**1. INTRODUCTION**

Chapter-1

**1.1 OBJECTIVE**

Gesture recognition can be termed as an approach in this direction. It is the process by which the gestures made by the user are recognized by the receiver. Gestures are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of:

• conveying meaningful information or

• interacting with the environment.

They constitute one interesting small subspace of possible human motion. A gesture may also be perceived by the environment as a compression technique for the information to be transmitted elsewhere and subsequently reconstructed by the receiver.

This Gesture Recognition methodology has become increasingly popular in a very short span of time. The low moderate cost and relative small size of the accelerometers are the two factors that make it an effective tool to detect and recognize human body gestures. Several studies have been conducted on the recognition of gestures from acceleration data using Artificial Neural Networks (ANNs) .

**1.2 ORGANIZATION OF THEISIS**

Chapter-1

The **Chapter 1** gives a brief objective of the project undertaken and an insight to various chapters present in the project report.

The **Chapter 2** gives the block diagram description of the transmitter and receiver with their detailed description and various components involved

The **Chapter 3** gives a detailed insight to the schematic diagram of the PCB board being employed in the project with its detailed description

The **Chapter 4** gives a detailed description of all the hardware components employed in this project with its role in the project

The **Chapter 5** gives a detailed description of all the software components used in the project with the source code and the procedure to interface the hardware and the Personal Computer

The **Chapter 6 & 7** gives the conclusion of the project, the findings and the future scope of the project.

The **Chapter 8** gives a detailed account of the references and the bibliography that were used in the completion of the project.

**2. BLOCK DIAGRAM**

Chapter-2

**2.1 TRANSMITTER**

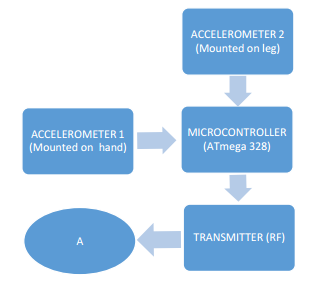
****

Figure-2.1 Block diagram of a transmitter

**2.2 RECIEVER**

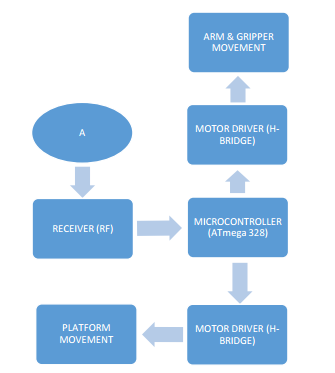


Figure-2.2 Block diagram of a receiver

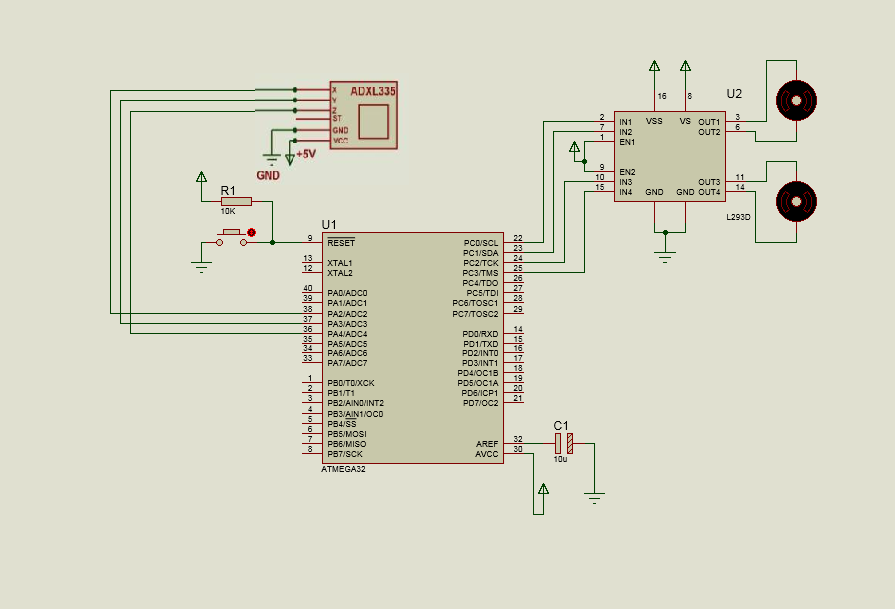
**2.3 DESCRIPTION**

Chapter-2

This part is the heart of the entire project. Without an effective and reliable communication system, no system / project can work. Similar is the case with this project also. The RF Module, details of which are mentioned under Section 3.2, is the only communication equipment required in this project. This Module is used to transmit the different hand and leg gestures made by the user (encoded in the form of 4-bit digital data) wirelessly to the receiver , which decodes the received 4-bit digital data and according to which the arm, gripper and platform moves. The block diagrams shown in Figure 2.1 & Figure 2.2 depict the entire communication system of the project. The Linker (Circle, named “A”) in Figure 2.1 and Figure 2.2 is used to show the connection (flow of signals) between the Transmitter End and the Receiver End.

**3. SCHEMATIC DIAGRAM**

Chapter-3

****

**Figure-3.1 schematic of the project**

**3.1 SCHEMATIC EXPLANATION:**

In this unit we are using a Avr series of microcontroller which is having inbuilt ADC. Accelerometer XY signals are applied to the ADC input of microcontroller for conversion of analog signal into digital form .Then these signal are compared with a threshold value using programming and transmitted in a form of 4 bit data signals to a RF encoder IC HT 12E.This IC converts this 4 bit information into serial form by which it can be transmitted by a RF transmitter.

The g-cell is a mechanical structure formed from semiconductor materials (polysilicon) using semiconductor processes (masking and etching). It can be modeled as a set of beams attached to a movable central mass that move between fixed beams. The movable beams can be deflected from their rest position by subjecting the system to an acceleration (Figure 3). As the beams attached to the central mass move, the distance from them to the fixed beams on one side will increase by the same amount that the distance to the fixed beams on the other side decreases. The change in distance is a measure of acceleration. The g-cell beams form two back-to-back capacitors.

