CHAPTER 1
INTRODUCTION

In today’s world, women safety has become a major issue as they can’t step out of their house at any given time due to physical/sexual abuse and a fear of violence. Even in the 21st century where the technology is rapidly growing and new gadgets were developed but still women’s and girls are facing problems. Women are adept at mobilizing diverse groups for a common reason. They often work across ethnic, religious, political, and cultural divides to promote liberty. We are all aware of importance of women safety, but we must analyze that they should be properly protected. Women are not as physically fit as men, in an emergency situation a helping hand would be assistance for them. The best way to cut tail your probability of becoming a dupe of violent crime (robbery, sexual assault, rape, domestic violence) is to recognize, defense and look up resources to help you out of hazardous situation. If you're in dilemma or get split from friends during a night out and don't know how to find back residence, this device with you will guard you and can reduce your risk and bring assistance when you need it.

1.1 EMBEDDED SYSTEM IMPLEMENTATION:

1.1.1 Introduction:

An embedded system is one kind of a computer system mainly designed to perform several tasks like to access, process, 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 neighborhood traffic control systems, etc.

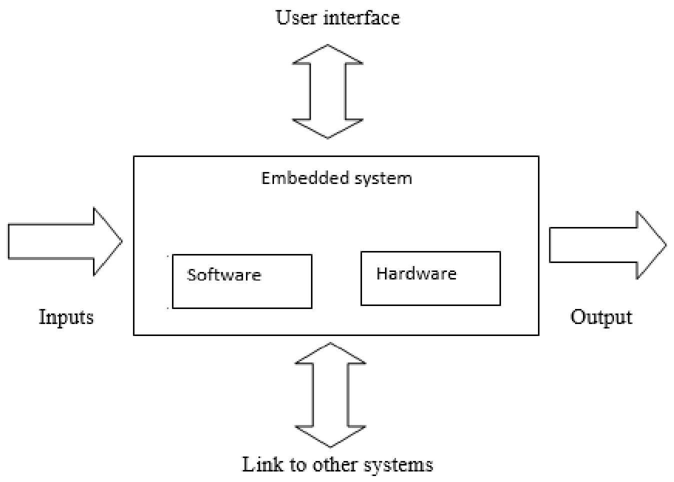


Fig 1.1: Overview of embedded system

1.1.2 Embedded System:

Embedded system includes mainly two sections, they are

1. Hardware
2. Software

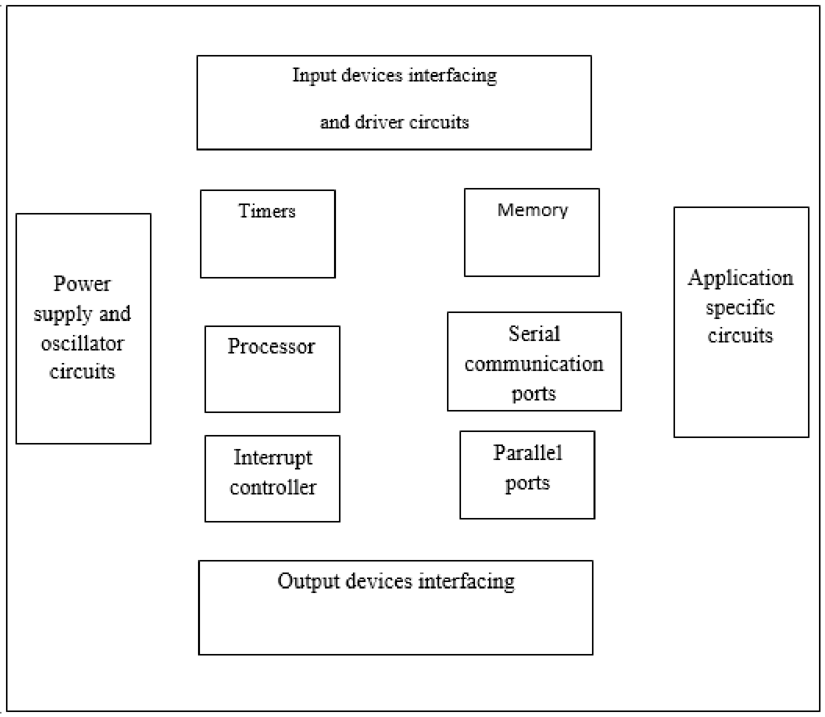

Input devices interfacing and driver circuits
Timers
Memory
Power supply and oscillator circuits
Application specific circuits
Processor
Serial communication ports
Interrupt controller
Parallel ports
Output devices interfacing

Fig 1.2: Overview of embedded system block diagram

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. The 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.

1.1.3 Bringing Software and Hardware together for Embedded System:

To make software to work with embedded systems we need to bring software and hardware togetherfor 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 asingle 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.

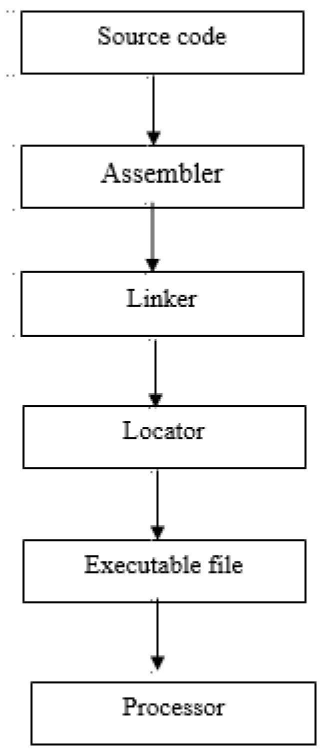


Fig1.3: Flow of burning source code to processor

1.2 APPLICATIONS:

Embedded systems have different applications. A few select applications of embedded systems are smart cards, telecommunications, satellites, missiles, digital consumer electronics, computer networking, etc.

Embedded Systems in Automobiles

- Motor Control System
- Engine or Body Safety
- Robotics in Assembly Line
- Mobile and E-Com Access

Embedded systems in Telecommunications

- Mobile computing
- Networking
- Wireless Communications

Embedded Systems in Smart Cards

- Banking
- Telephone
- Security Systems

1.3. 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.

CHAPTER 2
LITERATURE SURVEY

2.1EXISTING SYSTEM:

In previous system the alert system for the women is done through the application. For the security purpose the applications contain the SOS number which will alert the family members of the victim.

2.1.1 Disadvantages of Existing System:

- Victims phone may get lost.
- Battery may die

2.2 PROPOSED SYSTEM:

Proposed system increases the security features of the device by integrating a fingerprint module with the Arduino microcontroller. The proposed design uses GSM/GPS based module to track the device and to provide access to a person in a remote location. The design involves incorporation of a fingerprint identification module.

2.2.1. Block Diagram:

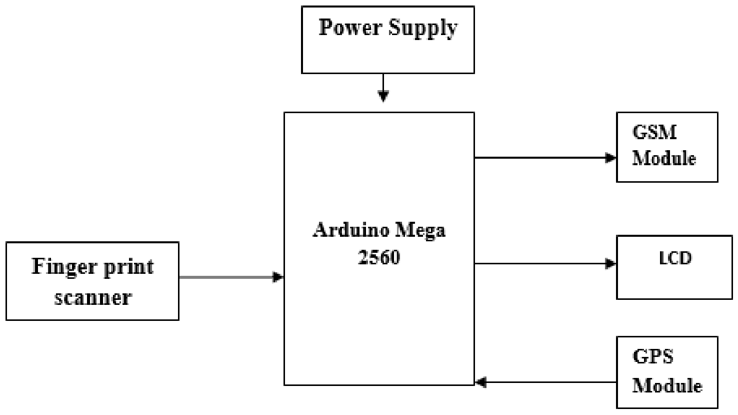


Fig2.1: Block diagram of the proposed system

CHAPTER 3
HARDWARE REQUIREMENTS

3.1 ARDUINO MEGA 2560:

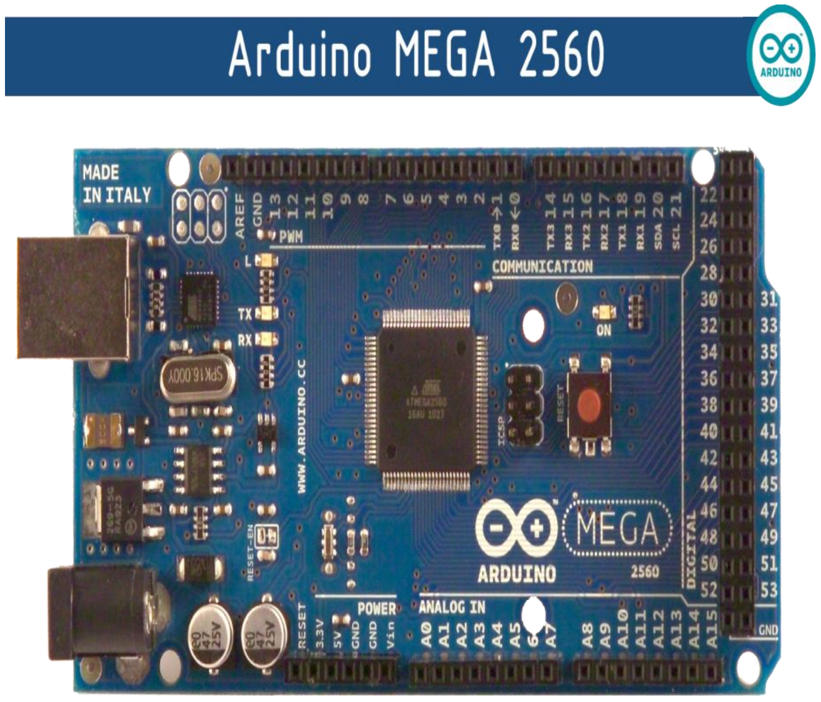


Fig3.1: Arduino MEGA 2560

3.1.1. What is Arduino Mega?

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

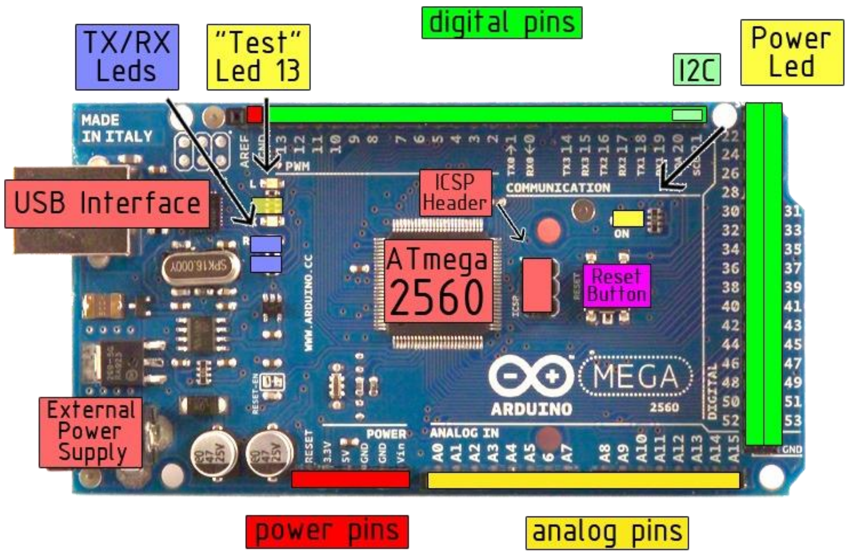


Fig : Arduino mega pin details

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).
- 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. A 3.3volt supply generated by the on-board regulator.
- GND. Ground pins.

3.1.2. Memory:

The ATmega2560 has 256 KB of flash memory for storing code and 8 KB of SRAM and 4 KB of EEPROM.

3.1.3. Pin Description:

- Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite() and digitalRead() functions. They operate at 5 volts. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). 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 attach Interrupt() function for details.
- PWM: 0 to 13. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- 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.
- I2C: 20 (SDA) and 21 (SCL). Support I2C (TWI) communication using the Wirelibrary(documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove.
- The Mega2560 has 16 analog inputs, 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 analogReference() function.
- 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.

