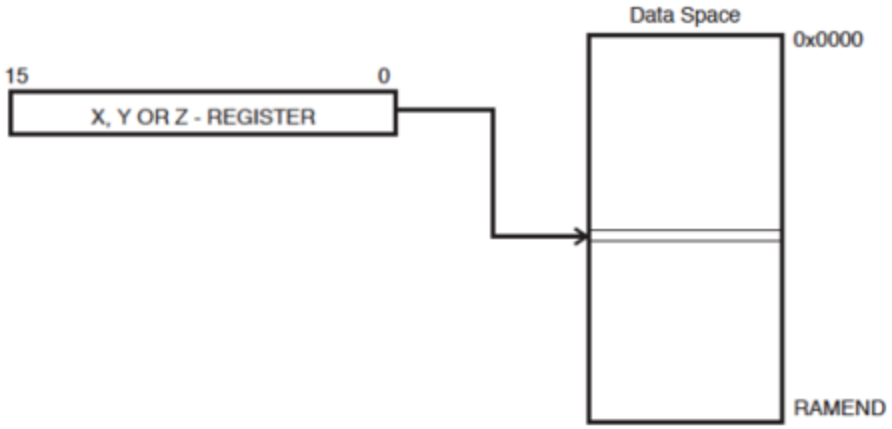
- Direct data memory addressing



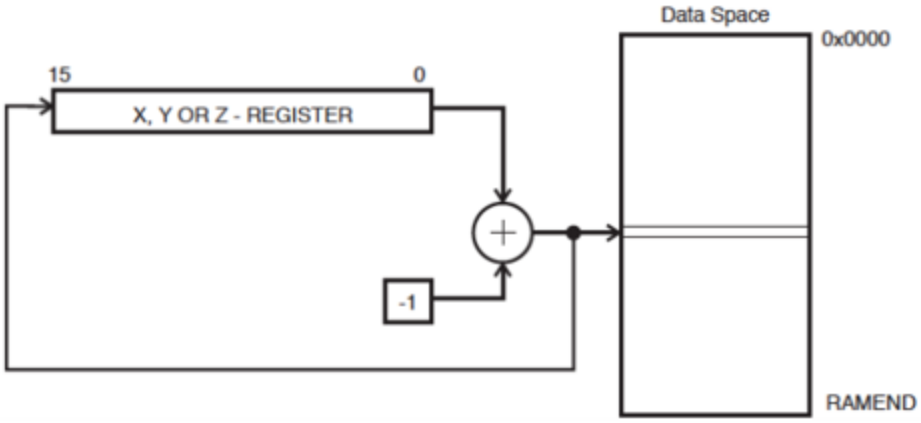
- Direct data memory with displacement addressing



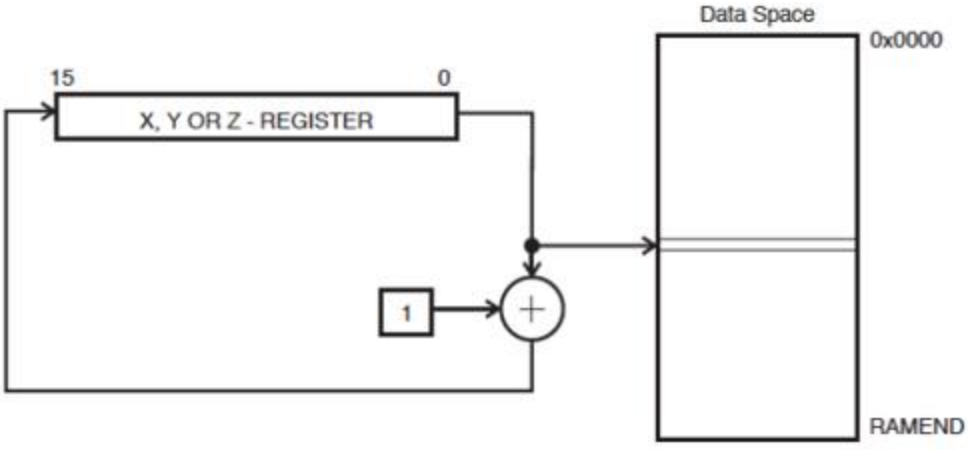
- Indirect data memory addressing



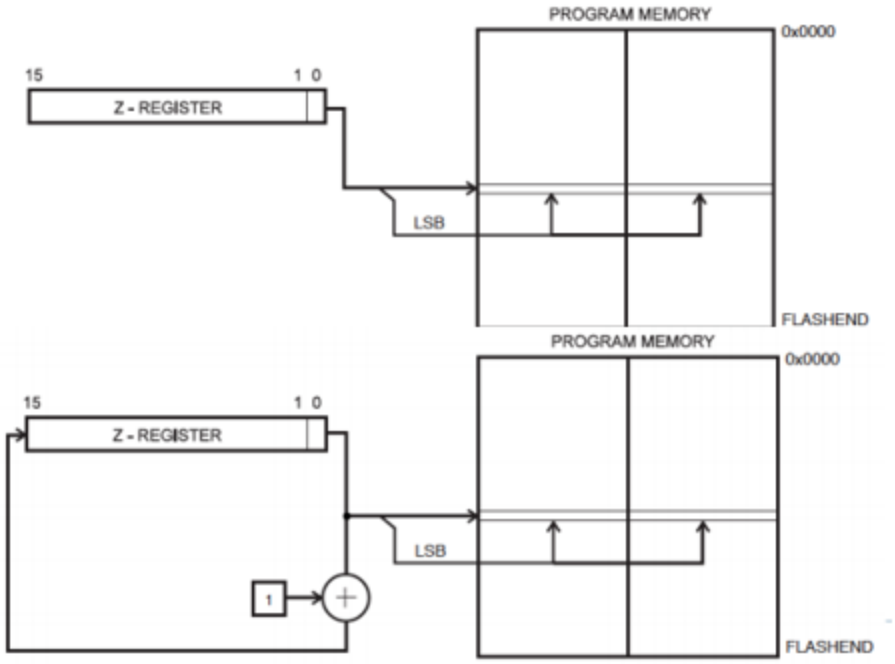
- Indirect data memory addressing with pre-decrement