Chapter-3

**4. HARDWARE COMPONENTS**

Chapter-4

**4.1 THE MICROCONTROLLER:**

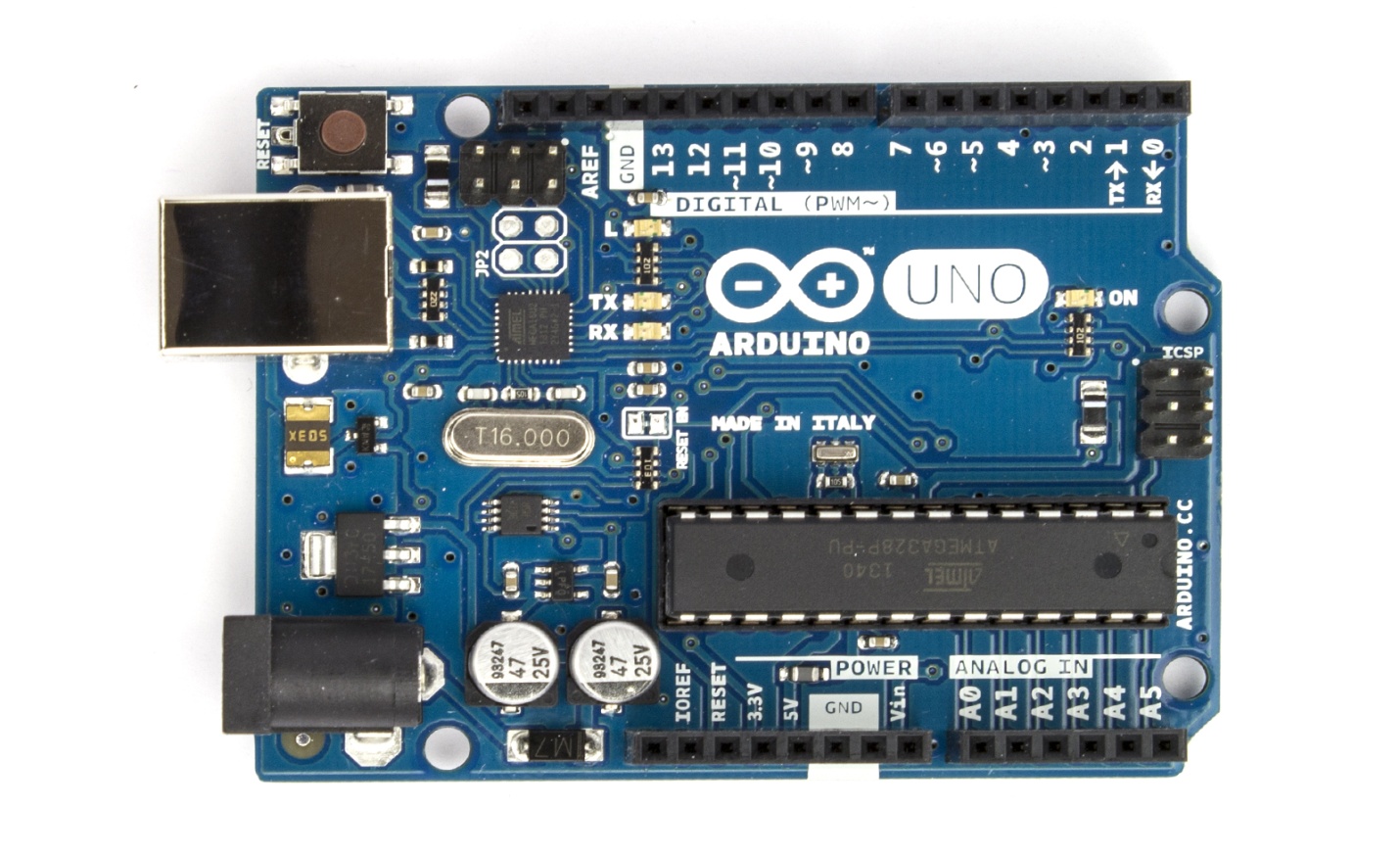
****

Fig 4.1 Arduino Uno Board

**4.1.1 Features**

|  |  |
| --- | --- |
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA  Chapter-4 |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Clock Speed | 16 MHz |

**4.1.1.1 Arduino:**

Circumstances that we find ourselves in today in the field of microcontrollers had their beginnings in the development of technology of integrated circuits. This development has made it possible to store hundreds of thousands of transistors into one chip. That was a prerequisite for production of microprocessors, and the first computers were made by adding external peripherals such as memory, input-output lines, timers and other. Further increasing of the volume of the package resulted in creation of integrated circuits. These integrated circuits contained both processor and peripherals. That is how the first chip containing a microcomputer, or what would later be known as a microcontroller came about.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Microprocessors and microcontrollers are widely used in embedded systems products. Microcontroller is a programmable device. A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports and a timer embedded all on a single chip. The fixed amount of on-chip ROM, RAM and number of I/O ports in microcontrollers makes them ideal for many applications in which cost and space are critical.

Chapter-4

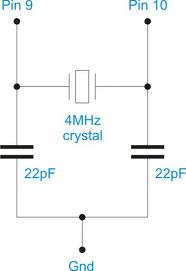
The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to One-Time Programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

**4.1.1.2 Crystal Oscillator:**

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, Either a quartz

Crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably.

****

Chapter-4

Fig 4.2 Crystal Oscillator

This mode has a limited frequency range and it cannot be used to drive other clock buffers. For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. For ceramic resonators, the capacitor values given by the manufacturer should be used. The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1

**4.1.1.3 Architecture:**

**Memory:** It has **8 Kb** of Flash program memory (10,000 Write/Erase cycles durability), **512 Bytes** of EEPROM (100,000 Write/Erase Cycles). **1Kbyte** Internal SRAM

**I/O Ports:** 23 I/ line can be obtained from three ports; namely Port B, Port C and Port D.

**Interrupts:**  Two External Interrupt source, located at port D. 19 different interrupt vectors supporting 19 events generated by internal peripherals.

**Timer/Counter:** Three Internal Timers are available, two 8 bit, one 16 bit, offering various operating modes and supporting internal or external clocking.

Chapter-4

**SPI (Serial Peripheral interface):** ATmega8 holds three communication devices integrated. One of them is Serial Peripheral Interface. Four pins are assigned to Atmega8 to implement this scheme of communication.

**USART:**One of the most powerful communication solutions is [USART](http://www.circuitstoday.com/how-to-establish-a-pc-micro-controller-usart-communication) and ATmega8 supports both synchronous and asynchronous data transfer schemes. It has three pins assigned for that. In many projects, this module is extensively used for PC-Micro controller communication.

**TWI (Two Wire Interface):** Another communication device that is present in ATmega8 is Two Wire Interface. It allows designers to set up a commutation between two devices using just two wires along with a common ground connection, As the TWI output is made by means of open collector outputs, thus external pull up resistors are required to make the circuit.