Programming:

The Mega 2560 board can be programmed with the Arduino Software (IDE). For details, see the reference and tutorials.The ATmega2560 on the Mega 2560 comes preprogrammed 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 using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. 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.

Warnings

The Mega 2560 has a resettable poly-fuse 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.

Power

The Mega 2560 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 become 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 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.
- **3.3V:** A 3.3volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND:** Ground pins.
- **IOREF:** This pin on the board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Input and Output:

Each of the 54 digital pins on the Mega 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 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- **Serial:** 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- **External Interrupts:** 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the attachInterrupt() function for details.
- **PWM:** 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.
- **SPI:** 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino /Genuino Uno and the old Duemilanove and Diecimila Arduino boards.
- **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.
- **TWI:** 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.

The Mega 2560 has 16 analog inputs, 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 analogReference() function is used.

The following are the pin description of Arduino mega:

- **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.

Communication:

The Mega 2560 board has a number of facilities for communicating with a computer, another board, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega8U2) on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 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 Mega 2560's digital pins.

The Mega 2560 also supports TWI and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the TWI bus; see the documentation for details. For SPI communication, use the SPI library. Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega 2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three 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 even multiple of the 100mil spacing of the other pins.

The Mega 2560 is designed to be compatible with most shields designed for the Uno and the older Diecimila or Duemilanove Arduino boards. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Furthermore, the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega 2560 and Duemilanove / Diecimila boards. Please note that I2C is not located on the same pins on the Mega 2560 board (20 and 21) as the Duemilanove / Diecimila boards (analog inputs 4 and 5).

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Mega 2560 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 is connected to the reset line of the ATmega2560 via a 100nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) 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 Mega 2560 board 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 half-second or so, the boot loader is running on the ATMega2560. 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 Mega 2560 board 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.

Revisions

The Mega 2560 does not use the FTDI USB-to-serial driver chip used in past designs. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 Arduino boards) programmed as a USB-to-serial converter.Revision 2 of the Mega

2560 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.Revision 3 of the Arduino board and the current Genuino Mega 2560 have the following improved features:

- **1.0 pinout**: SDA and SCL pins - near to the AREF pin - and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the board that uses ATSAM3X8E, that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2

3.2 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.

3.2.1. 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.

3.2.2. Pin Diagram:

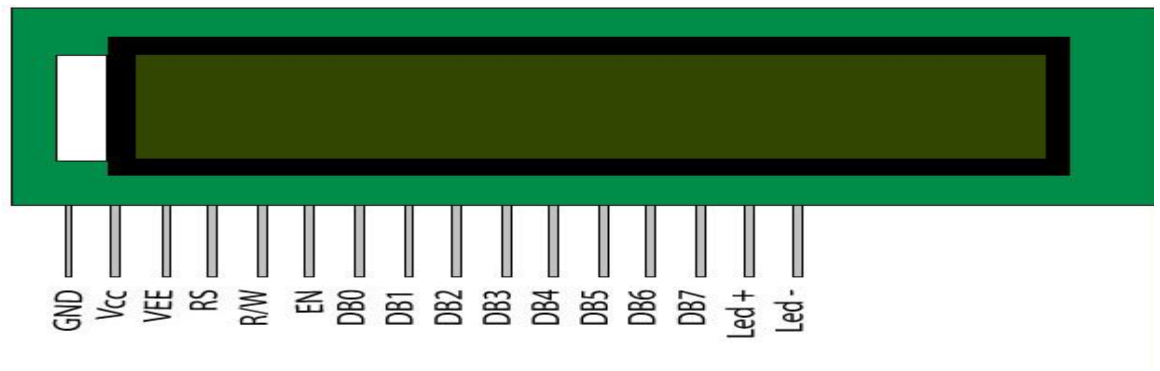


Fig: Pin diagram of LCD

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	V_{EE}
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 V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Table 3.1: Pin description of LCD

3.2.3 Block Diagram of LCD Display:

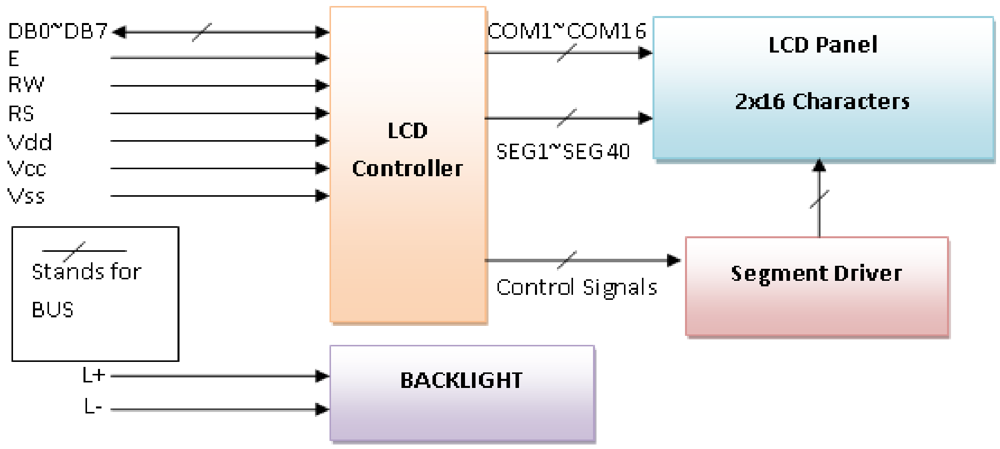


Fig3.2: Block diagram of LCD display

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

Table 3.2: Control and Display commands

Ac -Address Counter

Outline

Now the instruction can be divided mainly in four kinds

1) Function set instructions
2) Address set instructions
3) Data transfer instructions with internal RAM
4) Others

3.2.7. Details Of The Instructions

1) Read Data from RAM

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

Table 3.3: Read data from RAM

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) Write data to RAM

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

Table 3.4: Write data to RAM

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.

3) Read 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

Table 3.5: Read Busy Flag and Address

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.

4) Set DDRAM Address

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

Table 3.6: Set DDRAM Address

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”.

5) Set CGRAM address

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

Table 3.7: Set CGRAM address

Set CGRAM address to AC. This instruction makes CGRAM data available from MPU.

6) Function Set

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

Table 3.8: Function set

DL: Interface data length control bit

DL=’1’ means 8bit mode of data transfer.

DL=’0’ means 4bit mode of data transfer

When 4-bit mode is activated, the data needs to be transferred in two parts, first higher 4bits, and then lower 4 bits.

N: display line number control bit

N=’1’ will allows to characters to display in 2-lines

N=’0’ will allows to characters to display in the first line only

F: display font control bit

F=’0’ will use 5×8 dots format display mode

F=’1’ will use 5×11 dots format display mode

7) Cursor or display Shift

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

Table 3.9: Cursor or display shift

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.

8) 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

Table 3.10: Display On/Off Control

This instruction controls Display, Cursor and cursor blink.

D: Display On/Off control bit
D=’1’ means entire display is turned on
D=’0’ means entire display is turned off. But Display data remains in DDRAM.

C=’1’ turns on the cursor

C=’0’ turns off the cursor. But I/D register retains the data

B: Cursor blink On/Off control bit

B=’1’ makes cursor blink periodically.

B=’0’ stops the cursor to blink and cursor looks steady if the Cursor is turned on.

9) 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

Table 3.11: Entry mode set

This instruction sets the moving direction of cursor and display.

When I/D= ’1’ cursor moves to the right and DDRAM address is increased by 1.

When I/D= ’0’ cursor moves to the left and DDRAM address is decreased by 1.

10) Return Home

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

Table 3.12: Return Home

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.

11) Clear display

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

Table 3.13: Clear display

3.2.8. 8-BIT AND 4-BIT INTERFACING OF LCD

LCD Display Interfacing – Flowchart:-

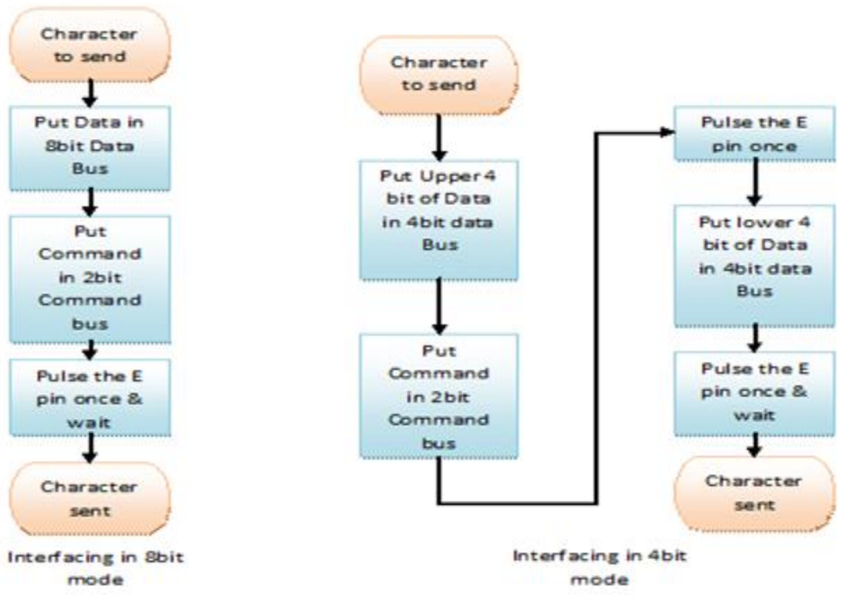

Character to send
Put Data in 8bit Data Bus
Put Command in 2bit Command bus
Pulse the E pin once & wait
Character sent
Interfacing in 8bit mode
Character to send
Put Upper 4 bit of Data in 4bit data Bus
Put Command in 2bit Command bus
Pulse the E pin once
Put lower 4 bit of Data in 4bit data Bus
Pulse the E pin once & wait
Character sent
Interfacing in 4bit mode

Fig : Flow chart of LCD display interfacing

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.

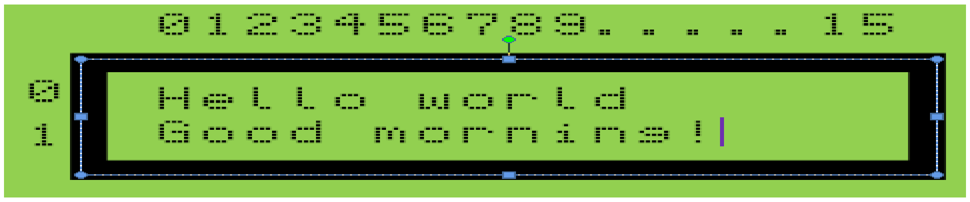


Fig : 8-bit mode of LCD display

Schematic description

- **Data Lines:** In this mode, all of the 8 data-lines DB0 to DB7 are connected from the microcontroller 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.

3.3. POWER SUPPLY

3.3.1 What is A Power Supply?

A power supply is a device which delivers an exact voltage to another device as per its needs.There are many power supplies available today in the market like regulated, unregulated, variable etc., and the decision to pick the correct one depends entirely on what device you are trying to operate with the power supply. Power supplies, often called power adapters, or simply adapters, are available in various voltages, with varying current capacities, which is nothing but the maximum capacity of a power supply to deliver current to a load (Load is the device you are trying to supply power to).

Why Make One Yourself?

One would ask himself, “Why do I make it myself, when it available in the market?” Well, the answer is- even if you buy one, it is bound to stop working in a while (and believe me, power supplies stop working without any prior indication, one day they'll work, the next day they'll just stop working!). So, if you build one yourself, you will always know how to repair it, as you will know exactly what component/part of the circuit is doing what. And further, knowing how to build one, will allow you to repair the ones you have already bought, without wasting your money on a new one.

What You Need?

1. Copper wires, with at least 1A current carrying capacity for AC mains
2. Step Down Transformer
3. 1N4007 Silica Diodes (×4)
4. 1000µF Capacitor
5. 10µF Capacitor
6. Voltage regulator (78XX) (XX is the output voltage reqd. I'll explain this concept later)
7. Soldering iron
8. Solder
9. General Purpose PCB
10. Adapter jack (to provide the output voltage to a device with a particular socket)

Optional

a. LED (for indication)
b. Resistor (Value explained later)
c. Heat Sink for The Voltage Regulator (For higher current outputs)
d. SPST Switch

Some Basic Concepts of Power Supplies

Transformers

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.