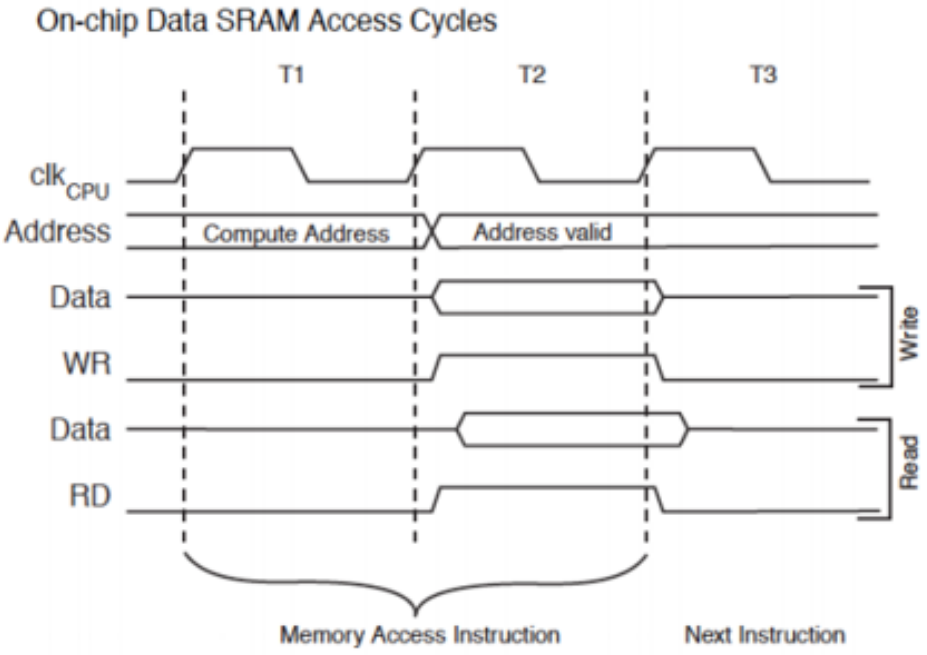
- Indirect data memory addressing with post-increment



- Program memory addressing (constant data)



SRAM Read/Write Timing



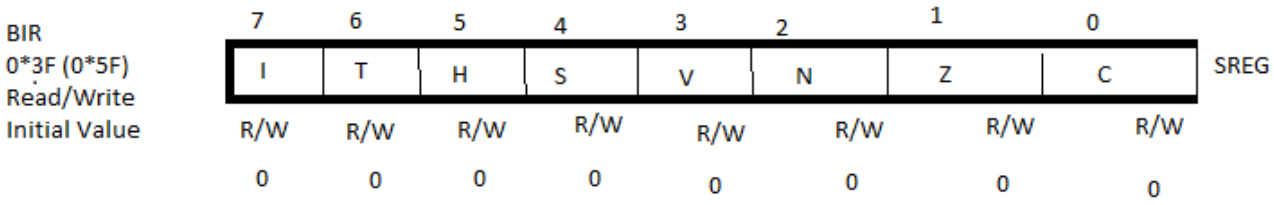
Stack Pointer Register

- 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

Stack Pointer Register

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

Program Status Register (PSR)



Status bits by instructions or checked by branch/skip instruction

I-Global Interrupt Enable
T-Flag Bit
H-Half Carry (BCD arithmetic)
S-Sign Flag
V- Overflow flag
N-Negative
Z-Zero flag
C-Carry flag

I/O Ports

- **3 8-bit Ports(B,C,D)**
- **Each Port controlled by 3 8-bit registers**
  Each bit controls one I/O pin

DDRx –Direction register
Defines whether a pin is an input (0) or and output (1)
PINx- Pin output value
Reading this "register" returns value of pin
PORTx-Pin output value
Writing this register sets of pin.

2.2 LCD

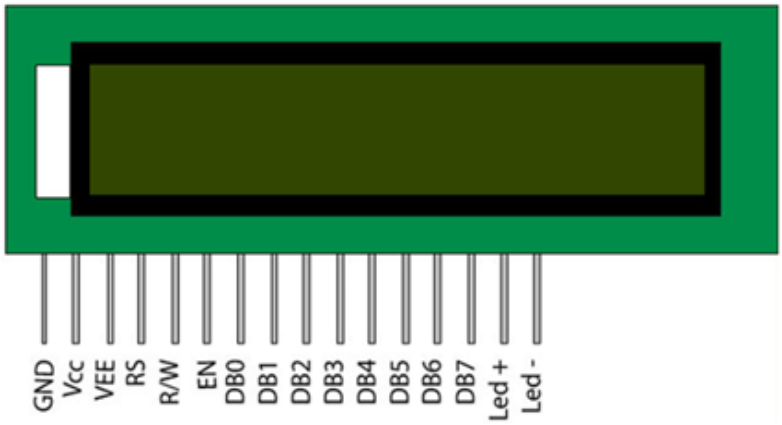
LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs.

The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

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

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

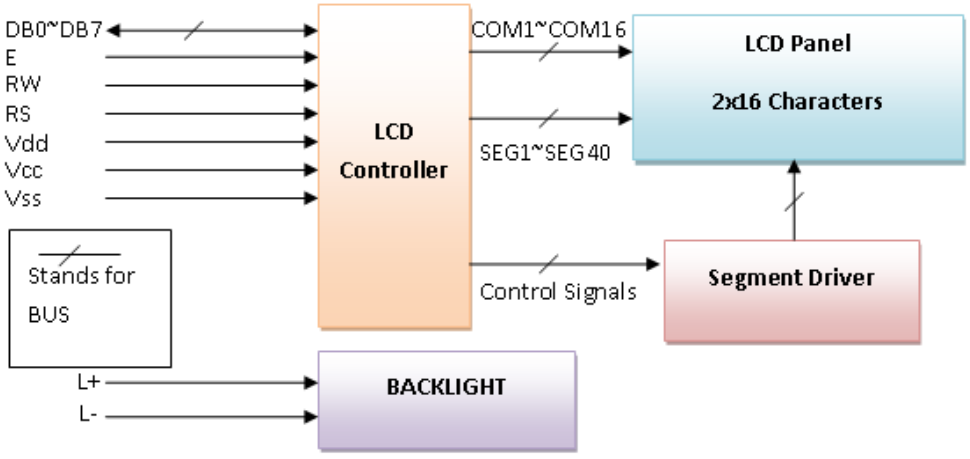
Pin Diagram



Pin Description

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast alternation; through a variable resistor	VEE
4	Selects command register when low; and information enlist when high Register Select	Register Select
5	Low to write to the register; High to peruse from the register Read/compose	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-