**Analog Comparator:** A comparator module is integrated in the IC that provides comparison facility between two voltages connected to the two inputs of the Analog comparator via External pins attached to the micro controller.

**Analog to Digital Converter:**Inbuilt analog to digital converter can convert an analog input signal into digital data of **10bit** resolution. For most of the low end application, this much resolution is enough.

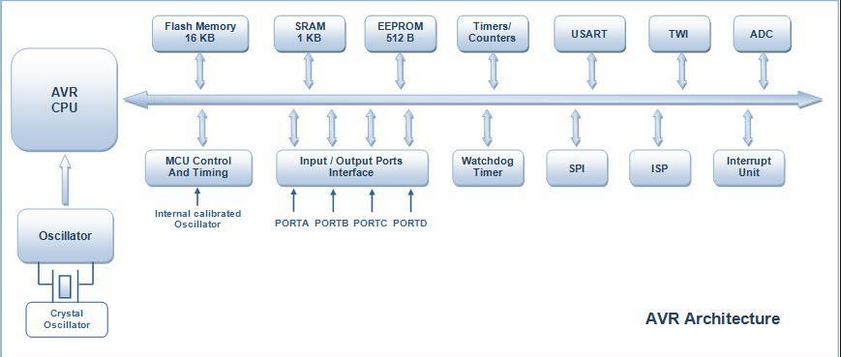


Fig 4.3 AVR Architecture

Chapter-4

**4.1.1.4 Microcontroller:**

 Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, Timers etc., required to perform some predefined task.

The computer on one hand is designed to perform all the general purpose tasks on a single machine like you can use a computer to run a software to perform calculations or you can use a computer to store some multimedia file or to access [internet](http://www.engineersgarage.com/articles/what-is-internet-history-working) through the browser, whereas the microcontrollers are meant to perform only the specific tasks, for e.g., switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

 There are number of popular families of microcontrollers which are used in different applications as per their capability and feasibility to perform the desired task, most common of these are [8051](http://www.engineersgarage.com/8051-microcontroller), **AVR** and [PIC](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial) microcontrollers. In this article we will introduce you with **AVR** family of microcontrollers.

**AVR** was developed in the year 1996 by Atmel Corporation. The architecture of **AVR** was developed by Alf-Egil Bogen and Vegard Wollan. AVR derives its name from its developers and stands for **Alf-Egil Bogen Vegard Wollan RISC microcontroller**, also known as **A**dvanced **V**irtual **R**ISC. The AT90S8515 was the first microcontroller which was based on **AVR architecture** however the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

**AVR microcontrollers** are available in three categories:

1.      **TinyAVR** – Less memory, small size, suitable only for simpler applications

2.      **MegaAVR** – These are the most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.

3.      **XmegaAVR** – Used commercially for complex applications, which require large program memory and high speed.

**4.1.1.4 ARCHITECTURE**

Chapter-4

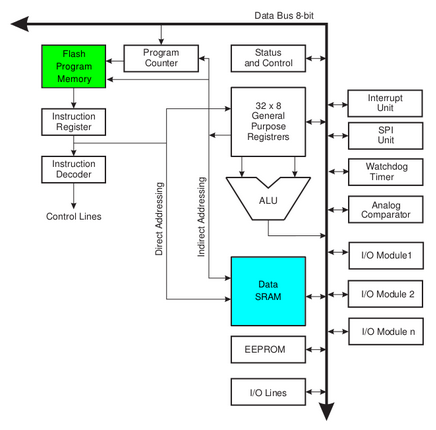


Fig 4.4 Internal Architecture

### Device architecture

Flash, EEPROM, and SRAM are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

#### Program memory

Program instructions are stored in non-volatile flash memory. Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.

The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash while the ATmega32x line has 32 kB).

Chapter-4

There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

#### Internal data memory

The data address space consists of the register file, I/O registers, and SRAM.

#### Internal registers

The AVRs have 32 single-byte registers and are classified as 8-bit RISC devices.

In most variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (000016–001F16) followed by the 64 I/O registers (002016–005F16).

Actual SRAM starts after these register sections (address 006016). (Note that the I/O register space may be larger on some more extensive devices, in which case the memory mapped I/O registers will occupy a portion of the SRAM address space.)

Even though there are separate addressing schemes and optimized opcodes for register file and I/O register access, all can still be addressed and manipulated as if they were in SRAM.

In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (000016–0FFF16). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space which can be used optionally for mapping the internal EEPROM to the data address space (100016–1FFF16). The actual SRAM is located after these ranges, starting at 200016.

#### EEPROM

Chapter-4

Almost all AVR microcontrollers have internal EEPROM for semi-permanent data storage. Like flash memory, EEPROM can maintain its contents when electrical power is removed.

In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions which makes EEPROM access much slower than other internal RAM.

However, some devices in the SecureAVR (AT90SC) family use a special EEPROM mapping to the data or program memory depending on the configuration. The XMEGA family also allows the EEPROM to be mapped into the data address space.

Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well designed EEPROM write routine should compare the contents of an EEPROM address with desired contents and only perform an actual write if the contents need to be changed.

Note that erase and write can be performed separately in many cases, byte-by-byte, which may also help prolong life when bits only need to be set to all 1s (erase) or selectively cleared to 0s (write).

### Program execution

Atmel's AVRs have a two stage, single level pipeline design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVRs relatively fast among eight-bit microcontrollers.

The AVR processors were designed with the efficient execution of compiled C code in mind and have several built-in pointers for the task.

**MCU speed**

The AVR line can normally support clock speeds from 0 to 20 MHz, with some devices reaching 32 MHz. Lower powered operation usually requires a reduced clock speed. All recent (Tiny, Mega, and Xmega, but not 90S) AVRs feature an on-chip oscillator, removing the need for external clocks or resonator circuitry. Some AVRs also have a system clock prescaler that can divide down the system clock by up to 1024. This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.