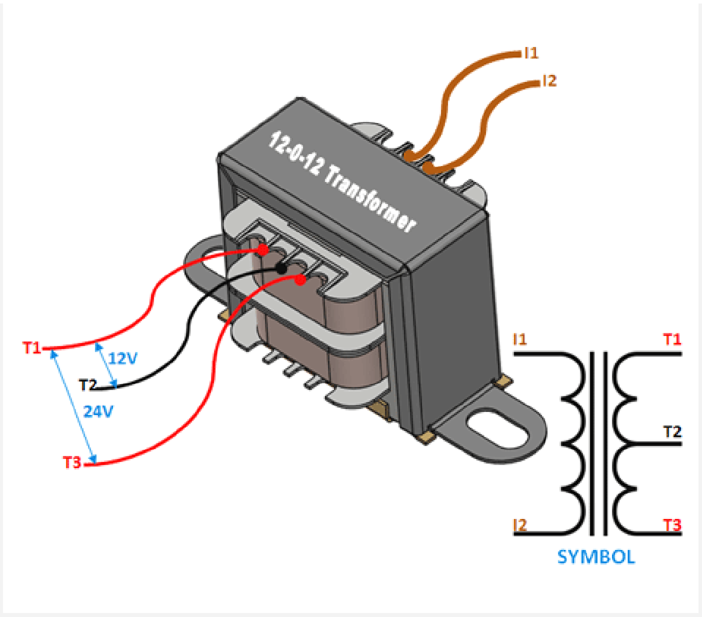


Fig : I/O Terminals of a Transformer

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.

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**.

NOTE: The input pin is denoted as '1', ground as '2' and output as '3'.

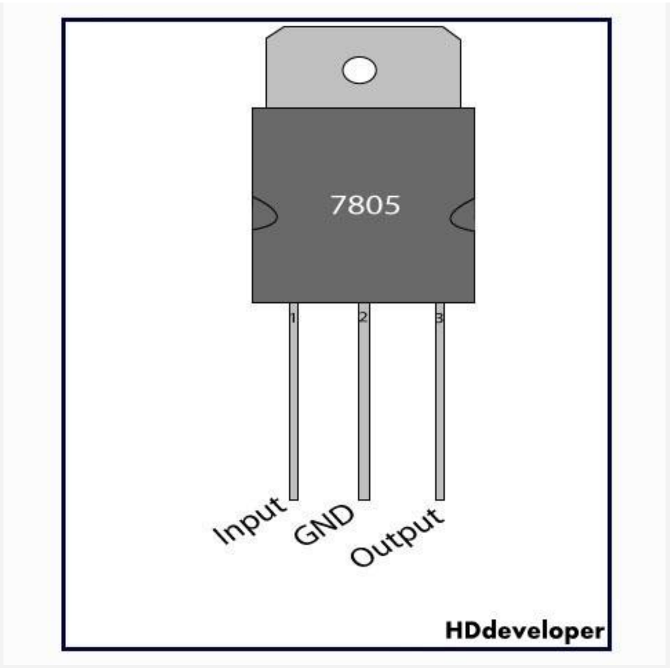


Fig : Voltage Regulator Schematic

Diode Bridge

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.

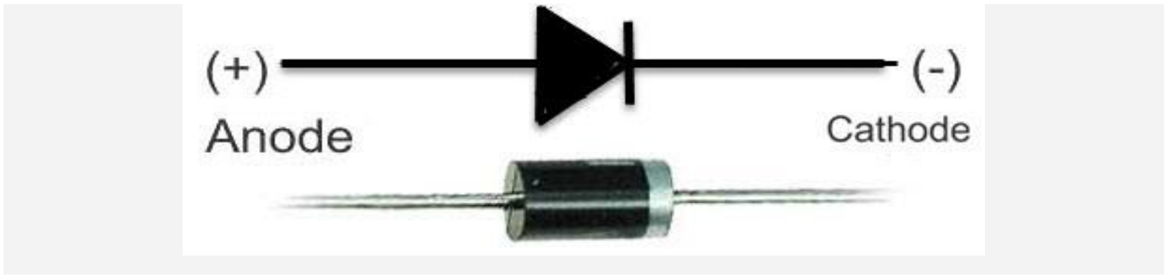


Fig : Schematic and Illustration of a Diode

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.

How To Build It?

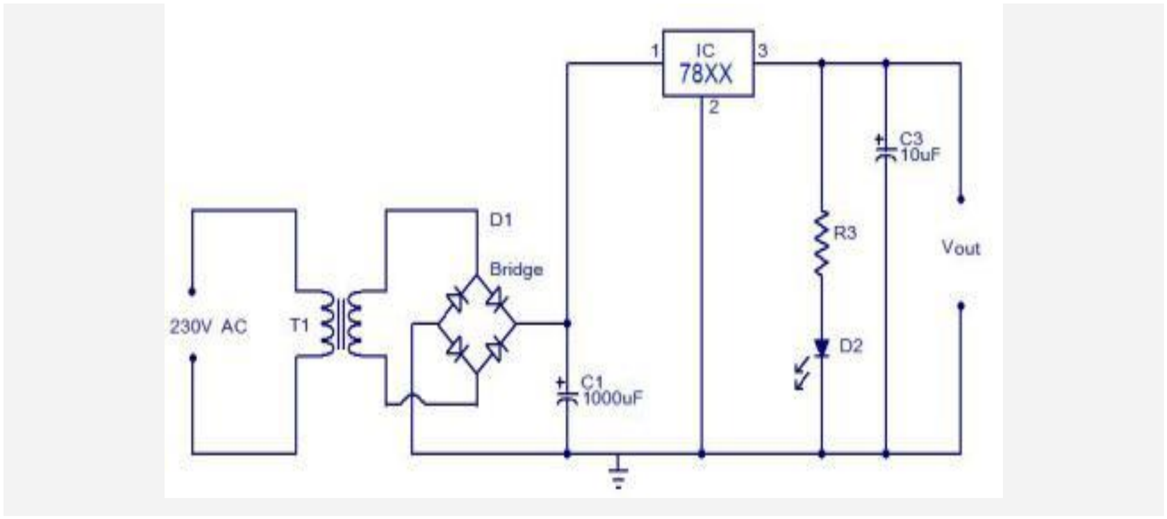


Fig 3.3: Power supply Circuit Diagram

3.3.2 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.

3.4FINGER-PRINT MODULE

3.4.1 General View

Fingerprint enrollment, image process, characters acquisition, fingerprint template creation, fingerprint template storage, fingerprint compare (1: 1, 1: N), fingerprint delete. This module can work with different devices based on UAWRT such as PC, SCM and so on. Only easy circuits and fingerprint module can enhance your product into fingerprint authentication power. It is widely used by electronics business, information security, access control, identity authentication and other security industry.

Fig 3.4: Fingerprint module

Application Solution

1. When FPRS is embedded into your system, the other functions will be controlled by MCU Controller, so developer can realize his own function logic, user interface and communication port through hard ware and soft ware development, such as fingerprint time and attendance and so on.

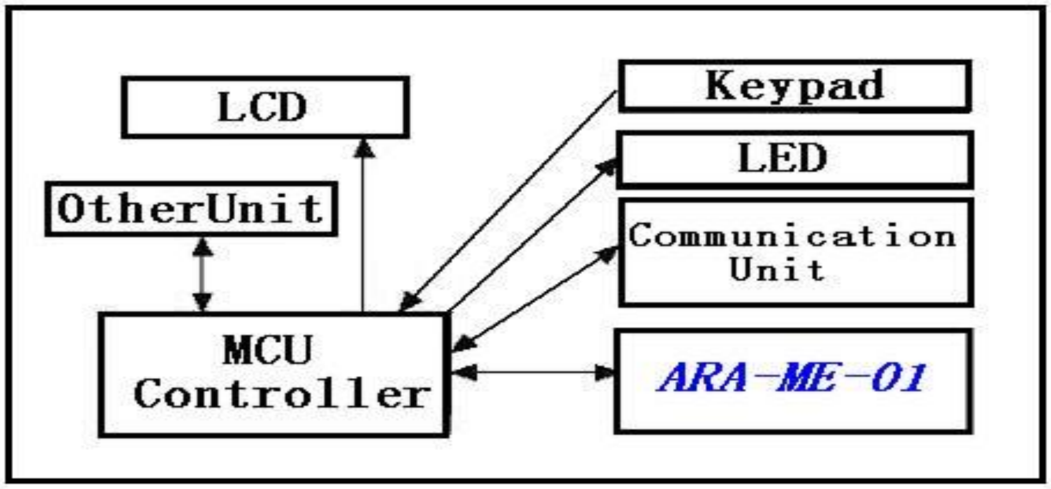


Fig : Application solution

2.Expand the port such as RS485, Wiegand, even GPRS wireless communication with central controller outside by MCU in the central controller system.

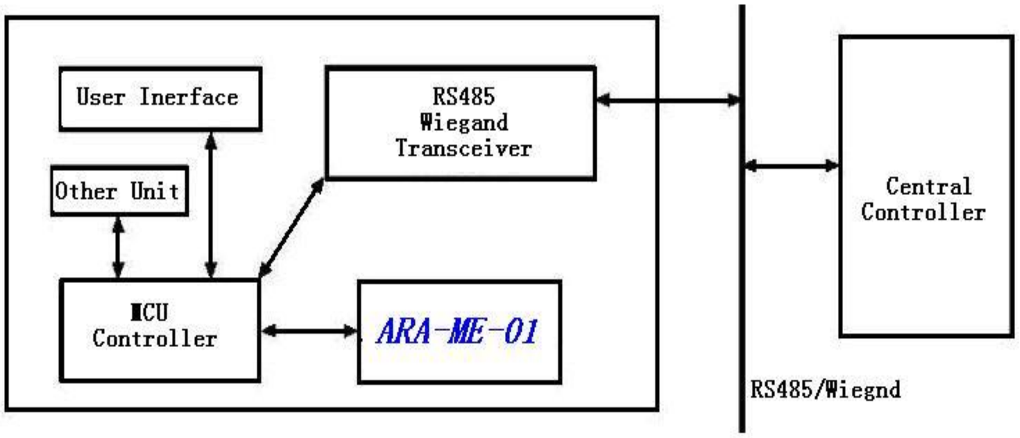


Fig : Central controller communication with GPRS

3.4.2 Pin Description

Pin	Name	Spec
1、	GND	Grand
2、	Reset	Module reset control

3	TXD	Serial sender
4	RXD	Serial receiver
5、	PowerEN	Power control
6、	VCC	5V

Table3.14: Pin description of GPRS

Software Part Specifications

1. Communication port UART (Universal Asynchronous Serial Port), 9600bps to 115200bps (option), start with 1 bit, stop with 1 bit, no check bit.
2. Communication protocol Module stays in slave mode, and host can direct the module work by different command. All the command of the host and response of the module and data transfer are in standard data pack. Host must pack and analyze the command and the data in standard format.

Fingerprint Performance

item	Spec
Sensor	AES2510
Image	256*288
Resolution	500DPI
Register time	<3
Math time （1：1）	<0.1
Math time （1：N）	<0.5
FRR	<0.1%
FAR	<0.001%
Fingerprints capacity	160 fingerprints

Table 3.15: Finger Print authentication performance.

Other Specifications

item	Symbol	min	classic	max	Measure
Work Voltage	Vn	4.5	5	5.5	V
Work Current	In	50	60	80	mA
Sleep current	Is	9	10	12	uA
Work Temperature	Tn	－20		70	℃
Humility	Hn	30		120	℉
Module size	Ln	40mm*25mm20mm*25mm			mm

Table 3.16: Other specifications of finger print module.

3.4.3 Interface

Type	**Description**
UART	3.3V CMOS level Baud rates up to 921.6kbps (factory default is 115.2kbps) RS232/422/485 supported via additional level converter
Digital I/O	3.3V CMOS level 8 ports separately configurable 26bit Wiegand I/O supported via additional level converter

Table 3.17: Interface of finger print module.