Block Diagram of LCD Display



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

In AC=Instruction counter there are four sorts of guidelines

i. Function set guidelines
ii. Address set guidelines
iii. Data exchange guidelines with inner ram.
iv. And other guidelines

Details of the Instructions

1) Reading Data from RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Read 8bit binary data from DDRAM/CGRAM

The selection of RAM is set by the previous address set instruction. If the address set instruction of RAM is not performed before this instruction, the data that is read first is invalid, because the direction of AC is not determined. If the RAM data is read several times without RAM address set instruction before read operation, the correct RAM data from the second, but the first data would be incorrect, as there is no time to transfer RAM data. In case of DDRAM read operation, cursor shift instruction plays the same role as DDRAM address set instruction; it also transfers RAM data to the output data registers.

After read operation, the data address counter is automatically increased or decreased by 1 according to the entry mode. After CGRAM read operation, display shift may not be executed properly.

*In case of RAM write operation, AC is increased or decreased by 1 like that of the read operation. In this time AC indicates the next address position, but the previous data can only by the read instruction

2) Writing data to ram

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write binary 8bit data to DDRAM/CGRAM. The selection of CGRAM or DRAM is set by the previous address set instruction; DDRAM address set, CGRAM address set. RAM set instruction can also determine the AC direction to RAM.

After write operation, the address is automatically increased or decreased by 1 according to the entry mode.

2) Busy Flag and Address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

By making this read out operation, it can be determined if the LCD is performing some internal operation or not. If Busy Flag (BF) is high, some internal operation is going inside the LCD at that particular moment. To perform further operation the data source (e.g. micro controller) must wait for the BF to go low. Here, the address counter value can also be read

3) Setting DDRAM Address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set DDRAM address to AC, this instruction makes DDRAM data available from MPU. In 1-line display mode, DDRAM address rangers from “00H” to “4FH”. In 2-line display mode, DDRAM address in the first line ranges from “00H” to “27H”, and DDRAM address in the 2nd line is from “40H” to “67H”.

4) Setting CGRAM address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Set CGRAM deliver to AC. This direction makes CGRAM information accessible from MPU.

5) Function Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

DL: Interface data length control bit
DL='1' implies 8bit method of information exchange.

DL='0' implies 4bit method of information exchange

At the point when 4-bit mode is enacted, the information should be moved in two sections, first higher 4bits, and after that lower 4 bits.

N: display line number control bit
N='1' will permit characters to show in 2-lines

N='0' will permit characters to show in the primary line as it were

F: display font control bit
F='0' will utilize the 5×8 dots design show mode

F='1' will utilize 5×11 specks arrange show mode

6) Cursor Shifting

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

Without writing or reading the display data, shifting right/left cursor position or display.

This instruction is made to correct or search or display data. During 2-line display mode, cursor moves to the 2nd line after the 40th digit of the 1st line.
When displayed data is shifted repeatedly, each line shifts individually.

When display shift is performed, the contents of the address counter are not changed.

7) Display On/Off Control

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

This guideline controls Display, Cursor and cursor flicker.

D: Display On/Off control bit

D='1' implies whole show is turned on

D='0' implies whole show is killed. Be that as it may, Display information stays in DDRAM.

C: cursor On/Off control bit

C='1' turns on the cursor

C='0' kills the cursor. Be that as it may, I/D enlist holds the information

B: Cursor blink On/Off control bit

B='1' influences cursor to squint intermittently.

B='0' stops the cursor to flicker and cursor looks unfaltering if the Cursor is turned on.

8) Entry Mode Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	SH

This guideline sets the moving heading of cursor and show.

When I/D= '1' cursor moves to one side and DDRAM address is expanded by 1.

When I/D= '0' cursor moves to one side and DDRAM address is diminished by 1.

CGRAM works similarly in this setting.

9) Return Home

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	X

This instruction sets the address counter to '00H', and returns the cursor to the first column of first line. And if display is shifted previously, this instruction shifts this too. The DDRAM contents don't change in this instruction.

10) Clear display

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

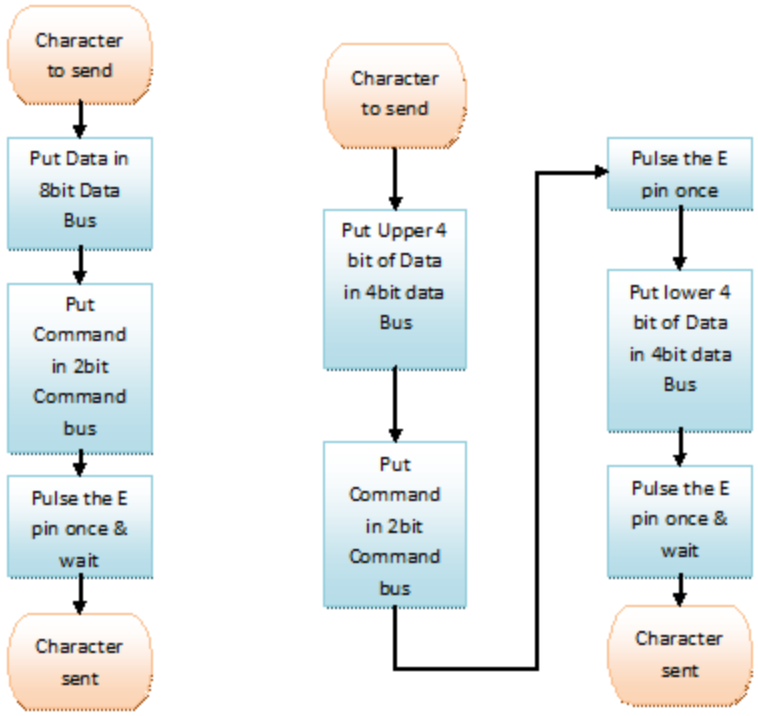
Clear all the display data by writing "20H" (ASCII code of 'space' character) to all DDRAM address, AND set value DDRAM address counter (AC) to "00H". It returns the cursor to the first column of first line and sets the entry mode to increment mode (I/D='1').