Since all operations (excluding literals) on registers R0 - R31 are single cycle, the AVR can achieve up to 1 MIPS per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS. Loads and stores to/from memory take two cycles, branching takes two cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices

Chapter-4

**Features:**

• High-performance, Low-power Atmel®AVR® 8-bit Microcontroller

**• Advanced RISC Architecture**

– 130 Powerful Instructions – Most Single-clock Cycle Execution

– 32 × 8 General Purpose Working Registers

– Fully Static Operation

– Up to 16MIPS Throughput at 16MHz

– On-chip 2-cycle Multiplier

**• High Endurance Non-volatile Memory segments**

– 8Kbytes of In-System Self-programmable Flash program memory

– 512Bytes EEPROM

– 1Kbyte Internal SRAM

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

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

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

**• Peripheral Features**

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

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

Mode

Chapter-4

– Real Time Counter with Separate Oscillator

– Three PWM Channels

– 8-channel ADC in TQFP and QFN/MLF package

Eight Channels 10-bit Accuracy

– 6-channel ADC in PDIP package

Six Channels 10-bit Accuracy

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

**• Special Microcontroller Features**

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and

Standby

**• I/O and Packages**

– 23 Programmable I/O Lines

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

**• Operating Voltages**

– 2.7V - 5.5V (ATmega8L)

– 4.5V - 5.5V (ATmega8)

**• Speed Grades**

– 0 - 8MHz (ATmega8L)

– 0 - 16MHz (ATmega8)

**• Power Consumption at 4Mhz, 3V, 25oC**

– Active: 3.6mA

– Idle Mode: 1.0mA

Chapter-4

– Power-down Mode: 0.5µA

**Brown-out Detector:**

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Brown-out Detection” on page 38 for details on how to configure the Brown-out Detector.

Internal Voltage Reference the Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to “Internal Voltage Reference” on page 40 for details on the start-up time. Watchdog Timer If the Watchdog Timer is not needed in the application, this module should be turned off.

If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Watchdog Timer” on page 41 for details on how to configure the Watchdog Timer. Port Pins When entering a sleep mode, all port pins should be configured to use minimum power.

The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock (clkI/O) and the ADC clock (clkADC) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section “Digital Input Enable and Sleep Modes” on page 53 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to VCC/2, the input buffer will use excessive power.

**Power-on Reset:**

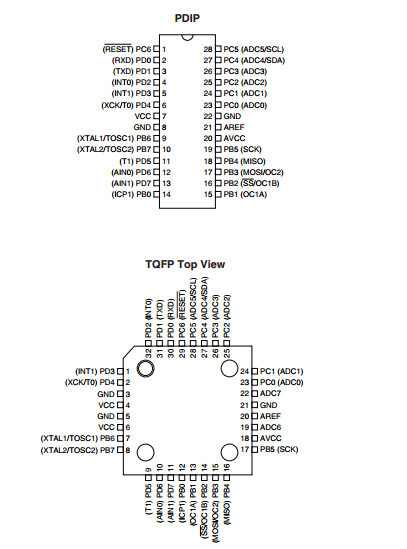
A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever VCC is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

Chapter-4

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after VCC rise. The RESET signal is activated again, without any delay, when VCC decreases below the detection level.

**External Reset**:

An External Reset is generated by a low level on the RESET pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – VRST on its positive edge, the delay counter starts the MCU after the time-out period tTOUT has expired.



Chapter-4

**3.2.3Pin diagram:**

Fig.3.2.4.PIN DIAGRAM OF ATMEGA8

**VCC**  
 Digital supply voltage magnitude of the voltage range between 4.5 to 5.5 V for the ATmega8 and 2.7 to 5.5 V for ATmega8L

Chapter-4

**GND**  
 Ground Zero reference digital voltage supply.

**PORTB (PB7.. PB0)**

PORTB is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTB pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**PORTC (PC5.. PC0)**

PORTC is a port I / O two-way (bidirectional) 7-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTC pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**PC6/RESET**  
 If RSTDISBL Fuse programmed, PC6 then serves as a pin I / O but with different characteristics. PC0 to PC5 If Fuse RSTDISBL not programmed, then serves as input Reset PC6. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running.

**PORTD (PD7.. PD0)**

PORTD is a port I / O two-way (bidirectional) 8-bit with internal pull-up resistor can be selected. This port output buffers have symmetrical characteristics when used as a source or sink. When used as an input, the pull-pin low externally will emit a current if the pull-up resistor is activated it. PORTD pins will be in the condition of the tri-state when RESET is active, although the clock is not running.

**RESET** Reset input pin. LOW signal on this pin with a minimum width of 1.5 microseconds will bring the microcontroller into reset condition, although the clock is not running. Signal with a width of less than 1.5 microseconds does not guarantee a Reset condition.

**AVCC**  
 AVCC is the supply voltage pin for the ADC, PC3 .. PC0, and ADC7..ADC6. This pin should be connected to VCC, even if the ADC is not used. If the ADC is used, AVCC should be connected to VCC through a low-pass filter to reduce noise.

Chapter-4

**Aref**  
 Analog Reference pin for the ADC.

**ADC7 .. ADC6**

ADC analog input there is only on ATmega8 with TQFP and QFP packages / MLF.

**PORTS**

Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status. Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller. Ports represent physical connection of Central Processing Unit with an outside world. Microcontroller uses them

The [Atmega8](http://www.protostack.com/product_by_model.php?model=IC-ATMEGA8-16PU) has 23 I/O ports which are organized into 3 groups:

* Port B (PB0 to PB7)
* Port C (PC0 to PC6)
* Port D (PD0 to PD7)

We will use mainly 3 registers known as **DDRX, PORTX**&**PINX**. We have total four PORTs on my ATmega16. They are **PORTA, PORTB, PORTC** and **PORTD**. They are multifunctional pins. Each of the pins in each port (total 32) can be treated as input or output pin.

**Applications**

Chapter-4

AVR microcontroller perfectly fits many uses, from automotive industries and controlling home appliances to industrial instruments, remote sensors, electrical door locks and safety devices. It is also ideal for smart cards as well as for battery supplied devices because of its low consumption.

EEPROM memory makes it easier to apply microcontrollers to devices where permanent storage of various parameters is needed (codes for transmitters, motor speed, receiver frequencies, etc.). Low cost, low consumption, easy handling and flexibility make ATmega8 applicable even in areas where microcontrollers had not previously been considered (example: timer functions, interface replacement in larger systems, coprocessor applications, etc.).

In System Programmability of this chip (along with using only two pins in data transfer) makes possible the flexibility of a product, after assembling and testing have been completed. This capability can be used to create assembly-line production, to store calibration data available only after final testing, or it can be used to improve programs on finished products.

### 4.1.2 Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

Chapter-4

* **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
* **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
* **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* **GND.** Ground pins.
* **IOREF.** This pin on the Arduino 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.

### 4.1.3 Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](http://www.arduino.cc/en/Reference/EEPROM)).

### 4.1.4 Input and Output

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

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

Chapter-4

* **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()](http://arduino.cc/en/Reference/AttachInterrupt) function for details.
* **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite ()](http://arduino.cc/en/Reference/AnalogWrite) function.
* **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the spi library.
* **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

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

* **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the [Wire library](http://arduino.cc/en/Reference/Wire).