Connector Specifications

Connector	**Usage**
J1	Host interface port I
J2	Host interface port II (Molex 53261-8090 compatible)
J3	Debug port for factory use only

J4	Sensor interface port, 20 pin FPC/FFC

Table 3.18: Connector specifications

Flash Storage Structure:

Address	Spec	Size
0x0000-0x7fff	Some codes	32k
0x8000-0x8fff	System parameters storage	4k
0x9000-0xffff	Fingerprint database	28k- the end

Table 3.19: Flash storage structure

Packet Structure

Send command

The packet flag=01	command packet
The packet flag=02	data packet, and data packet followed
The packet flag=08	the last data packet

All the data packet's head is 0xEF01,0xFFFFFFFF

- The data packet can't run itself; it must follow the command packet or the response packet.
- Download or upload the data packet in the same format.
- Packet length = the total bytes quantity of from the packet length to the checksum including checksum but not including the bytes quantity of the packet length itself.

Command Response

Response packet/data packet:

The Packet flag = 07	response packet
The packet flag = 02	data packet, and packet followed
The packet flag = 08	the last data packet

3.4.4 Physical Dimensions

Main board:

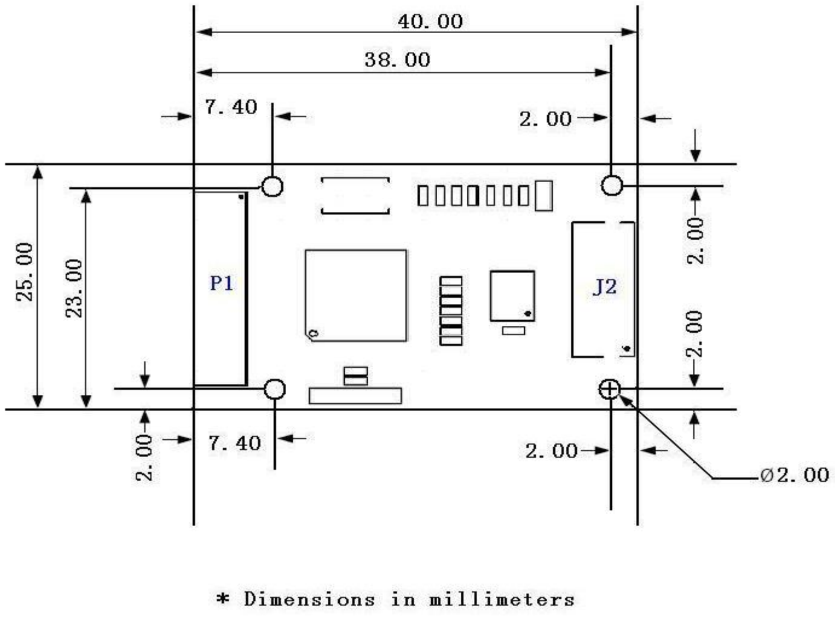


Fig 3.5: Main board physical dimensions

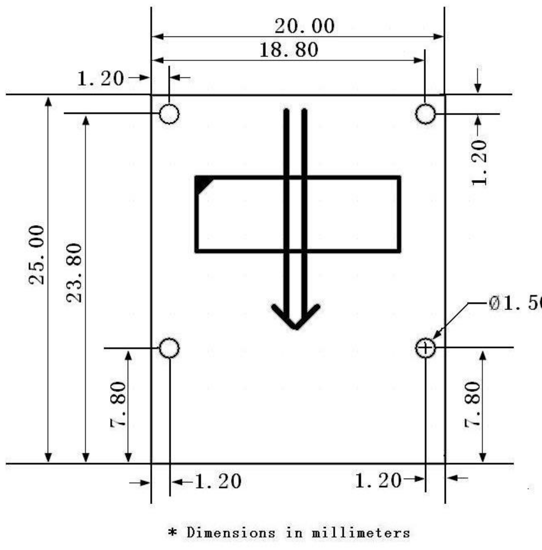


Fig : Sensor physical dimensions

3.5 GSM

GSM stands for Global System for Mobile Communications. It is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones. A modem is a device which modulates and demodulates signals as required to meet the communication requirements. It modulates an analog carrier signal to encode digital information, and also demodulates such a carrier signal to decode the transmitted information.

A GSM Modem is a device that modulates and demodulates the GSM signals and in this particular case 2G signals. The modem we are using is SIMCOM SIM900. It is a Tri-band GSM/GPRS Modem as it can detect and operate at three frequencies (EGSM 900 MHz, DCS 1800 MHz and PCS1900 MHz). Default operating frequencies are EGSM 900MHz and DCS 1800MHz.

Sim900 is a widely used in many projects and hence many variants of development boards for this have been developed. These development boards are equipped with various features to make it easy to communicate with the SIM900 module. Some boards provide only TTL interface while some boards include an RS232 interface and some others include an USB interface. If your PC has a serial port(DB9) you can buy a GSM Modem that has both TTL and RS232 interfacings in economy.

Sim900 GSM module used here consists of a TTL interface and an RS232 interface. The TTL interface allows us to directly interface with a microcontroller while the RS232 interface includes a MAX232 IC to enable communication with the PC. It also consists of a buzzer, antenna and SIM slot. Sim900 in this application is used as a DCE (Data Circuit-terminating Equipment) and PC as a DTE (Data Terminal Equipment).

3.5.1Why Use a GSM?

GSM Technology has grown so much, that literally there isn’t a place on earth where there is no GSM signal. In such a scenario GSM provides us a wide scope in controlling things remotely from any place just with our finger tips. GSM also provides ease to easily communicate in a more robust way.

A GSM module has an RS232 interface for serial communication with an external peripheral. In this case, the transmit pin (Tx) of the computer's Serial port is connected with the receive pin (Rx) of the GSM module's RS-232 interface. The transmit pin (Tx) of the RS-232 of GSM module is connected to receive pin (Rx) of microcontroller's serial transmission pin. And the serial transmit pin of the microcontroller is connected to the receive pin of the computer's Serial port. Therefore the commands and their results are transmitted and received in a triangular fashion as depicted below.

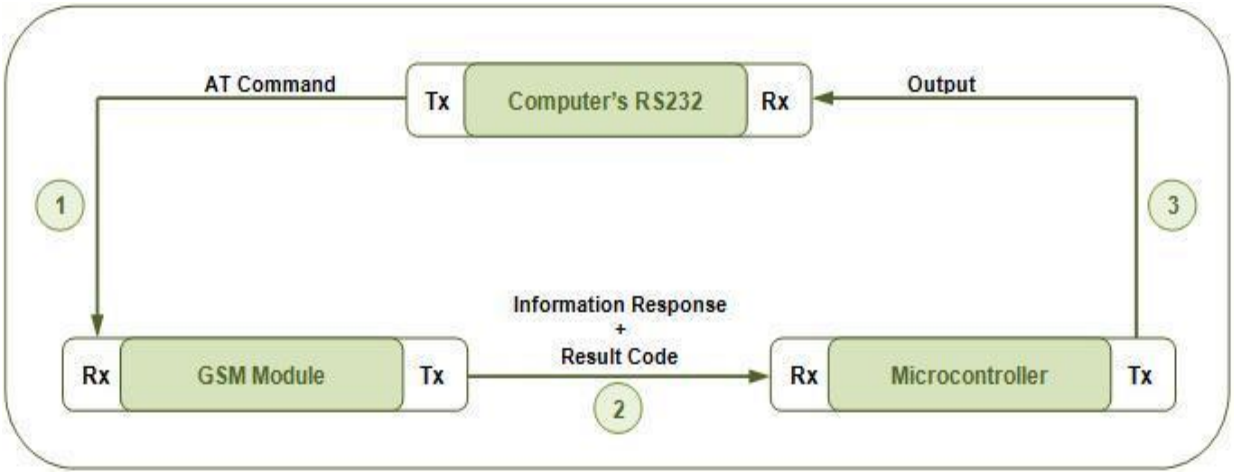


Fig 3.6: GSM module interfacing

In subsequent projects (see MC075 & MC076), the HyperTerminal will be replaced by the microcontroller itself; thus avoiding the need of using a Computer to establish an interface. This would lead to an independent GSM based system.

The microcontroller is programmed to receive and transmit data at a baud rate of 9600. For more details on setting the baud rate of microcontroller, refer serial communication with 8051.

The controller can receive data signals either by polling or by making use of serial interrupt (ES). Serial interrupt has been explained in interrupt programming. In polling, the controller continuously scans serial port for incoming data from the GSM module.

3.6. GPS

GPS is used in vehicles for both tracking and navigation. Tracking systems enable a base station to keep track of the vehicles without the intervention of the driver where, as navigation system helps the driver to reach the destination. Whether navigation system or tracking system, the architecture is more or less similar. When an accident occurred in any place then GPS system tracks the position of the vehicle and sends the information to the particular person through GSM by alerting the person through SMS or by a call.

3.6.1. Introduction to GPS Functions/Features

GPS use satellite data to calculate an accurate position on the earth. These calculations can relate the user’s position to almost any map projection within milli-seconds. All GPS work in a similar manner but they often look very different and have different software. The most significant difference between GPS receivers is the number of satellites they can simultaneously communicate with. Most receivers are described as 12 channel meaning they can communicate with 12 satellites. Older models may be 8 or even 5 channel with more modern receivers capable of communicating with 14 – 20. Given the current (2005) makeup of the GPS satellite’s constellation 12 channel is more than adequate.

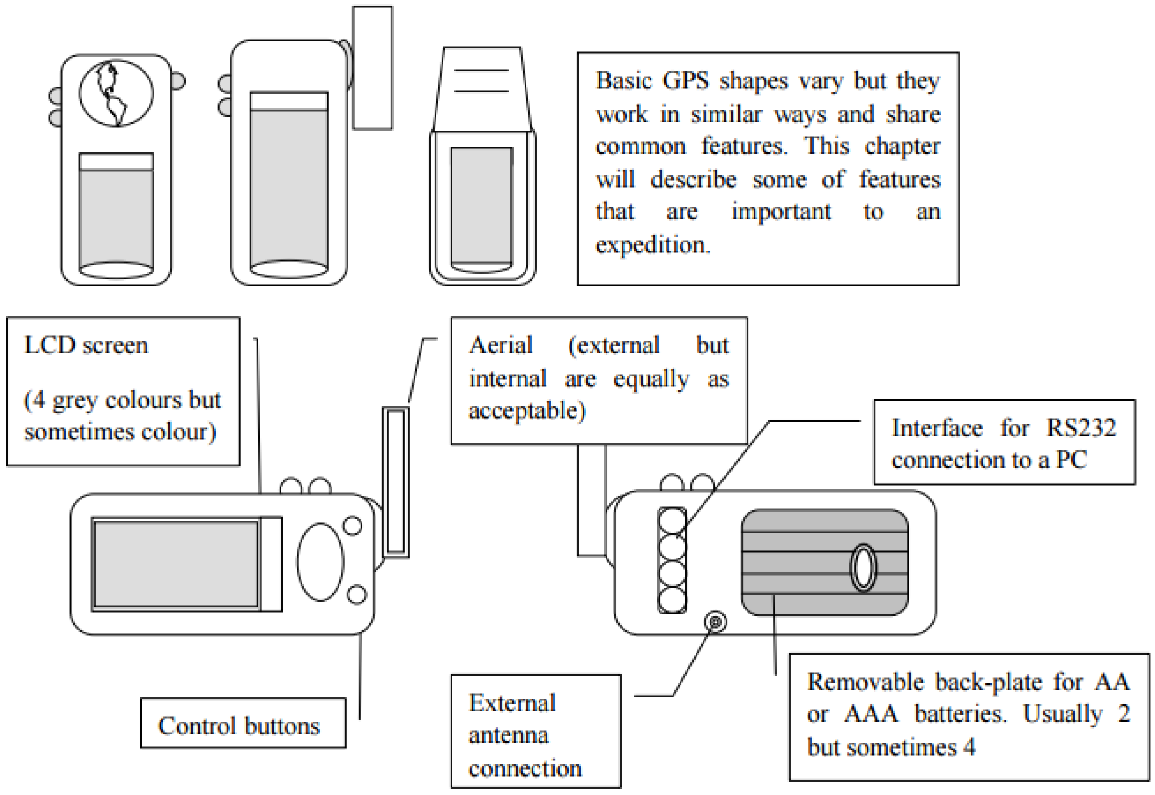

Basic GPS shapes vary but they work in similar ways and share common features. This chapter will describe some of features that are important to an expedition.
LCD screen
(4 grey colours but sometimes colour)
Aerial (external but internal are equally as acceptable)
Interface for RS232 connection to a PC
Control buttons
External antenna connection
Removable back-plate for AA or AAA batteries. Usually 2 but sometimes 4

Fig 3.7: GPS system overview

Almost all units have an LCD screen or at least software that links to a PC/PDA with an output screen. The unit might have several different pages that can be displayed on screen but usually the default page is very similar. Commonly on starting a receiver you will be presented with a map of the satellites in view. The GPS receiver shows a view of the sky split into four quadrants. These represent the NE, SE, SW, NW parts of the sky, with the concentric circles representing the horizon at 90° from the zenith, with the inner circles representing 60° and 30°. The cross at the centre represents the zenith. The dots/circles represent the satellites and the bars at the bottom represent satellite signal strength. The higher the bar the stronger the signal. This display is typical of a 12 channel set. The dots and bars will commonly be labelled with a number to represent the identity of the satellite. The bars are commonly either hollow or solid (usually white or black on a monochrome display). Hollow lines represent a satellite for which the Ephemeris data is not known. It is therefore not being used to calculate a position. Black bars represent "Fixed" satellites whose ephemeris data has been collected successfully.