8-bit and 4-bit interfacing of LCD

Now the question is how to display data in the LCD or give command to it. There is two modes of data transfer are supported by LCD displays. One is 4bit mode, another is 8 bit mode. To transfer data In 8 bit mode, first put your data in the 8bit bus, then put command in the command bus and then pulse the enable signal.

To send data in 4bit mode; first put upper 4bit in the 4 bit data bus connected to 4MSB pins of LCD display, then put control signals in the control bus, then pulse the E pin once. Next put the lower 4 bit in the data bus and pulse the E pin again. Here is a flowchart simply describing it.

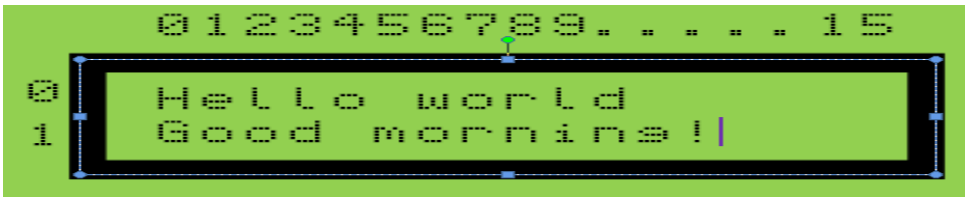
Flowchart of LCD Display Interfacing



Flow Chart of Interfacing LCD Display

8 BIT MODE

There is lot of stuff that can be done with the LCDs, to start with we will simple display a couple of strings on the 2 lines of the LCD as shown in the image.



Schematic description

- **Data Lines:** In this mode, all of the 8 data lines DB0 to DB7 are connected from the micrcontroller to a LCD module as shown the schematic.
- **Control Lines:'** The RS, RW and E are control lines, as discussed earlier.
- **Power & contrast**: Apart from that the LCD should be powered with 5V between **PIN 2(VCC)** and **PIN 1(gnd)**. **PIN 3** is the contrast pin and is output of center terminal of potentiometer (voltage divider) which varies voltage between 0 to 5v to vary the contrast.

- **Back-light:** The PIN 15 and 16 are used as backlight. The led backlight can be powered through a simple current limiting resistor as we do with normal leds.

4 BIT MODES
Schematic
There are following differences in 4 bit mode.

- Only data lines D4 to D7 are used as shown in the schematic below.
- In code, we need to send the command to select 4 bit mode as shown in the instruction set above.

The main program remains exactly as in 8 bit mode, we simply include the lcd_4_bit.c to work in 4 bit mode.

2.3 DUST SENSOR

Description:

Sharp's GP2Y1010AU0F is an optical air quality sensor, intended to detect tidy particles. An infrared producing diode and a phototransistor are askew organized into this gadget, to enable it to distinguish the reflected light of tidy in the air. It is particularly viable in distinguishing fine particles like tobacco smoke, and is ordinarily utilized as a part of air purifier frameworks.

The sensor has a low current utilization (20mA max, 11mA run of the mill), and can be powered with up to 7VDC.

Fig. of Dust Sensor

The o/p of the sensor is a simple voltage corresponding to the deliberate clean thickness, with a sensitivity of
0.5V/0.1mg/m3.

Features/Highlights of GP2Y1010AU0F

- Compact & thin bundle (46 × 30 × 17.6mm).
- With the utilization of pulse o/p framework, the gadget can identify even single house dust.
- House dust and tobacco smoke can be recognized.

Specifications

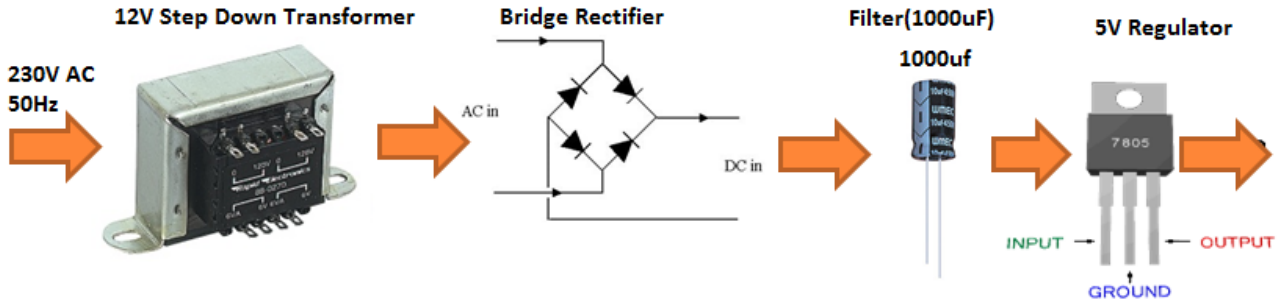
- Low Current Consumption (MAX: 20mA).
- Typical Operating Voltage: 4.5V to 5.5V (MAX: 7V).
- The nearness of dust can be distinguished by the photometry of just a single pulse.
- Enable to recognize smoke from house dust.
- Dimensions: 1.81 x 1.18 x 0.69'' (46.0 x 30.0 x 17.6mm).

2.4 FAN

In this project we are using exhaust fan, generally it is in computer CPU for its cooling purpose. Here we are using as exhaust fan, whenever dust detects it run automatically with help of Arduino microcontroller. The rotation of this exhaust around 23,000 rpm. (ie. 23,000 rotations per minute)

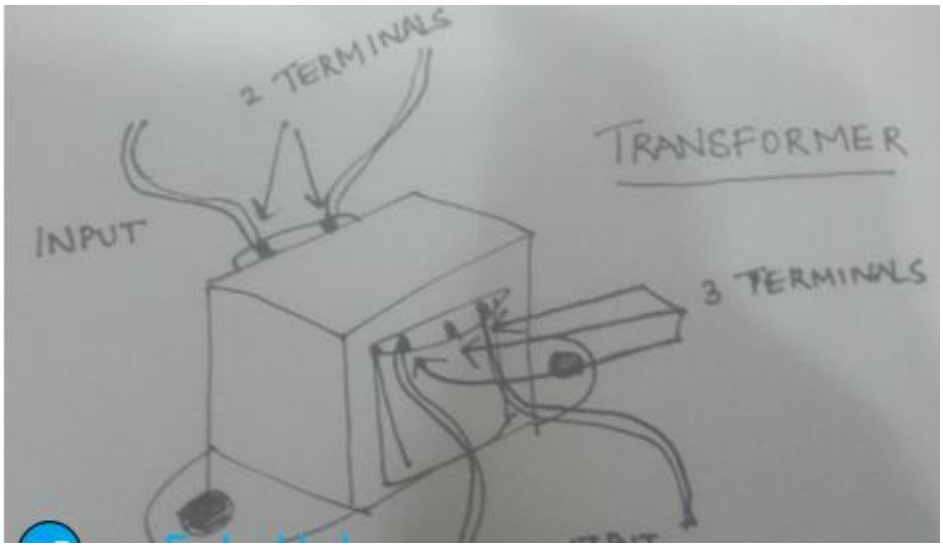
Exhaust Fan

POWER SUPPLIES:



2.5 TRANSFORMER

Transformers are devices which step down a relatively higher AC input Voltage into a lower AC output voltage. To find the input and output terminals of a transformer is very tricky. Refer to the following illustration or the internet to understand where what is.

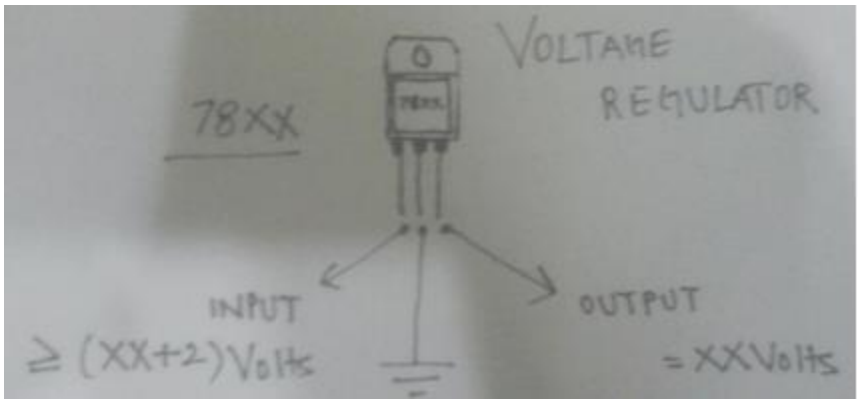


I/O Terminals of a Transformer

Basically, there are two sides in a transformer where the coil winding inside the transformer ends. Both ends have two wires each (unless you are using a center-tapped transformer for full wave rectification). On the transformer, one side will have three terminals and the other will have two. The one with the three terminals is the stepped down output of the transformer, and the one with the two terminals is where the input voltage is to be provided.

2.6 Voltage Regulators

The 78XX series of voltage regulators is a widely used range of regulators all over the world. The XX denotes the voltage that the regulator will regulate as output, from the input voltage. For instance, 7805, will regulate the voltage to 5V. Similarly, 7812 will regulate the voltage to 12V. The thing to remember with these voltage regulators is that they need at least 2 volts more than their output voltage as input. For instance, 7805 will need at least 7V, and 7812, at least 14 volts as inputs. This excess voltage which needs to be given to voltage regulators is called **Dropout Voltage.**



Voltage Regulator Schematic

2.7 Diode Bridge-Rectifier

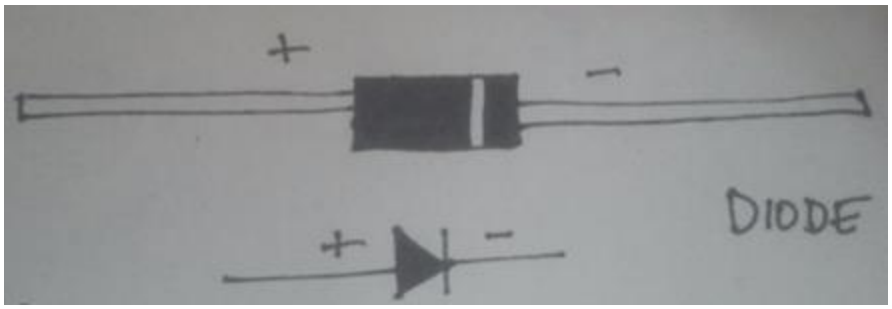
A bridge rectifier consists of an assembly of four ordinary diodes, by means of which we can convert AC Voltage into DC Voltage. It is found to be the best model for AC to DC conversion, over Full wave and Half wave rectifiers. You can use any model you want, but I use this for the sake of high efficiency (If you are using the full wave rectifier model, you'll need a center-tapped transformer, and you will only be able to use half of the transformed voltage).

One thing to note about diodes is that they drop about 0.7V each when operated in forward bias. So, in bridge rectification we will drop 1.4V because at one instant two diodes are conducting and each will drop 0.7V. In case of Full wave rectifier, only 0.7V will be dropped.

So how does this drop affect us? Well, this comes in handy while choosing the correct step down voltage for the transformer. See, our voltage regulator needs 2 Volts more than its output voltage. For the sake of explanation, let's assume that we are making a 12V adapter. So the voltage regulator needs at least 14 Volts as input.

So the output of the diodes (which goes into the voltage regulator) will have to be more than or equal to 14 Volts. Now for the diodes' input voltage. They'll drop 1.4 Volts in total, so the input to them has to be greater than or equal to 14.0 + 1.4 = 15.4Volts. So I would probably use a 220 to 18 Volt step down transformer for that.

So basically, the transformer step down voltage should be at least 3.4V more than the desired Power Supply output.



