# IMPLEMENTATION METHODOLOGY

## Hardware Implementation

* Hardware Implementation describes the process by which the Analog Probe Test Kit is replaced by Automatic Fuel Probe Test Kit.
* Analog Probe Test Kit, which has a power supply for both 28volts and 5volts, CRO for measuring the time period and a test kit for giving 28volts and 5volts to the fuel probe and taking Frequency signal from fuel probe.
* The Automatic Fuel Probe Test Kit is powered by 220V AC. Test kit internal circuit generates 5Vand 28V for powering microcontrollers, LCD, relays and for probe excitation. It also has an LCD display for displaying the height of fuel and corresponding time period. It is connected to motor via two 230V/5V relays. This relay is controlled by the control signals from microcontroller thus implementing PID control of height of fuel in the tank by suitable program.
* Microcontroller is connected to a computer through a ‘COM’ port. An application is made for sending calibration chart in to the controller and saving height – timeperiod metrology of the probe that is being tested.
* The microcontroller used is Arduino Uno (ATMEGA 328P)

## Automated Fuel Probe Test Kit

The kit consists of

* Fuel tank with Fuel probe
* Three phase Motor
* Controller kit with LCD display
* Computer

## List of Components

* Two Arduino Uno Board
* One 28V/5V Relay
* Two Channel 230V/5V Relay Module
* LCD Display (16x2)
* Rotary Switch
* Buzzer
* Five DPDT Switches
* D-type Male connector
* Different resistors

## POWER SUPPLY

Automatic Fuel Probe Test Kit has a Power cable socket where you plug the power cable from 220v AC. The Test Kit has two AC to DC converters. One is 230V/5V convertor and the other one is 230V/28V convertor. 5V supply is used to power Arduinos (microcontrollers), LCD, relays and for probe excitation.28V is for excitation of Low-Level Sensor in fuel probe.

## LCD DISPLAY

LCD, an acronym for Liquid Crystal Display revolutionized the modern display technology with its compactness and versatility. Today it is seen embedded in various electronic gadgets and devices like T.V., Computers, Laptops, Watches, etc. A Liquid crystal coating is the heart of the display which is sandwiched between two polarized glasses.

LCD’s are available in various shapes and sizes depending on the configurations. A [16x2 LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet) shown in the image below can display 32 characters with 16 characters in each row. It is capable of displaying any character with ASCII values ranging from 0 to 255.

****

****

The Liquid Crystal Library allows you to control LCD Displays that are compatible with the **JHD204A** Driver.There are many of them out there , and you can usually tell them by the 16pin interface. The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins.

****

Character Lcd's have a controller built into them named HD44780. We actually talk with this controller in order to display a character on the LCD screen. HD44780 must be properly handled and initialized before sending any data to it. HD44780 has some registers which are initialized and manipulated for character displaying on the LCD. These registers are selected by the pins of c Rs(Register select)

Register select selects the HD44780 controller registers. It switches between Command and data register.

• Command Register

• Data Register

**Command Register**:

When we send commands to LCD these commands go to Command register and are process them. Commands with their full description are given in the picture below. When Rs=0 command register is selected.

Data Register:

When we send Data to LCD it goes to the data register and is processed there. When Rs=1 data register is selected.

**R/W(Read - Write):**

R/W pin is used to read and write data to HD44780 data and command registers. When R/W=1 we can read data from LCD. When R/W=0 we can write to LCD.

**En(Enable signal):**

When we select the register Rs(Command and Data) and set Rw(read - write) and placed the raw value on 8-data lines, now it’s time to execute the instruction. By instruction I mean the 8-bit data or 8-bit command present on Data lines of LCD. For sending the final data/command present on the data lines we use this enable pin. Usually, it remains en=0 and when we want to execute the instruction we make it high en=1 for some mills seconds. After this we again make it ground en=0.

**VE (Set Lcd contrast):**

To set LCD display sharpness use this pin. The best way is to use variable resistor such as potentiometer a variable current makes the character contrast sharp. Connect the output of the potentiometer to this pin. Rotate the potentiometer knob forward and backward to adjust the LCD contrast.

There are power supply pins(+5v and Gnd) and LCD Backlight(BKLt+ and BKLt-)that you can use to power the LCD, and turn on and off the LCD backlight.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register.The Liquid Crystal Library simplifies this for you so you don’t need to know the low-level instructions.The JHD204A LCDs can be controlled in two modes:4bit or 8bit.The 4bit mode requires seven I/O pins from the Arduino, while the 8bit mode requires 11 pins.For displaying text on the screen, you can do everything in 4bit mode.