These satellites are thus available for calculating a position. This is not consistent across all models and some may use grey bars as well as hollow bars to represent satellites not yet fixed.

The number, position and strength of signal from the satellites allows the GPS to calculate a rough estimate of the error in its reported position. This error or dilution of precision is a good guide to how accurate any reading would be. It should be closely monitored and readings should only be taken when this is below 10 m (ideally below 5 m).

CHAPTER 4
SOFTWARE REQUIREMENTS

4.1 ARDUINOIDE:

Now that you are a proud owner of an Arduino, or an Arduino clone, it might help if you knew what it was and what you can do with it.

In its simplest form, an Arduino is a tiny computer that you can program to process inputs and outputs going to and from the chip.

The Arduino is what is known as a Physical or Embedded Computing platform, which means that it is an interactive system, that through the use of hardware and software can interact with it’s environment.

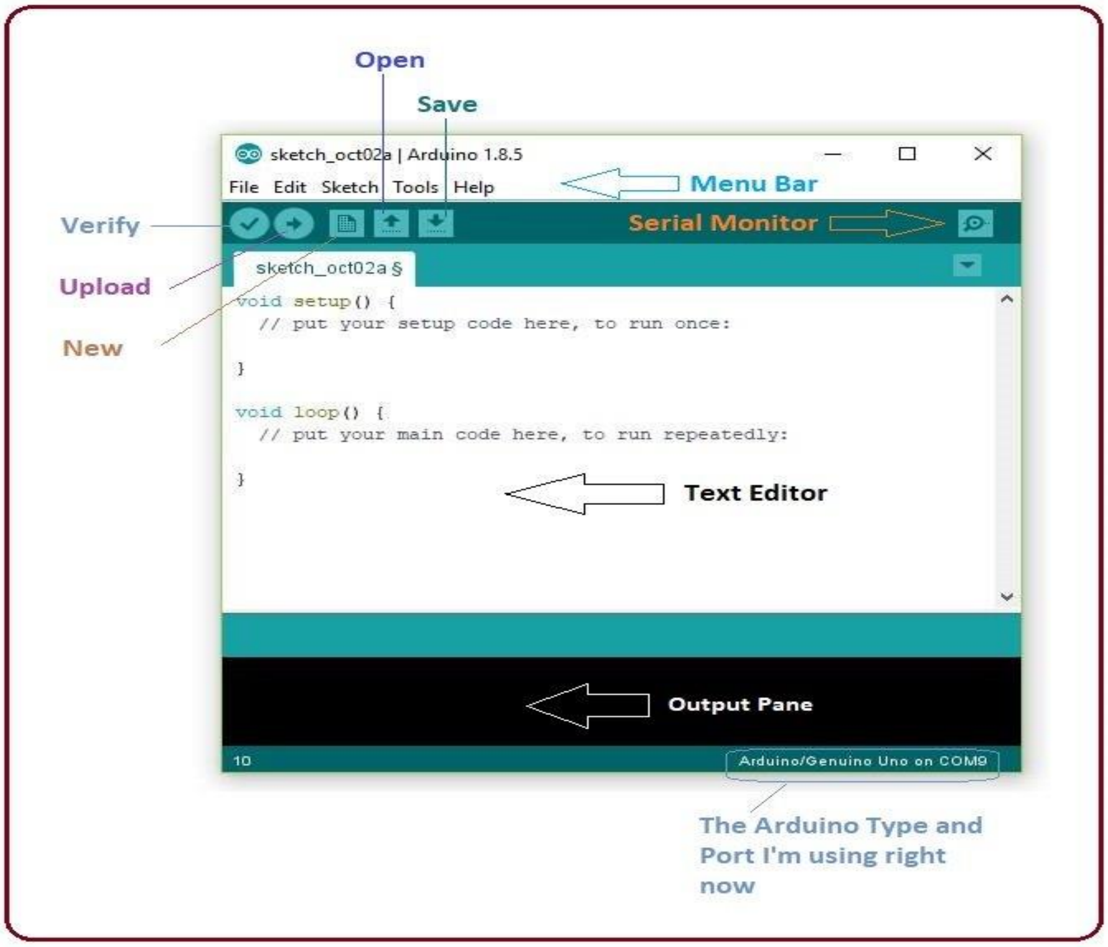
For example, a simple use of the Arduino would be to turn a light on for a set period of time, let’s say 30 seconds, after a button has been pressed (we will build this very same project later in the book). In this example, the Arduino would have a lamp connected to it as well as a button. The Arduino would sit patiently waiting for the button to be pressed. When you press the button it would then turn the lamp on and start counting. Once it had counted 30 seconds it would then turn the lamp off and then carry on sitting there waiting for another button press. You could use this set-up to control a lamp in an under-stairs cupboard for example. You could extend this example to sense when the cupboard door was opened and automatically turn the light on, turning it off after a set period of time.

The Arduino can be used to develop stand-alone interactive objects or it can be connected to a computer to retrieve or send data to the Arduino and then act on that data (e.g. Send sensor data out to the internet). The Arduino can be connected to LEDs. Dot Matrix displays, LED displays, buttons, switches, motors, temperature sensors, pressure sensors, distance sensors, webcams, printers, GPS receivers, Ethernet modules,

The Arduino board is made of an Atmel AVR Microprocessor, a crystal or oscillator (basically a crude clock that sends time pulses to themicrocontroller to instructions and interact with the world outside. In the Arduino world, programs are known as sketches.

The Arduino hardware and software are both Open Source, which means the code, the schematics, design, etc. are all open for anyone to take freely and do what they like with it. This means there is nothing stopping anyone from taking the schematics and PCB designs of the Arduino and making their own and selling them. This is perfectly legal, and indeed the whole purpose of Open Source, and indeed the Fredonia that comes with the Earthshine Design Arduino Starter Kit is a perfect example of where someone has taken the Arduino PCB design, made their own and are selling it under the Fredonia name. You could even make your ownArduino, with just a few cheap components, on a breadboard.

The only stipulation that the Arduino development team put on outside developers is that the Arduino name can only be used exclusively by them on their own products and hence the clone boards have names such as Fredonia, Boarding, Borodino, etc.


Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Upload
New
Serial Monitor
sketch_oct02a §
Text Editor
Output Pane
10
Arduino/Genuino Uno on COM9
The Arduino Type and Port I'm using right now

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 4.1: Arduino IDE

As the designs are open source, any clone board, such as the Fredonia, is 100% compatible with the Arduino and therefore any software, hardware, shields, etc. will all be 100% compatible with a genuineArduino. The Arduino can also be extended with the use of shields which are circuit boards containing other devices (e.g. GPS receivers, LCD Displays, Ethernet connections, etc.) that you can simply slot into the top of your Arduino to get extra functionality. You don't have to use a shield if you don't want to as you can make the exact same circuitry using a breadboard, some board or even by making your own PCBs.

Probably the most versatile Arduino, and hence the reason it is the most popular, is the Duemilanove. This is because it uses a standard 28 pin chip, attached to an IC Socket. The beauty of this systems is that if you make something neat with the Arduino and then want to turn it into something permanent (e.g. Or under- stairs cupboard light), then instead of using the relatively expensive Arduino board, you can simply use the Arduino to develop your device, then pop the chip out of the board and place it into your own circuit board in your custom device. You would then have made a custom embedded device, which is really cool.

Then, for a couple of quid or bucks you can replace the AVR chip in your Arduino with a new one. The chip must be pre-programmed with the Arduino Bootloader to enable it to work with the Arduino IDE, but you can either burn the Bootloader yourself if you purchase an AVR Programmer, or you can buy these pre- programmed from many suppliers around the world. Of course, Earthshine Design provide pre-programmed Arduino chips in its store for a very reasonable price.

If you do a search on the Internet by simply typing 'Arduino' into the search box of your favorite search engine, you will be amazed at the huge-amountof websites dedicated to the Arduino. You can find a mind-boggling amount of information on projects made with the Arduino and if you have a project in mind, will easily find information that will help you to get your project up and running easily.

The Arduino is an amazing device and will enable you to make anything from interactive works of art to robots. With a little enthusiasm to learn how to program the Arduino and make it interact with other components a well as a bit of imagination, you can build anything you want. This book and the kit will give you the necessary skills needed to get started in this exciting and creative hobby. So, now you know what an Arduino is and what you can do with it, let's open up the starter kit and dive right in.

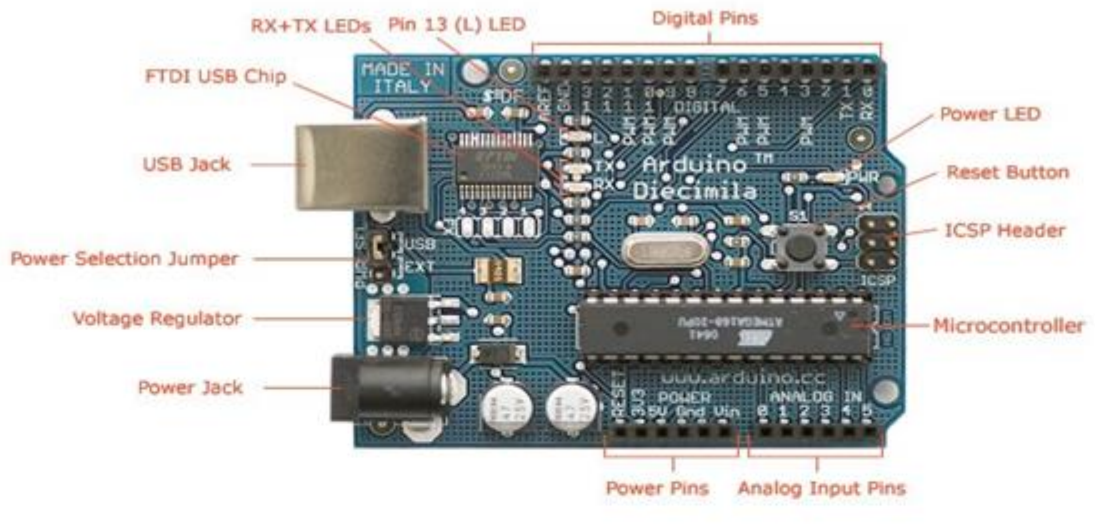


Fig 4.2: Arduino board

Firstly, get your Freeduino board and lay it on thetable in front of you. Take the USB cable and plug the B plug (the fatter squarer end) into the USB socket on the Freeduino.