Schematic and Illustration of a Diode

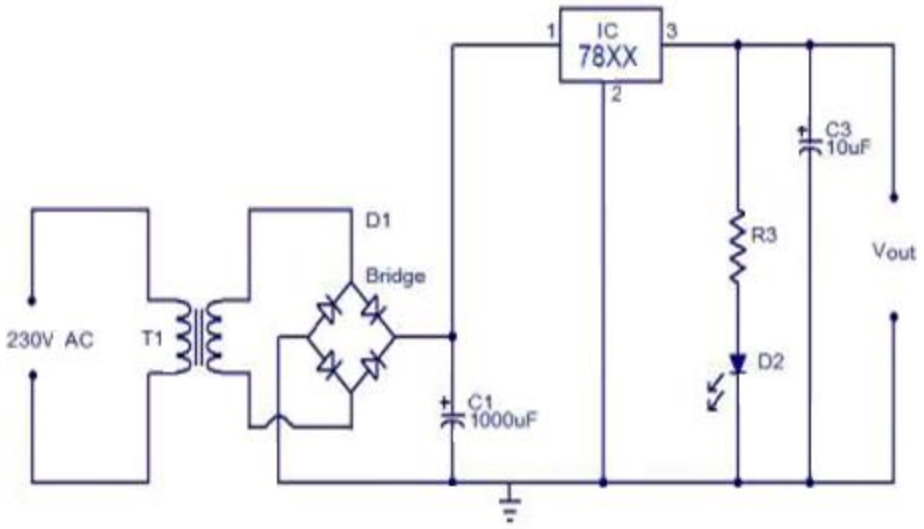
2.8 Filter Circuit

We filter, at the output of the voltage rectifier in order to get the smoothest DC Voltage as possible, from our adapter, for which we use capacitors. Capacitors are the simplest current filters available, they let AC current pass through and block DC, so they are used in parallel to the output. Furthermore, if there is a ripple in the input or output, a capacitor rectifies it by discharging the charge stored in it.

.

HOW TO BUILD IT?

Here is the circuit diagram for the Power Supply:



Circuit Diagram

How it works

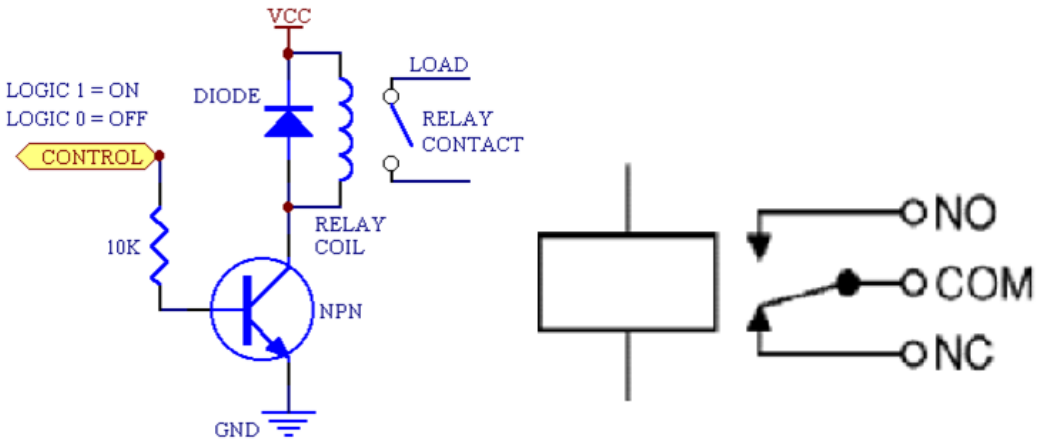
The AC mains are fed to the transformer, which steps down the 230 Volts to the desired voltage. The bridge rectifier follows the transformer thus converting AC voltage into a DC output and through a filtering capacitor feeds it directly into the input (Pin 1) of the voltage regulator. The common pin (Pin 2) of the voltage regulator is grounded. The output (Pin 3) of the voltage regulator is first filtered by a capacitor, and then the output is taken.

Make the circuit on a general purpose PCB and use a 2 Pin (5A) plug to connect the transformer input to the AC mains via insulated copper wires.

.

2.9 RELAY

It is an Electro Magnetic switch used to control the electrical devices, copper core magnetic flux plays main role here.



Relay Circuit Diagram

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

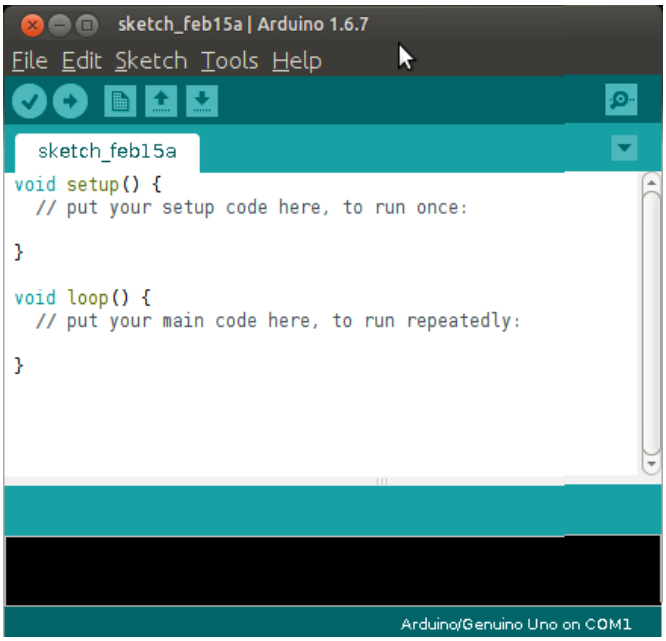
- **COM:** Common, always connect to this; it is the moving part of the itch.
- **NC:** normally Closed, COM is connected to this when the relay coil is off.
- **NO:** Normally Open, COM is connected to this when the relay coil is on.

CHAPTER-3

SOFTWARE REQUIREMENTS

ARDUINO IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a word processor for composing code, a message territory, a content reassures, a toolbar with catches for normal capacities and a progression of menus. It interfaces with the Arduino and Genuino equipment to transfer programs and Communicating with them.



Arduino IDE Window Layout

The Arduino IDE Environment having the tools **File, Edit, Sketch, Tools, Help**

- **File**
  Containing tools are New, Open, Open Recent, Sketchbook, Examples, Close, Save, Save as, Page Setup, Print, Preferences, Quit.
- **Edit**
  Containing tools are Undo/Redo, Cut, Copy, and Copy for Forum, Copy as HTML, Paste, Select All, Comment/Uncomment, Increase/Decrease Indent, Find, Find Next, and Find Previous.
- **Sketch**
  Containing tools are Verify/Compile, Upload, Upload Using Programmer, Export Compiled Binary, Show Sketch Folder, Include Library, and Add File.

- **Tools**
  Containing tools are Auto Format, Archive Sketch, Fix Encoding & Reload, Serial monitor, Board, Port, Programmer, Burn Boot loader.
- **Help**
  Containing tools are Getting Started, Reference.