There are a couple of other pins on the board:

* **AREF.** Reference voltage for the analog inputs. Used with [analogReference](http://arduino.cc/en/Reference/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. The mapping for the Atmega8, 168, and 328 is identical.

### 4.1.5 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Chapter-4

### 4.1.6 Programming

The Arduino Uno can be programmed with the Arduino software ,Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board).

The ATmega328 on the Arduino Uno comes preburned with a [boot loader](http://arduino.cc/en/Tutorial/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](http://www.atmel.com/dyn/resources/prod_documents/doc2525.pdf), [C header files](http://www.atmel.com/dyn/resources/prod_documents/avr061.zip)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [Arduino ISP](http://arduino.cc/en/Main/ArduinoISP) or similar; s

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

* On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
* On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use [Atmel's FLIP software](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886) (Windows) or the [DFU programmer](http://dfu-programmer.sourceforge.net/) (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)..

Chapter-4

### 4.1.7 Automatic (Software) Reset

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

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line

Chapter-4

Chapter-4

### 4.1.8 USB Over current Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. 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.

### 4.1.9 Physical Characteristics

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

**4.1.10 Actuators (l293d)**

The Actuator's are those devices which actually gives the movement or to do a task like motor's. In the real world there are various types of motors available which works on different voltages. So we need motor driver for running them through the controller. To get interface between motor and micro controller . We use L293D motor driver IC in our circuit.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

Chapter-4

|  |
| --- |
|  |

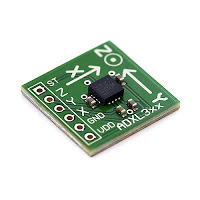
**4.2 ACCELEROMETER**

An Accelerometer is a kind of sensor which gives an analog data while moving in X,Y,Z direction or may be X,Y direction only depend's on the type of the sensor.Here is a small image of an Accelerometer shown. We can see in the image that their are some arrow showing if we tilt these sensor's in that direction then the data at that corresponding pin will change in the analog form.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The Accelerometer having 6 pin-

Chapter-4

[](http://3.bp.blogspot.com/-geileThx7nU/T5xoWCFwt6I/AAAAAAAAAXg/MV4PVsmgLIg/s1600/5041671737_a17bcfeb6a.jpg)

1- VDD- We will give the +5volt to this pin

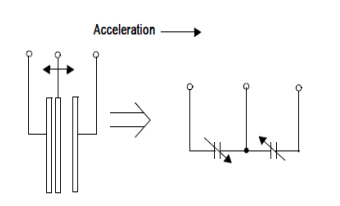
2- GND- We simply connect this pin to the ground for biasing.

3- X- On this pin we will receive the analog data for x direction movement.

4- Y- On this pin we will receive the analog data for y direction movement.

5- Z-  On this pin we will receive the analog data for z direction movement.

6- ST- this pin is use to set the sensitivity of the accelerometer 1.5g/2g/3g/4g.



Chapter-4

Figure 4.2.1 Simplified transducer module

The g-cell is a mechanical structure formed from semiconductor materials (polysilicon) using semiconductor processes (masking and etching). It can be modeled as a set of beams attached to a movable central mass that move between fixed beams. The movable beams can be deflected from their rest position by subjecting the system to an acceleration (Figure 3). As the beams attached to the central mass move, the distance from them to the fixed beams on one side will increase by the same amount that the distance to the fixed beams on the other side decreases. The change in distance is a measure of acceleration. The g-cell beams form two back-to-back capacitors. As the center beam moves with acceleration, the distance between the beams changes and each capacitor's value will change, (C = Aε/D). Where A is the area of the beam, ε is the dielectric constant, and D is the distance between the beams. IC uses switched capacitor techniques to measure the g-cell capacitors and extract the acceleration data from the difference between the two capacitors. IC also signal conditions and filters (switched capacitor) the signal, providing a high level output voltage that is ratio metric and proportional to acceleration

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Chapter-4

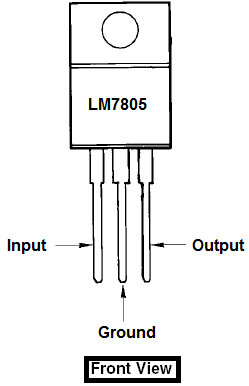
**4.3 POWER SUPPLY**

**4.3.1 Description**

A variable regulated power supply, also called a variable bench power supply, is one where you can continuously adjust the output voltage to your requirements. Varying the output of the power supply is the recommended way to test a project after having double checked parts placement against circuit drawings and the parts placement guide.

This type of regulation is ideal for having a simple variable bench power supply. Actually this is quite important because one of the first projects a hobbyist should undertake is the construction of a variable regulated power supply. While a dedicated supply is quite handy e.g. 5V or 12V, it's much handier to have a variable supply on hand, especially for testing.

Most digital logic circuits and processors need a 5 volt power supply. To use these parts we need to build a regulated 5 volt source. Usually you start with an unregulated power supply ranging from 9 volts to 24 volts DC To make a 5 volt power supply, we use a LM7805 voltage regulator IC (Integrated Circuit). The IC is shown below.



Chapter-4

Figure 4.4.1 power supply

The LM7805 is simple to use. You simply connect the positive lead of your unregulated DC power supply (anything from 9VDC to 24VDC) to the Input pin, connect the negative lead to the common pin and then when you turn on the power, you get a 5 volt supply from the Output pin.

**4.4 HAND HELD UNIT**

In this unit we are using a Avr series of microcontroller which is having inbuilt ADC. Accelerometer XY signals are applied to the ADC input of microcontroller for conversion of analog signal into digital form .Then these signal are compared with a threshold value using programming and transmitted in a form of 4 bit data signals to a RF encoder IC HT 12E.This IC converts this 4 bit information into serial form by which it can be transmitted by a RF transmitter.

# 4.5 SERIAL COMMUNICATION

Chapter-4

In order to make two devices communicate, whether they are desktop computers, microcontrollers, or any other form of integrated circuit, we need a method of communication and an agreed-upon language. The most common form of communication between electronic devices is serial communication. Communicating serially involves sending a series of digital pulses back and forth between devices at a mutually agreed-upon rate. The sender sends pulses representing the data to be sent at the agreed-upondata rate, and the receiver listens for pulses at that same rate. This is what’s known as asynchronous serial communication*.* There isn’t one common clock in asynchronous serial communication; instead, both devices have their own clock and agree on a rate to which to set their clocks.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

For example, let’s say two devices are to exchange data at a rate of 9600 bits per second. First, we would make three connections between the two devices:

Chapter-4

* a common ground connection, so both devices have a common reference point to measure voltage by;
* one wire for the sender to send data to the receiver on (transmit line for the sender);
* one wire for the receiver to send date to the sender on (receive line for the sender).