Fig: Arduino USB cable

At this stage do NOT connect the Freeduino to your PC or Mac yet. Download the Arduino IDEfrom the Arduino download page. As of the time of writing this book, the latest IDE version is 0015. The file is a ZIP file so you will need to un-compress it. Once thedownloadhas finished, unzip the file, making sure that you preserve the folder structure as it is and do not make any changes.

- If you double-click the folder, you will see a few files and sub-folders inside.

- Install the USB Drivers
- If you are using Windows, you will find the drivers in the drivers/FTDIUSBDriversdirectory of the Arduino distribution.

If you have a Mac these are in the driversdirectory. If you have an older Mac like a PowerBook, iBook, G4 or G5, you should usethe PPC drivers:

FTDIUSBSerialDriver_v2_1_9.dmg. If you have a newer Mac with an Intel chip, you need the Intel drivers: FTDIUSBSerialDriver_v2_2_9_Intel.dmg. Double-click to mount the disk image and run the includedFTDIUSBSerialDriver.pkg.

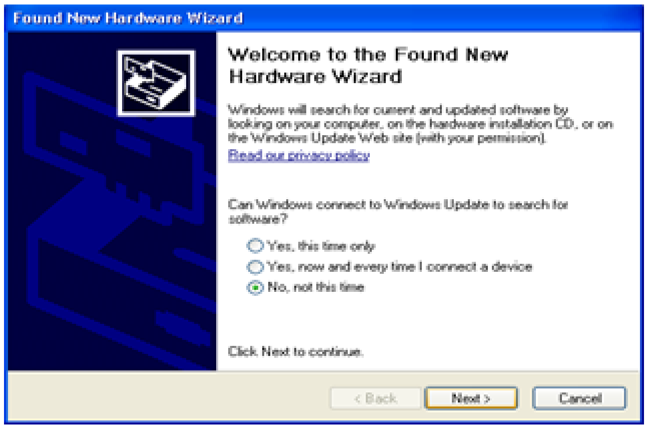
The latest version of the drivers can be found on the FTDI website.Connect the Freeduino. First, make sure that the little power jumper, between the power and USB sockets, is set to USB and not external power (not applicable if you have a Roboduino board, which has an Auto Power Select function).

Using the jumper, you can either power the board from the USB port (good for low current devices like LEDs, etc.) or from an external power supply (6-12V DC).

Now, connect the other end of the USB cable into the USB socket on your PC or Mac. You will now see the small power LED (marked PWR above the RESET switch) light up to show you have power to the board.

If you have a Mac, this stage of the process is complete and you can move on to the next Chapter. If you are using Windows, there are a few more steps to complete.

On Windows, the Found New Hardware Wizard will now open up as Windows will have detected that you have connected a new piece of hardware (your Freeduino board) to your PC. Tell it NOT to connect to Windows update (Select No, not at this time) and then click Next.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

Yes, this time only

Yes, now and every time I connect a device

[✓ No, not this time]

Click Next to continue.

Fig 4.3: Arduino installation wizard

Then follow on screen instructions to finish the hardware installation of the Arduino. You are ready to go, now. You will see the Sketch inside the white code window.

Now, before we upload the Sketch, we need to tell the IDE what kind of Arduino we are using and the details of our USB port. Go to the file menu and click Tools, then clock on Board. You will be presented with a list of all of the different kinds of Arduino board that can be connected to the IDE. Our Freeduino board will either be fitted with an Atmega328 or an Atmega168 chip so choose "Arduino Duemilanove w/ATmega328" if you have a 328 chip or "Arduino Diecimila or Duemilanove w/ ATmega328P" if you have a 328 chip.

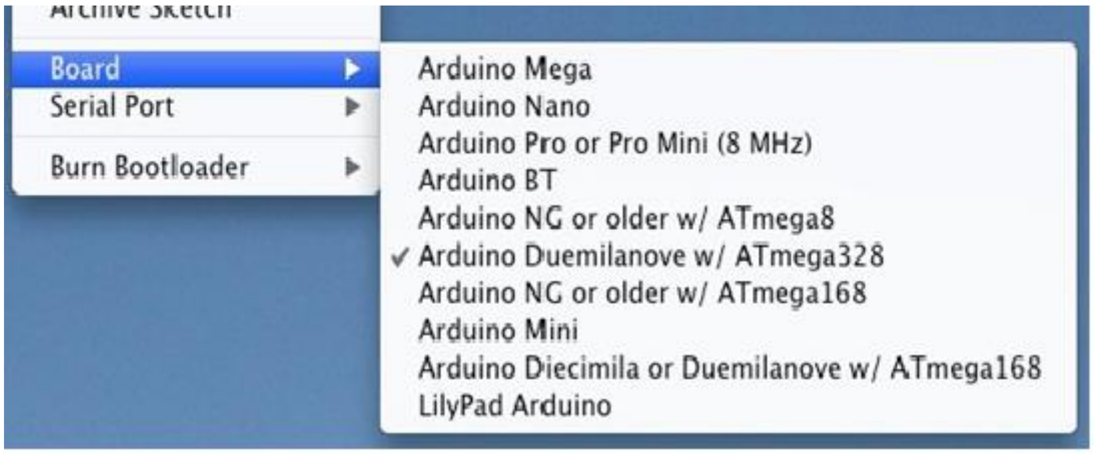

Archive Sketch
Board
Serial Port
Burn Bootloader
Arduino Mega
Arduino Nano
Arduino Pro or Pro Mini (8 MHz)
Arduino BT
Arduino NG or older w/ ATmega8
✓ Arduino Duemilanove w/ ATmega328
Arduino NG or older w/ ATmega168
Arduino Mini
Arduino Diecimila or Duemilanove w/ ATmega168
LilyPad Arduino

Fig : Configure the IDE to required Arduino board

Now that your Freeduino has been connected and the drivers for the USB chip have been installed, we are now ready to try out the Arduino for the first time and upload your first Sketch.

Navigate to your newly unzipped Arduino folder and look for the Arduino IDE icon, which looks something like this.

Double click the ICON to open up the IDE. You will then be presented with a blue and white screen with a default sketch loaded inside.

This is the Arduino IDE (Integrated Development Environment) and is where you will write your Sketches (programs) to upload to your Arduino board.

We will take a look at the IDE in a little more detail in the next chapter. For now, simply click File in the file menu and scroll down to Sketchbook. Then scroll down to Examples and click it. You will be presented with a list of Example sketches that you can use to try out your Arduino. Now click on Digital and inside there you will find an example Sketch called Blink. Click on this. The Blink Sketch will now be loaded into the IDE.

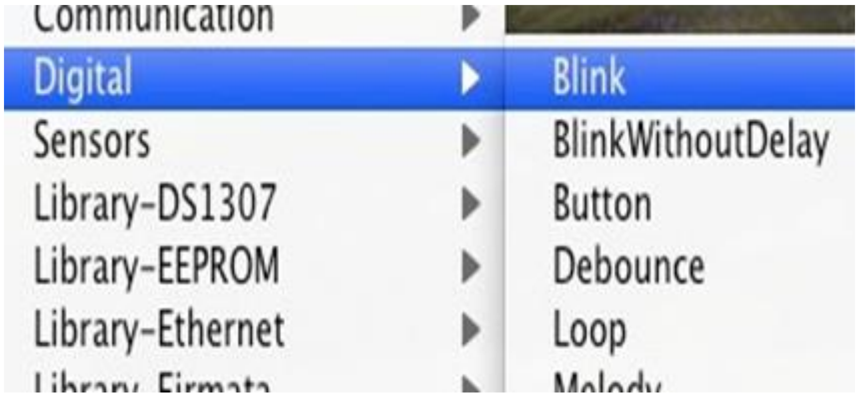


Fig : Select the required program to dump in board

Now you need to tell the IDE the details of your USB port, so now click on Tools again, scroll down to Serial Port and a list of the available serial ports on your system will be displayed. You need to choose the one that refers to your USB cable, which is usually listed as something like /dev/tty.usbserial-xxxx ona Mac or something like Com 4 on Windows so click on that. If not sure, try each one till you find one that works.

Now that you have selected the correct board and USB port you are ready to upload the Blink Sketch to the board.

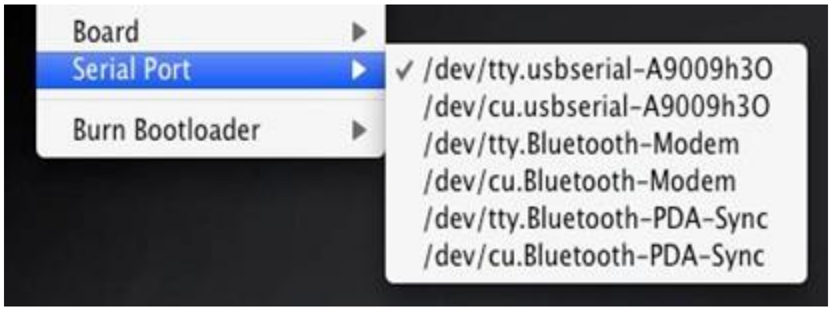

Board
Serial Port
Burn Bootloader
✓ /dev/tty.usbserial-A9009h3O
/dev/cu.usbserial-A9009h3O
/dev/tty.Bluetooth-Modem
/dev/cu.Bluetooth-Modem
/dev/tty.Bluetooth-PDA-Sync
/dev/cu.Bluetooth-PDA-Sync

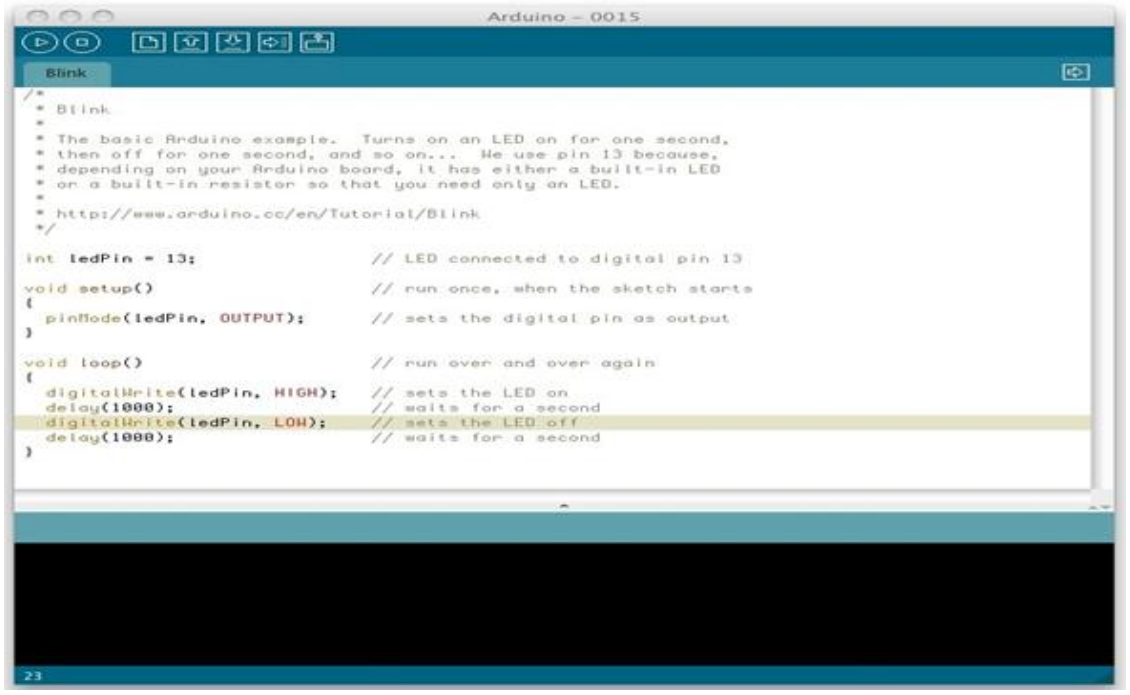
Fig : Configuring the serial port

You can either click the Upload button, which is the 6th button from the left at the top with an arrow pointing to the right (hover your mouse pointer over the buttons to see what they are) or by clicking on File in the file menu and scrolling down to Upload to I/O Board and clicking on that.