**FEATURES**

• 5\*8 dots with cursor

• Built-in controller(HD44780)

• +5V power supply (Also available for +3V)

• 1/16 duty cycle

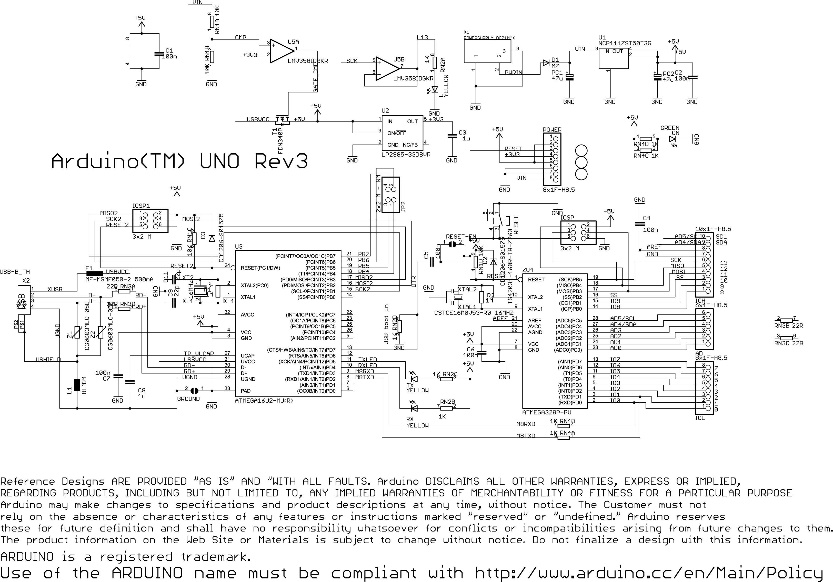
**LCD PINS AND ARDUINO CONNECTIONS**

|  |  |
| --- | --- |
| LCD PIN | ARDUINO PIN |
| 1. VSS | GND |
| 1. VDD | 5v |
| 1. Contrast | 3.3V |
| 1. RS | 3 |
| 1. R/W | GND |
| 1. Enable | 4 |
| 1. D7 | 9 |
| 1. D6 | 10 |
| 1. D5 | **11** |
| 1. D4 | 12 |
| 1. D3 | No connection |
| 1. D2 | No connection |
| 1. D1 | No connection |
| 1. D0 | No connection |
| 1. LCD +ve | 3.8V |
| 1. LCD-ve | **GND** |

**Interfacing Pin Configuration**

**Arduino Microcontroller**

Arduino is a single-board microcontroller to make using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.





**Hardware Specifications**

• Microcontroller: ATmega328

• Operating Voltage: 5V

• Input Voltage (recommended):7-12V

• Input Voltage (limits): 620V

• Digital I/O Pins: 14 (of which 6 provide PWM output)

• Analog Input Pins: 6

• DC Current per I/O Pin: 40 mA

• DC Current for 3.3V Pin: 50 mA

• Flash Memory: 32 KB (ATmega328)

• SRAM: 2 KB (ATmega328)

• EEPROM: 1 KB (ATmega328)

• Clock Speed: 16 MHz

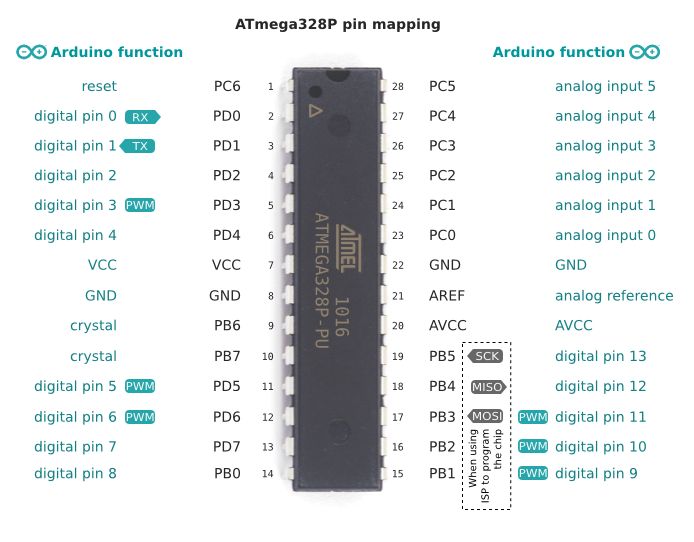
An Open source design:-The advantage of being an open source is that it has a large community of people using and troubleshooting it.This makes it easy to find someone to help you to debug your project .