Serial Monitor

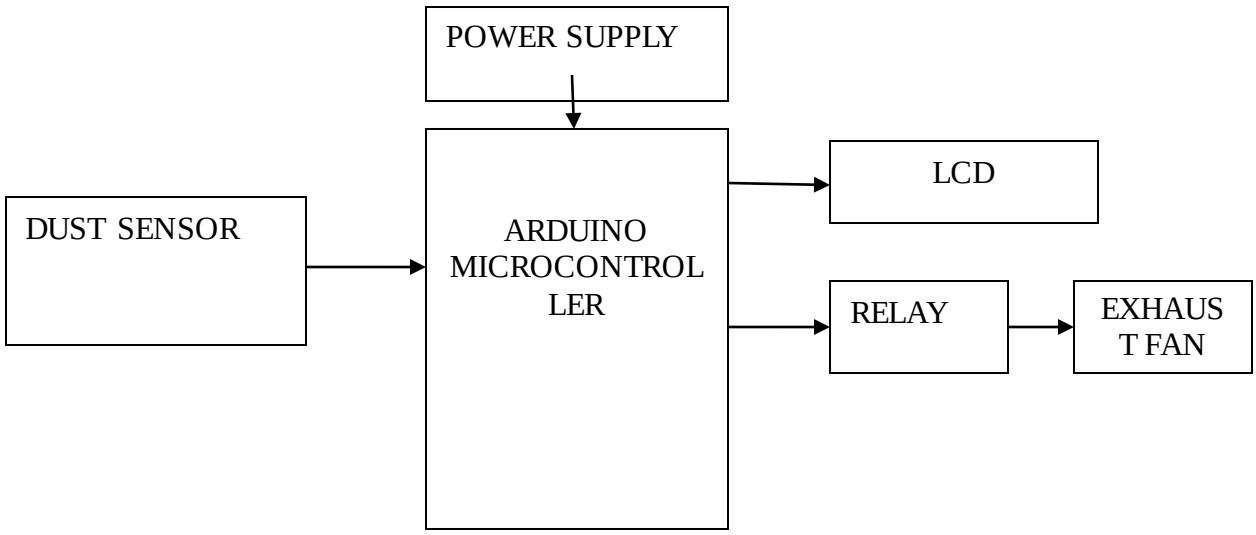
This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

.

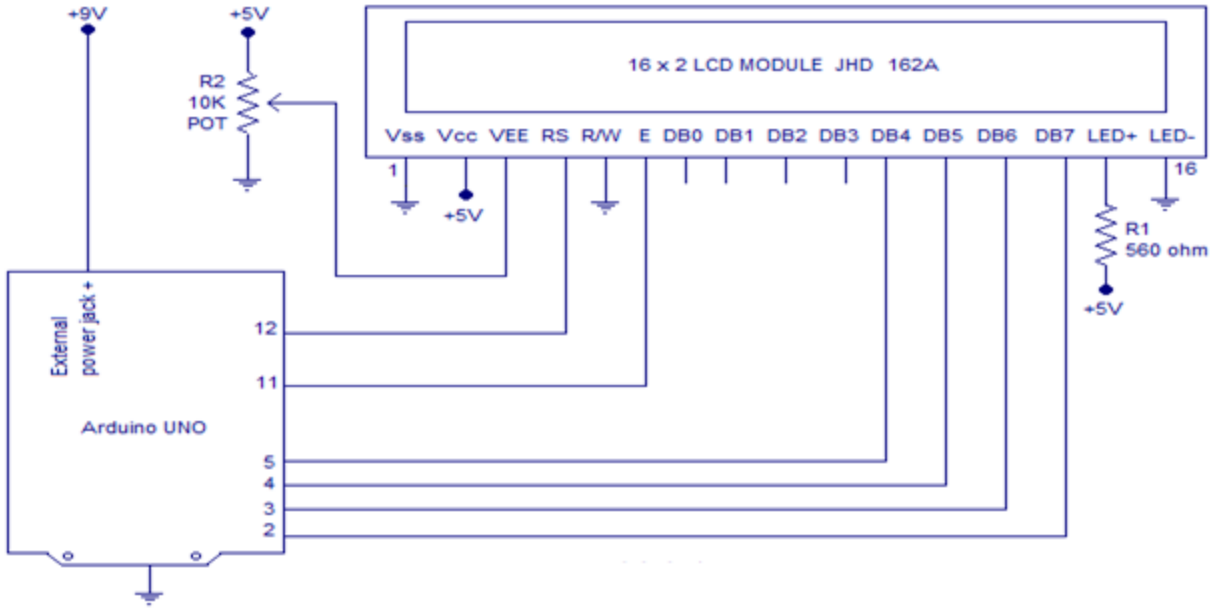
CHAPTER-4

BLOCK DIAGRAMS

4.1 Block Diagram of Automatic Exhaust with Dust Sensor



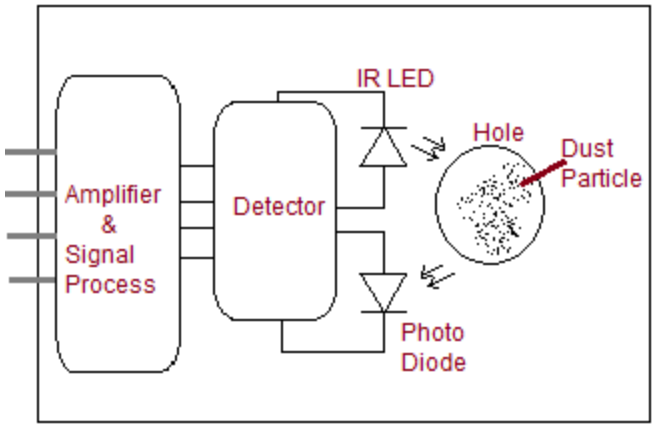
4.2 Block Diagram of Interfacing LCD and Arduino