Once the data has been uploaded to the board successfully you will get a Done Uploading messagein the IDE and the RX/TX LEDs will stop flashing.The Arduino will now reset itself and immediately start to run the Sketch that you have just uploaded.Just below the code window too.

The Blink-sketch is a very simple sketch that blinks LED 13, which is a tiny green LED soldered to the board and also connected to Digital Pin 13 from the Microcontroller, and will make it flash-on and off every 1000 milliseconds, or 1 second. If your sketch has uploaded successfully, you will now see this LED happily flashing on and off slowly on your board.

If so, congratulations, you have just successfully installed your Arduino, uploaded and ran your first sketch. We will now explain a bit more about the Arduino IDE and how to use it before moving onto the projects that you can carry out using the hardware supplied with the kit. For our first project we will carry out this Blink LED sketch again, but this time using an LED that we will physically connect to one of the digital output pins on the Arduino. We will also explain the hardware and software involved in this simple project. But first, let's take a closer look at the Arduino IDE.



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Fig 4.4: An example program-blinking LED

When you open up the Arduino IDE it will look very similar to the image above. If you are using Windows or Linux there will be some slight differences but the IDE is pretty much the same no matter what OS you are using.

The IDE is split up into the Toolbar across the top, the code or Sketch Window in the centre and the Serial Output window at the bottom.

The Toolbar consists of 7 buttons, underneath the Toolbar is a tab, or set of tabs, with the filename of the code within the tab. There is also one further button on the far right hand side.

Along the top is the file menu with drop down menus headed under File, Edit, Sketch, Tools and Help. The buttons in the Toolbar provide convenient access to the most commonly used functions within this file menu.

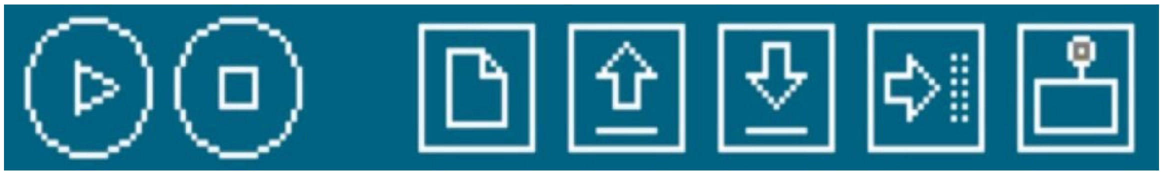


Fig 4.5: Toolbar of Arduino IDE

The Toolbar buttons are listed above. The functions of each button are as follows :-

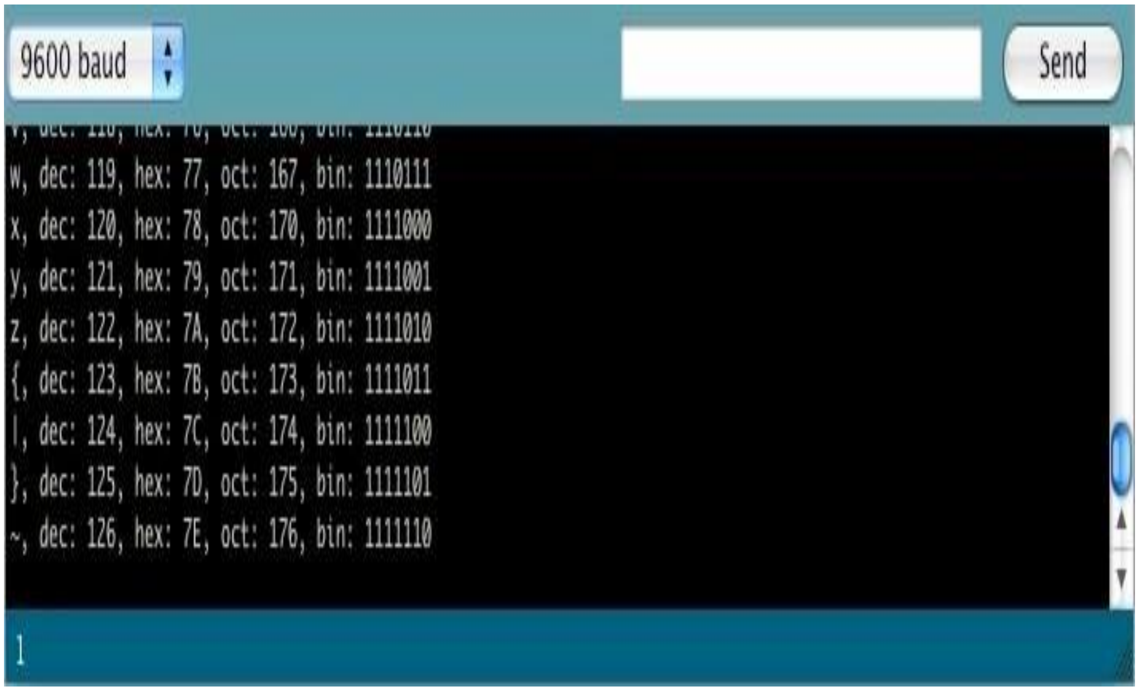
Verify/Compile	Checks the code for errors
Stop	Stops the serial monitor, or un-highlights other buttons
New	Creates a new blank Sketch
Open	Shows a list of Sketches in your sketchbook
Save	Saves the current Sketch
Upload	Uploads the current Sketch to the Arduino
Serial Monitor	Displays serial data being sent from the Arduino

Table 4.1: Arduino IDE Toolbar description

- The Verify/Compile button is used to check that your code is correct, before you upload it to your Arduino.
- The Stop button will stop the Serial Monitor from operating. It will also un-highlight other selected buttons. Whilst the Serial Monitor is operating you may wish to press the Stop button to obtain a 'snapshot' of the serial data so far to examine it. This is particularly useful if you are sending data out to the Serial Monitor quicker than you can read it.
- The New button will create a completely new and blank Sketch read for you to enter code into. The IDE will ask you to enter a name and a location for your Sketch (try to use the default location if possible) and will then give you a blank

Sketch ready to be coded. The tab at the top of the Sketch will now contain the name you have given to your new sketch.

- The Open button will present you with a list of Sketches stored within your sketchbook as well as a list of Example sketches you can try out with various peripherals once connected.
- The Save button will save the code within the sketch window to your sketch file. Once complete you will get a Done Saving message at the bottom of the code window
- The Upload to I/O Board button will upload the code within the current sketch window to your Arduino. You need to make sure that you have the correct board and port selected (in the Tools menu) before uploading. It is essential that you Save your sketch before you upload it to your board in case a strange error causes your system to hang or the IDE to crash. It is also advisable to Verify/Compile the code before you upload to ensure there are no errors that need to be debugged first.
- The Serial Monitor is a very useful tool, especially for debugging your code. The monitor displays serial data being sent out from your Arduino (USB or Serial board). You can also send serial data back to the Arduino using the Serial Monitor. If you click the Serial Monitor button you will be presented with an image like the one above.
- On the left-hand side, you can select the Baud Rate that the serial data is to be sent to/from the Arduino. The Baud Rate is the rate, per second, that characters (data) is sent to/from the board. The default setting is 9600 baud, which means that if you were to send a text novel over the serial communications line (in this case your USB cable) then 9600 letters, or symbols, of the novel, would be sent per second.



```
w, dec: 119, hex: 77, oct: 167, bin: 1110111
x, dec: 120, hex: 78, oct: 170, bin: 1111000
y, dec: 121, hex: 79, oct: 171, bin: 1111001
z, dec: 122, hex: 7A, oct: 172, bin: 1111010
{, dec: 123, hex: 7B, oct: 173, bin: 1111011
|, dec: 124, hex: 7C, oct: 174, bin: 1111100
}, dec: 125, hex: 7D, oct: 175, bin: 1111101
~, dec: 126, hex: 7E, oct: 176, bin: 1111110
```

Fig : Serial Monitor

To the right of this is a blank text box for you to enter text to send back to the Arduino and a Send button to send the text within that field. Note that no serial data can be received by the Serial Monitor unless you have set up the code inside your sketch to do so. Similarly, the Arduino will not receive any data sent unless you have coded it to do so.

Finally, the black area is where your serial data will be displayed. In the image above, the Arduinois running the ASCIITable sketch, that can be found in the Communications examples. This program outputs ASCII characters, from the Arduino via serial (the USB cable) to the PC where the Serial monitor then displays them.

To start the Serial Monitor,press the Serial Monitor button and to stop it press the Stop button. On a Mac or in Linux, Arduino board will reset itself (rerun the code from the beginning) when you click the Serial Monitor button.

Once you are proficient at communicating via serial to and from the Arduino you can use otherprograms suchas Processing, Flash, MaxMSP, etc. To communicate between the Arduino and your PC.

We will make use of the Serial Monitor later on in our projects when we read data from sensors and get the Arduino to send that data to the Serial Monitor, in human readable form, for us to see.

The Serial Monitor window is also were you will see error messages (in red text) that the IDE will display to you when trying to connect to your board, upload code or verify code.

Below the Serial Monitor at the bottom left you will see a number. This is the current line that the cursor, within the code window, is at. If you have code in your window and you move down the lines of code (using the ↓ key on your keyboard) you will see the number increase as you move down the lines of code. This is useful for finding bugs highlighted by error messages. Across the top of the IDE window (or across the top of your screen if you are using a Mac) you will see the various menus that you can click on to access more menu items.

Fig 4.6: Menu Bar

The menu bar across the top of the IDE looks like the image above (and slightly different in Windowsand Linux). I will explain the menus as they are on a Mac, the details will also apply to the Windows and Linux versions of the IDE.

The first menu is the Arduino menu.Within this is the About Arduino option, which when pressed will show you the current version number, a list of the people involved in making this amazing device and some further information.

Underneaththat isthe Preferences option. This will bringup thePreferences window where you can change various IDE options, such as were you default Sketchbook is stored, etc.Also, is the Quit option, which will Quit the program.

The next menu is the File menu. In here you get access to options to create a New sketch, take a look at Sketches stored in your Sketchbook (as well as the Example Sketches), options to Save your Sketch (or Save As if you want to give it a different name). You also have the option to upload your sketch to the I/O Board (Arduino) as well as the Print options for printing out your code.

Next is the Edit menu. In here you get options to enable you to Cut, Copy and Paste sections of code. Select All of your code as well as Find certain words or phrases within the code. Also included are the useful Undo and Redo options which come in handy when you make a mistake.

Our next menu is the Sketch menu which gives us access to the Verify/Compile functions and some other useful functions you will use later on. These include the Import Library option, which when clicked will bring up a list of the available libraries, stored within your

The next menu in the IDE is the Tools menu. Within this are the optionsto select the Board and Serial Port we are using, as we did when setting upthe Arduino for the first time. Also, we have the Auto Format functionthat formats your code to make it look nicer.

The Copy for Forum option will copy the code within the Sketch window, but in a format that when pasted into the Arduino forum (or most other Forums for that matter) will show up the same as it is in the IDE, along with syntax coloring etc.

The Archive Sketch option will enable you tocompress your sketch into a ZIP file and asks you were you want to store it.