An ICSP connector for bypassing the USB port and interfacing the arduino directly as a serial device .This port is necessary to re-boot load your chip if it corrupts &can no longer talk to your computer.

An on-board LED attached to digital pin 13 for the fast and easy debugging of code &last ,but not the least , a button to reset the program on the chip .

An Easy USB interface:-The chip on the board plugs straight into your USB port &registers on your computer as a virtual serial port .This allows you to interface with it were a serial device .The benefit of this setup is that serial communication is an extremely easily (&time tested ) protocol &USB makes connecting it to the modern computers really convinent.

Very convenient power management & built in voltage regulation, you can connect an external power source up to 12v and it will be regulate it to both 5v and 3.3 v.It also can be powered directly of USB port without external power.



**Digital Pins**

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

* Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
* 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. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
* BT Reset: 7. (Arduino BT-only) Connected to the reset line of the bluetooth module.
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
* LED: 13. On the Diecimila and LilyPad, 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.

**Analog Pins**

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

* I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

**Power Pins**

* VIN (sometimes labelled "9V"). 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. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
* 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply
* 3V3. (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
* GND. Ground pins.

**Other Pins**

* AREF. Reference voltage for the analog inputs. Used with analogReference().
* Reset. (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board

# SOFTWARE IMPLEMENTATION

## Procedure to execute the program

This document explains how to connect your Uno board to the computer and upload your first sketch. The Arduino Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards and running both online and offline.

If you want to program your Arduino/Genuino Uno while offline you need to install the Arduino Desktop IDE. The Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards. Before you can move on, you must have installed the Arduino Software (IDE) on your PC.

Connect your Uno board with an A B USB cable; sometimes this cable is called a USB printer cable.

The USB connection with the PC is necessary to program the board and not just to power it up. The Uno automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power LED (labelled PWR) should go on.

**Install the board drivers**

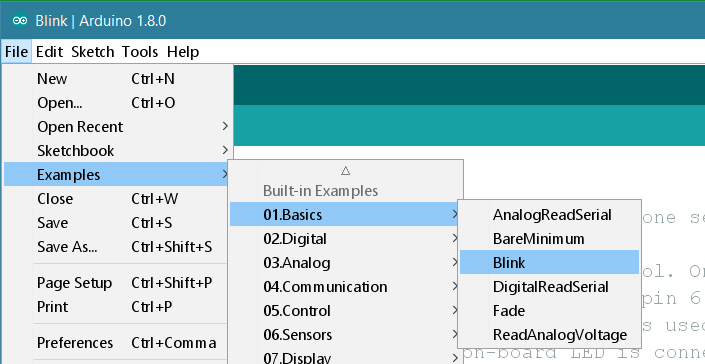
If you used the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board.

If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below.

* Click on the Start Menu, and open up the Control Panel.
* While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager
* Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
* Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
* Next, choose the "Browse my computer for Driver software" option.
* Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf"
* Windows will finish up the driver installation from there.

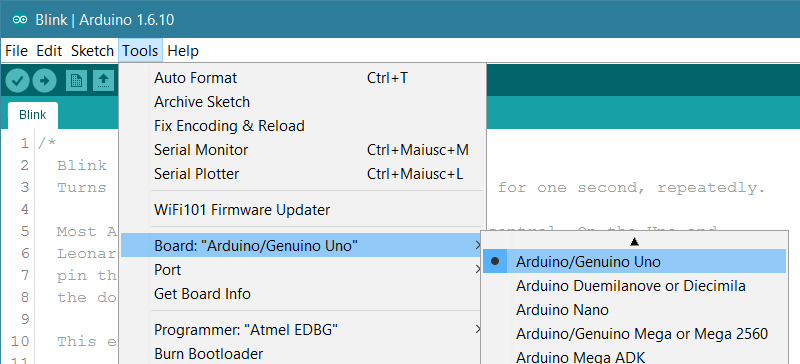
**Open your first sketch**

Open the LED blink example sketch: File > Examples >01.Basics > Blink.



**Select your board type and port**

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.



**Upload the program**

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