Now, since the data rate is 9600 bits per second (sometimes called 9600 baud), the receiver will continually read the voltage that the sender is putting out, and every 1/9600th of a second, it will interpret that voltage as a new bit of data. If the voltage is high (+5V in the case of Wiring/Arduino, the PIC, and BX-24), it will interpret that bit of data as a 1. If it is low (0V in the case of Wiring/Arduino, the PIC, and BX-24), it will interpret that bit of data as a 0. By interpreting several bits of data over time, the receiver can get a detailed message from the sender. at 9600 baud, for example, 1200 bytes of data can be exchanged in one second. If you have a home computer and a modem, you’ve seen serial communication in action. Your computer’s modem exchanges information with your service provider’s modem serially.

Let’s look at a byte of data being exchanged. Imagine I want to send the number 90 from one device to another. First, I have to convert the number from the decimal representation 90 to a binary representation. in binary, 90 is 01011010. So my sending device will pulse its transmit line as follows:

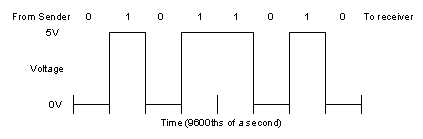


Figure 4.5.1 sender and receiver bits

As you can tell from this diagram, both devices also have to agree on the order of the bits. Usually the sender sends the highest bit (or most significant bit) first in time, and the lowest (or least significant bit) last in time. As long as we have an agreed upon voltage, data rate, and order of interpretation of bits, we can exchange any data we want serially.

Chapter-4

## Serial Communication to a PC or Mac

The RS-232 serial ports on Windows-based PC’s looks like this:

PC serial cable (facing the soldering lugs of a female connector)

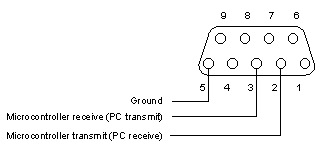


Figure 4.6.2 pc serial cable

The connections for this are the same as the connector used to program the BX-24, and to debug from both the PIC and the BX-24, so you can use the same connector, or make a second one just like it. From there, you will need a 9-pin serial cable. Use any serial port you want, but be sure that all other programs that might be trying to use the serial port are turned off.

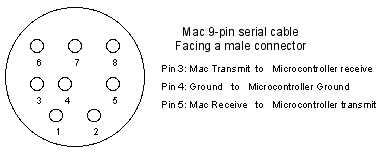
It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

Wiring and Arduino boards have a built-in USB-to-serial converter, so they communicate serially using the USB port.

Chapter-4

Macintoses used to have two serial ports, one for the printer, one for the modem. The current Mac models (and increaslingly, many PCs) do not come with a built-in serial port, and one must be installed. An RS-232 to USB serial port adaptor such as the [Keyspan](http://www.keyspan.com/products/USB/usa28x/)USB USA19HS Adaptor will do fine.

## **Classic Mac serial cable (Facing the male pins of a cable)**

figure 4.6.3 mac 9-pin serial cable

### Serial Output from the Microcontroller

Once you’ve got the computer and themicrocontroller connected, you’ll need to write a program to address the serial ports. The process is slightly different on the different microcontrollers, but there are some elements common to all of them.

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

Desktop computers have a place they store incoming data from the serial port, called a serial buffer. It’s a little area of memory to store whatever comes in the serial port. Because of this, they can do other tasks while waiting for data to come in, and act on the data from the buffer. Microcontrollers usually don’t have a serial buffer, but the BX-24 can have a small one, as do Wiring and Arduino. Details on how to set it up on the BX-24 follow. It’s automatically set up for you on a Wiring or Arduino module. On the PIC, you don’thave a serial buffer, so you don’t have to set it up.

Multimedia computers have one or more serial ports, and each port can be controlled by only one program at a time. To use a port, you have to open it, set its parameters, and then look for data. On the PIC, there are no serial ports. Each serial command defines the parameters of communication within the command. On the BX-24, you have to define the port, open it, then look for data, as on the PC.

Chapter-4

On Wiring or Arduino boards, the serial pins are fixed, and you can’t change them. You can, however, use the [software serial library for Arduino](http://www.arduino.cc/en/Tutorial/SoftwareSerial)if you need to use different pins, or if you need more than one serial port.

See the[PIC serial notes](http://www.tigoe.net/pcomp/code/pic/pic-serial-pbpro.shtml) for the software details of serial output using PicBasic Pro. See the [BX-24 serial notes](http://www.tigoe.net/pcomp/code/bx/bx-serial.shtml)for the software details using a BX-24. See the [Arduino serial lab](http://itp.nyu.edu/physcomp/Labs/Serial) for more on serial on Arduino.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Chapter-4

If you’re sending a single byte out serially, it’s easy for the receiving device to know what’s coming. It will read a byte and know the value. Sometimes, however, you have to send multiple bytes to get a message across. Those bytes have to be reassembled on the receiving side. More on that in the [serial data interpretation notes](http://www.tigoe.net/pcomp/code/serialdata.shtml)

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.

**DCE and DTE Devices**

Two terms you should be familiar with are DTE and DCE. DTE stands for Data Terminal Equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices are usually DCE devices.

If you have trouble keeping the two straight then replace the term "DTE device" with "your PC" and the term "DCE device" with "remote device" in the following discussion

The RS-232 standard states that DTE devices use a 25-pin male connector, and DCE devices use a 25-pin female connector. You can therefore connect a DTE device to a DCE using a straight pin-for-pin connection. However, to connect two like devices, you must instead use a null modem cable. Null modem cables cross the transmit and receive lines in the cable, and are discussed later in this chapter.

The listing below shows the connections and signal directions for both 25 and 9-pin connectors.