CHAPTER-5

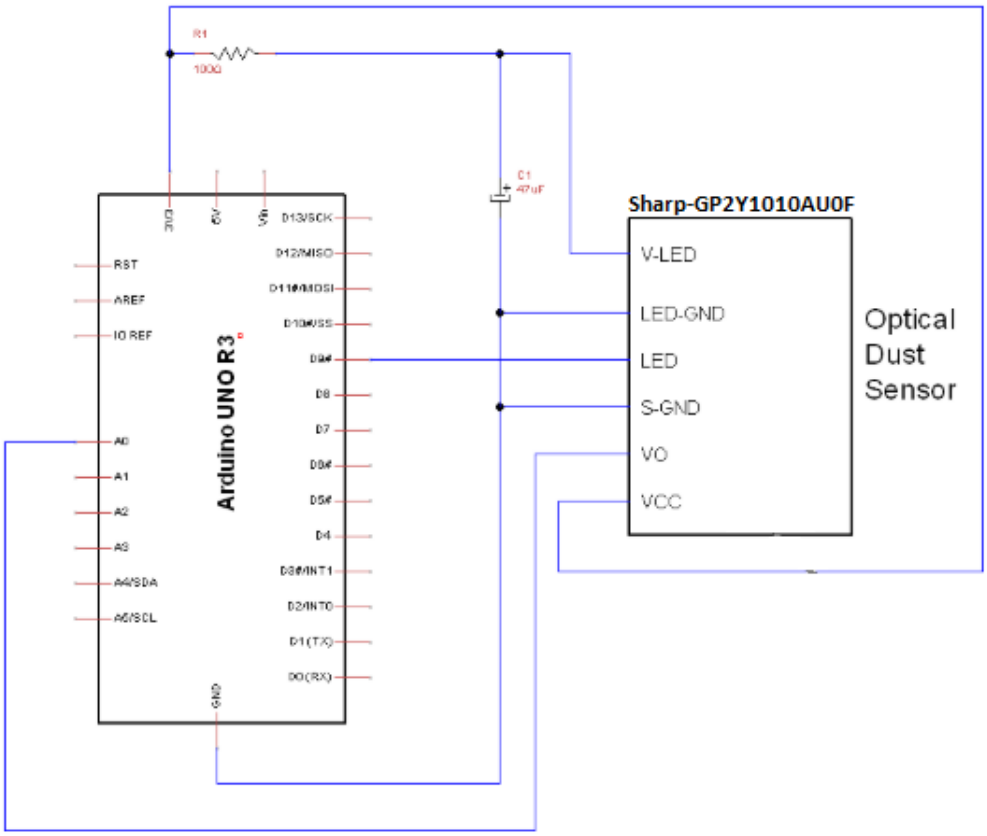
WORKING PROCEDURE

5.1 Dust Sensor Circuit Diagram & Explanation



The basic way of sensing dust is through photometry and this method uses IR LED emitter and photovoltaic diode (Photo diode) receiver, the sensor will have Air flow path and the IR LED and Photo diodes are placed near to the Air flow path and pulse signal will make IR LED to glow, due to the dust particles in Air it may scattered and received by photo diode, finally the signal is amplified and processed to get difference level and it can be measured as dust density level.

5.2 Arduino Dust Sensor Interfacing



Connect the dust sensor with Arduino board as illustrated, here the 3.3V from Arduino board is given to Sensor VCC pin and V-LED through 100Ω Resistor, LED-GND and S-GND pins are commonly connected to Arduino Ground pin. Digital pin D9 is connected with LED pin and it is declared as output pin, Vo pin gives Analog output hence it is connected to the Arduino Analog input pin A0.

5.3 Coding

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);
int ledPower = 8;
int fan=9;
int sam = 280,s;
int delta = 40;
int sleep = 9680;
float voMeasured = 0;
float calcVoltage = 0;

void setup(){
Serial.begin(9600);
pinMode(ledPower,OUTPUT);
pinMode(fan,OUTPUT);
lcd.begin(16,2);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("air pollution");
lcd.setCursor(0,1);
lcd.print ("Monitoring");
digitalWrite(fan,LOW);
}
void loop()
{
digitalWrite(ledPower,LOW); // power on the LED
delayMicroseconds(sam);
voMeasured = analogRead(A0); // read the dust value
delayMicroseconds(delta);
digitalWrite(ledPower,HIGH); // turn the LED off
delayMicroseconds(sleep);
Serial.print("Raw Signal Value (0-1023): ");
Serial.println(voMeasured);
if (voMeasured>10)
{
s=map(voMeasured,0,35,0,100);
Serial.print(" calcVoltage");
Serial.println( calcVoltage);
Serial.print("dust content: ");
Serial.println(s);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Dust content:");
```

```
lcd.setCursor(14,0);
lcd.print(s);
delay(500);
if(s>80)
{
Serial.println("dust content increasing....");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Dust content:");
lcd.setCursor(14,0);
lcd.print(s);
lcd.setCursor(0,1);
lcd.print("Fan on....");
digitalWrite(fan,HIGH);
delay(10000);
s=35;
}
}
if(s<80)
{
digitalWrite(fan,LOW);
}
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Dust content:");
lcd.setCursor(14,0);
lcd.print(s);
delay(1000);
}
```

CHAPTER-6

APPLICATION & ADVANTAGES

Applications

This is very important at several places like:

- Semiconductor manufacturing Industries,
- Manufacturing Industries,
- Medical research Labs,

And some more applications:

- Detection for dust in air, indoor air quality monitoring.
- Air conditioner.
- Air purifier.
- Air Cleaner.
- Smoke type fire alarm application by different sensor adjustments.

Advantages

- It avoids energy consumption as the exhaust runs only when it is necessary to run.
- Low cost.
- Less energy consumption.
- Easy to maintain.
- Construction is Easy.

CHAPTER-7
CONCLUSION

Based on the dust content existing in the surroundings if it is high it switches on exhaust fan else it switches off exhaust fan.This project is very useful in manufacturing engineering (like semiconductor, chip designing and medical research etc.). The system can be future upgraded and can be connected with an alarm system if necessary. It can have a GSM system to intimate the user remotely.

CHAPTER-8
REFERENCES

- "http://dspace.ewubd.edu/bitstream/handle/123456789/2589/Farhana_%20Sultana.pdf?
- "http://www.labsguru.com/kits_internal.php?product=New%20Projects"
- "http://docplayer.net/26684624-Lecture-6-introduction-to-the-atmega328-and-ardunio-cse-p567.html"
- "http://ijartet.com/3434/v5s09mar2018idhaya/conference"
- "https://electronicsforu.com/resources/learn-electronics/16x2-lcd-pinout-diagram"
- "http://arduinodev.woofex.net/2012/12/01/standalone-sharp-dust-sensor/"