CHAPTER 5
FLOW CHART AND WORKING PRINCIPLE

5.1 FLOW CHART:

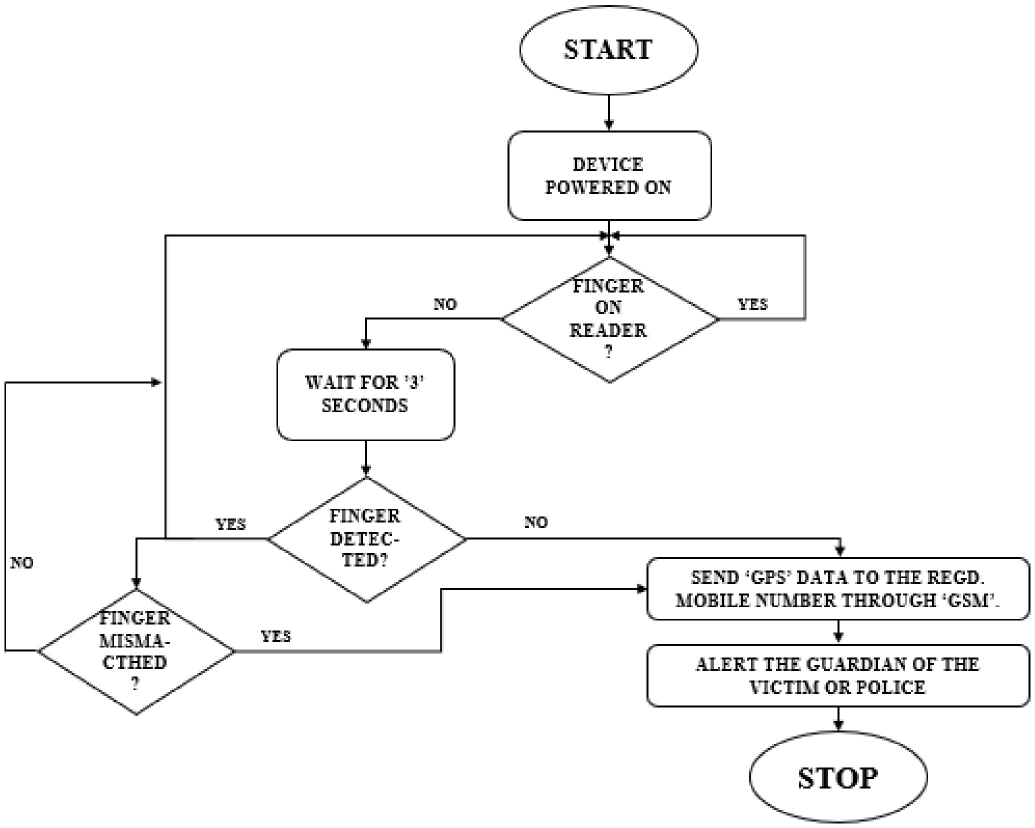


Fig 5.1: Flow Chart

5.2 WORKING PRINCIPLE:

This project clearly uses two main modules of GSM and a microcontroller. The user when sends the messages through his phones those reaches the GSM through the AT commands all those messages reaches the microcontroller. That microcontroller takes the data in terms of bits through the Max232.This informationwill be transmitted to the LCD display.

5.3 ALGORITHM:

1. Initialize GPS sensor with 9600 baud-rate.
2. Connect GPS TX Pin connected to Aurdino RX pin 0.
3. Once power is on it takes 3 min to 5 min to activate GPS sensor.
4. GPS sensor is giving different data like GPGGA, GPGSV,GPGSA.
5. In that we require GPGMC.
6. From that we have to extract the required data.
7. Finally, display the data on the LCD display.

5.4 APPLICATIONS:

The following are the applications:

- Can be utilized for the security of ladies, kids, impaired and matured individuals.
- Can be utilized as a legitimate proof of crime with exact location information for prosecution.
- Location data for indictment.

5.5 ADVANTAGES:

The following are the advantages:

- Safety Device which can be conveyed by everybody.
- Ultra-low power utilization.
- Compact in size with Wireless network.
- Easy and quick to install & Easy Maintenance.
- Low taken a toll with elite.

CHAPTER 6

RESULTS

6.1 RESULTS:

BEFORE EXECUTION:

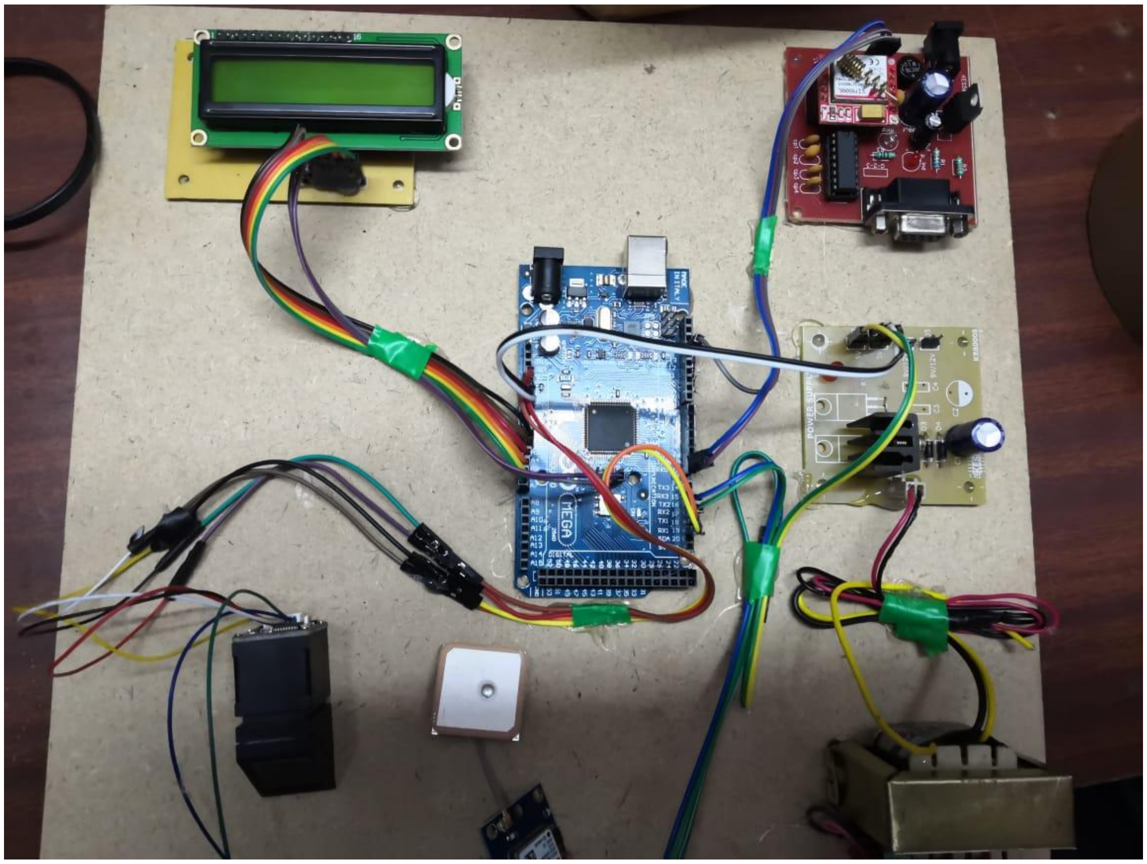
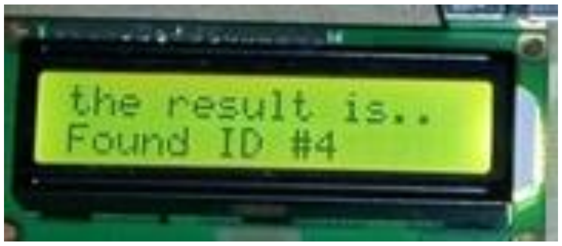


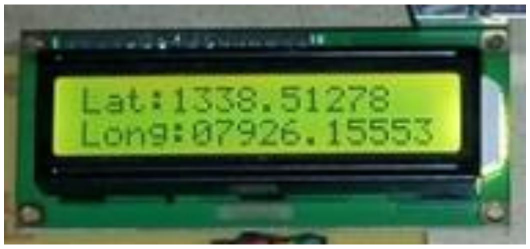
Fig6.1: Project kit

AFTER EXECUTION:



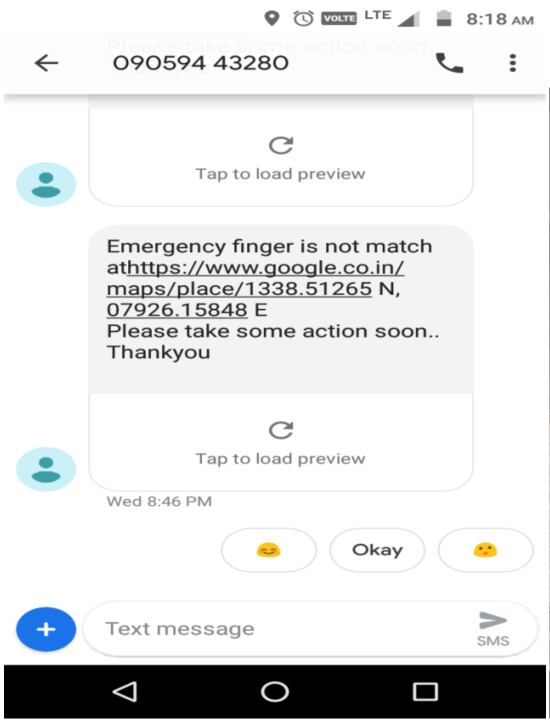
the result is..
Found ID #4

Fig6.2: Safe Message- ID found



Lat:1338.51278
Long:07926.15553

Fig6.3: Latitude and Longitude information



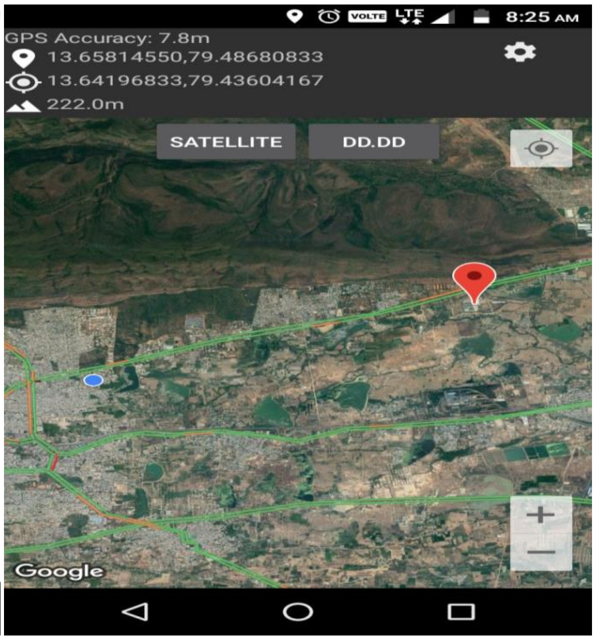
090594 43280

Tap to load preview

Emergency finger is not match athttps://www.google.co.in/maps/place/1338.51265 N, 07926.15848 E
Please take some action soon..
Thankyou

Tap to load preview

Wed 8:46 PM



GPS Accuracy: 7.8m
13.65814550,79.48680833
13.64196833,79.43604167
222.0m

Fig6.3:Message received and live location of the victim.

CHAPTER 7
CONCLUSION

In this paper an alternative approach for device switching which combines fingerprint identification technique with GSM and GPS functionalities has been proposed. The device switching from remote location removes the necessity of the person to be present near the device to operate it. This approach allows more than one person to control the device functionality and the authentication facility provided by the fingerprint sensor helps to reduce the fault correction time. This provides women security. Being safe and secure is the demand of the day.

CHAPTER 8
REFERENCES

- "Pantelopoulos A, Bourbakis NG. An unmistakable view on sensor-based systems for prosperity checking and foresight. IEEE Transactions on Systems, Man and Cybernetics – part C: Applications and Reviews. 2010 Jan; 40(1):1–12."
- "Suraksha. A instrument to help ladies stuck in an unfortunate situation: An action by an understudy of ITM University Gurgaon. efytimes. com. 2013. Available from: http://efytimes.com/e1/118387/SURAKSHA-A-Device-To-Help-Women-In-Distress-An-Initiative-By-A-Student-Of-ITM-University-Gurgaon.pdf"
- "Vigneshwari S, Aramudhan M. Social data recovery in light of semantic elucidation and hashing upon the diverse ontologies. Indian Journal of Science and Technology. 2015 Jan; 8(2):103–7."
- "Toney G. Jaban F, Puneeth S. et al. Diagram and utilization of security arm band for women and children using ARM7. 2015 International Conference on Power and Advanced Control Engineering (ICPACE); Bangalore. 2015 Aug 12-14. p. 300–3."
- "Chand D, Nayak S, Bhat KS, Parikh S. An intelligent system for Women's Safety: WoS App. 2015 IEEE Region 10 Conference TENCON; Macao. 2015 Nov 1-4. p. 1–5."