Chapter-4

|  |  |
| --- | --- |
| Male RS232 DB25 | * 25 pin to 9 pin connector |
| * **Pin Number** | * **Direction of signal:** |
| * 1 | * Protective Ground |
| * 2 | * Transmitted Data (TD) Outgoing Data (from a DTE to a DCE) |
| * 3 | * Received Data (RD) Incoming Data (from a DCE to a DTE) |
| * 4 | * Request To Send (RTS) Outgoing flow control signal controlled by DTE |
| * 5 | * Clear To Send (CTS) Incoming flow control signal controlled by DCE |
| * 6 | * Data Set Ready (DSR) Incoming handshaking signal controlled by DCE |
| * 7 | * Signal Ground Common reference voltage |
| * 8 | * Carrier Detect (CD) Incoming signal from a modem |
| * 20 | * Data Terminal Ready (DTR) Outgoing handshaking signal controlled by DTE |
| * 22 | * Ring Indicator (RI) Incoming signal from a modem |

Chapter-4

**5. SOFTWARE**

Chapter-5

5.1 ABOUT ARDUINO SOFTWARE

The Arduino [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is a [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) application written in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and derives from the IDE for the [Processing programming language](http://en.wikipedia.org/wiki/Processing_(programming_language)) and the [Wiring](http://en.wikipedia.org/wiki/Wiring_(development_platform)) projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as [syntax highlighting](http://en.wikipedia.org/wiki/Syntax_highlighting), [brace matching](http://en.wikipedia.org/wiki/Brace_matching), and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a sketch.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Arduino programs are written in [C](http://en.wikipedia.org/wiki/C_(programming_language)) or [C++](http://en.wikipedia.org/wiki/C%2B%2B). The Arduino IDE comes with a [software library](http://en.wikipedia.org/wiki/Software_library) called "[Wiring](http://en.wikipedia.org/wiki/Wiring_(development_platform))" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable[cyclic executive](http://en.wikipedia.org/wiki/Cyclic_executive) program:

Chapter-5

* setup(): a function run once at the start of a program that can initialize settings
* loop(): a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an [LED](http://en.wikipedia.org/wiki/Light-emitting_diode) on and off. In the Arduino environment, the user might write a program like this:[[16]](http://en.wikipedia.org/wiki/Arduino#cite_note-Blink_Tutorial-16)

[](http://en.wikipedia.org/wiki/File:Arduino_led-5.jpg)

The integrated pin 13 LED

#define LED\_PIN 13

void setup () {

pinMode (LED\_PIN, OUTPUT); // Enable pin 13 for digital output

}

void loop () {

digitalWrite (LED\_PIN, HIGH); // Turn on the LED

delay (1000); // Wait one second (1000 milliseconds)

digitalWrite (LED\_PIN, LOW); // Turn off the LED

delay (1000); // Wait one second

}

Arduino is an [open-source](http://en.wikipedia.org/wiki/Open-source) computer hardware and software company, project and user community that designs and manufactures kits for building digital devices and interactive objects that can sense and control the physical world. Arduino boards may be purchased preassembled, or as [do-it-yourself](http://en.wikipedia.org/wiki/Do-it-yourself) kits; at the same time, the hardware design information is available for those who would like to assemble an Arduino from scratch.

Chapter-5

The project is based on a family of [microcontroller](http://en.wikipedia.org/wiki/Microcontroller) board designs manufactured primarily by SmartProjects in Italy,[[2]](http://en.wikipedia.org/wiki/Arduino#cite_note-2) and also by several other vendors, using various 8-bit [Atmel](http://en.wikipedia.org/wiki/Atmel) [AVR](http://en.wikipedia.org/wiki/Atmel_AVR) microcontrollers or 32-bit Atmel [ARM](http://en.wikipedia.org/wiki/AT91SAM) processors. These systems provide sets of digital and analog [I/O](http://en.wikipedia.org/wiki/I/O) pins that can be interfaced to various extension boards and other circuits. The boards feature serial communications interfaces, including [USB](http://en.wikipedia.org/wiki/USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino platform provides an [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) based on the [Processing](http://en.wikipedia.org/wiki/Processing_(programming_language)) project, which includes support for [C](http://en.wikipedia.org/wiki/C_programming_language) and [C++](http://en.wikipedia.org/wiki/C%2B%2B_programming_language) programming languages.

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Chapter-5

The first Arduino was introduced in 2005. The project leaders sought to provide an inexpensive and easy way for hobbyists, students, and professionals to create devices that interact with their environment using [sensors](http://en.wikipedia.org/wiki/Sensors) and [actuators](http://en.wikipedia.org/wiki/Actuators). Common examples for beginner hobbyists include simple [robots](http://en.wikipedia.org/wiki/Robots), [thermostats](http://en.wikipedia.org/wiki/Thermostats) and motion detectors. [Adafruit Industries](http://en.wikipedia.org/wiki/Adafruit_Industries) estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

Chapter-5

An Arduino board consists of an [Atmel](http://en.wikipedia.org/wiki/Atmel) 8-bit AVR [microcontroller](http://en.wikipedia.org/wiki/Microcontroller) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an [I²C](http://en.wikipedia.org/wiki/I%C2%B2C) [serial bus](http://en.wikipedia.org/wiki/Serial_bus)—so many shields can be stacked and used in parallel. Official Arduinos have used the [megaAVR](http://en.wikipedia.org/wiki/MegaAVR) series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt [linear regulator](http://en.wikipedia.org/wiki/Linear_regulator) and a 16 MHz [crystal oscillator](http://en.wikipedia.org/wiki/Crystal_oscillator) (or [ceramic resonator](http://en.wikipedia.org/wiki/Ceramic_resonator) in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a [boot loader](http://en.wikipedia.org/wiki/Boot_loader)that simplifies uploading of programs to the on-chip [flash memory](http://en.wikipedia.org/wiki/Flash_memory), compared with other devices that typically need an external[programmer](http://en.wikipedia.org/wiki/Programmer_(hardware)). This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

Chapter-5

Chapter-5

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an [RS-232](http://en.wikipedia.org/wiki/RS-232) serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a level shifter circuit to convert between RS-232-level and[TTL](http://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic)-level signals. Current Arduino boards are programmed via [USB](http://en.wikipedia.org/wiki/Universal_Serial_Bus), implemented using USB-to-serial adapter chips such as the [FTDI](http://en.wikipedia.org/wiki/FTDI)FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable,[Bluetooth](http://en.wikipedia.org/wiki/Bluetooth) or other methods. (When used with traditional microcontroller tools instead of the Arduino [IDE](http://en.wikipedia.org/wiki/Integrated_development_environment), standard AVR [ISP](http://en.wikipedia.org/wiki/In-system_programming) programming is used.)

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce [pulse-width modulated](http://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into [solderless breadboards](http://en.wikipedia.org/wiki/Solderless_breadboard).

Chapter-5

Chapter-5

There are many Arduino-compatible and Arduino-derived boards. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education to simplify the construction of buggies and small robots. Others are electrically equivalent but change the form factor—sometimes retaining compatibility with shields, sometimes not. Some variants use completely different processors, with varying levels of compatibility.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground; a convenient feature for many simple tests.The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program.

Chapter-5

Chapter-5

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce [pulse-width modulated](http://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into [solderless breadboards](http://en.wikipedia.org/wiki/Solderless_breadboard).

Chapter-5

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce [pulse-width modulated](http://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into [solderless breadboards](http://en.wikipedia.org/wiki/Solderless_breadboard).

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce [pulse-width modulated](http://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into [solderless breadboards](http://en.wikipedia.org/wiki/Solderless_breadboard).

A [SoftwareSerial library](http://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus

The Arduino IDE uses the [GNU toolchain](http://en.wikipedia.org/wiki/GNU_toolchain) and AVR Libc to compile programs, and uses avrdude to upload programs to the board.

Chapter-5

As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

**5.1 SOURCE CODE :**

Chapter-5

**const int analogInPin1 = A0; // Analog input pin that the potentiometer is attached to**

**const int analogInPin2 = A1;**

**const int analogOutPin =13; // Analog output pin that the LED is attached to**

**int sensorValue1 = 0; // value read from the pot**

**int sensorValue2 = 0; // value output to the PWM (analog out)void setup() {**

**pinMode(13, OUTPUT);**

**pinMode(8, OUTPUT);**

**pinMode(9, OUTPUT);**

**pinMode(10, OUTPUT);**

**pinMode(11, OUTPUT);**

**}**

**void loop() {**

Chapter-5

**// read the analog in value:**

**sensorValue1 = analogRead(analogInPin1);**

**sensorValue2 = analogRead(analogInPin2); if(sensorValue1 > 400)**

**{**

**digitalWrite(8, HIGH);**

**digitalWrite(9, LOW);**

**digitalWrite(10, HIGH);**

**digitalWrite(11, LOW);**

**delay(1000);**

**}**

**else if(sensorValue1 < 300)**

**{**

Chapter-5

**digitalWrite(9, HIGH);**

**digitalWrite(8, LOW);**

**digitalWrite(11, HIGH);**

**digitalWrite(10, LOW);**

**delay(1000);**

**}**

**else if(sensorValue2 > 400)**

**{**

**digitalWrite(8, HIGH);**

**digitalWrite(9, LOW);**

**digitalWrite(11, HIGH);**

**digitalWrite(10, LOW);**

**delay(1000);**

Chapter-5

**}**

**else if(sensorValue2 < 300)**

**{**

**digitalWrite(9, HIGH);**

**digitalWrite(8, LOW);**

**digitalWrite(10, HIGH);**

**digitalWrite(11, LOW);**

**delay(1000);**

**}**

**else**

**{**

**digitalWrite(8, LOW);**

**digitalWrite(9, LOW);**

Chapter-5

**digitalWrite(11, LOW);**

**digitalWrite(10, LOW);**

**digitalWrite(13, HIGH);**

**delay(500);**

**digitalWrite(13, LOW);**

**}**

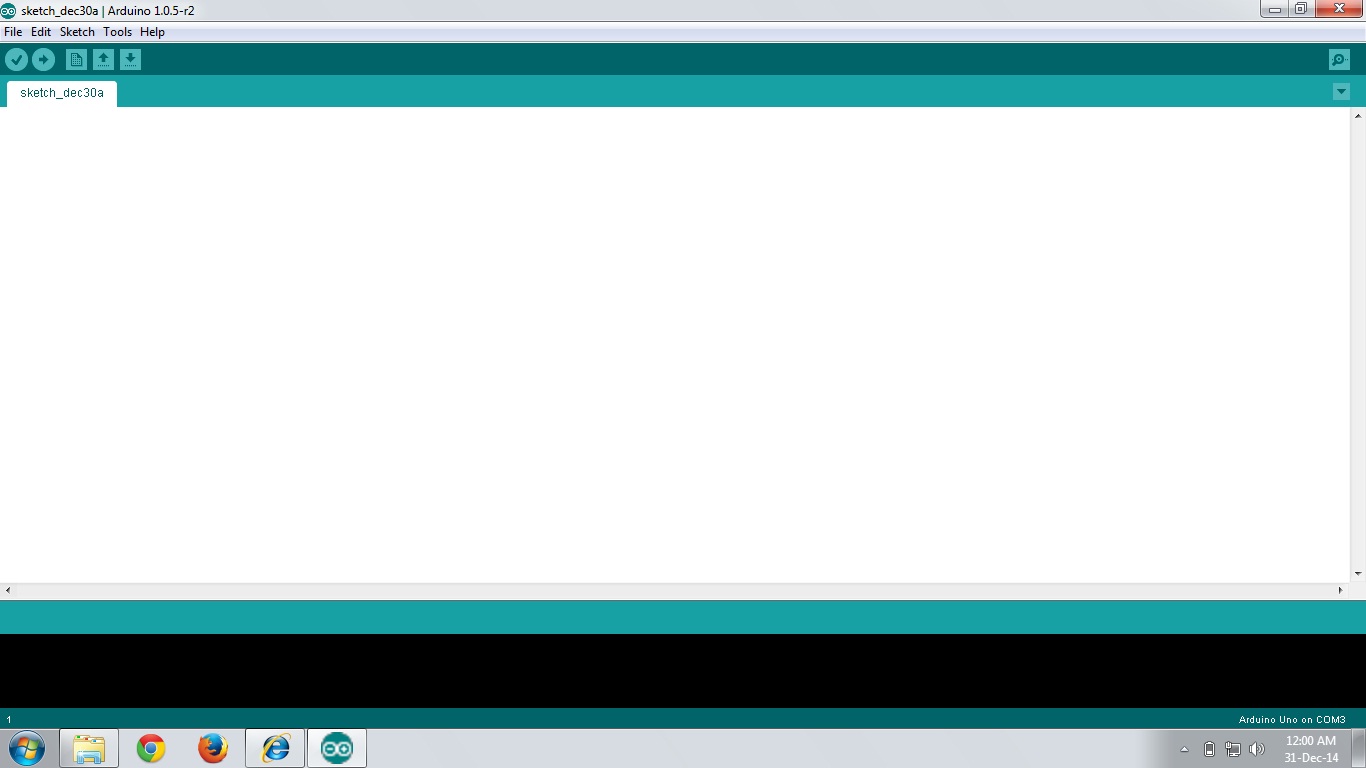
**}**

**5.2 PROCEDURE FOR INTERFACING SOFTWARE:**

Chapter-5

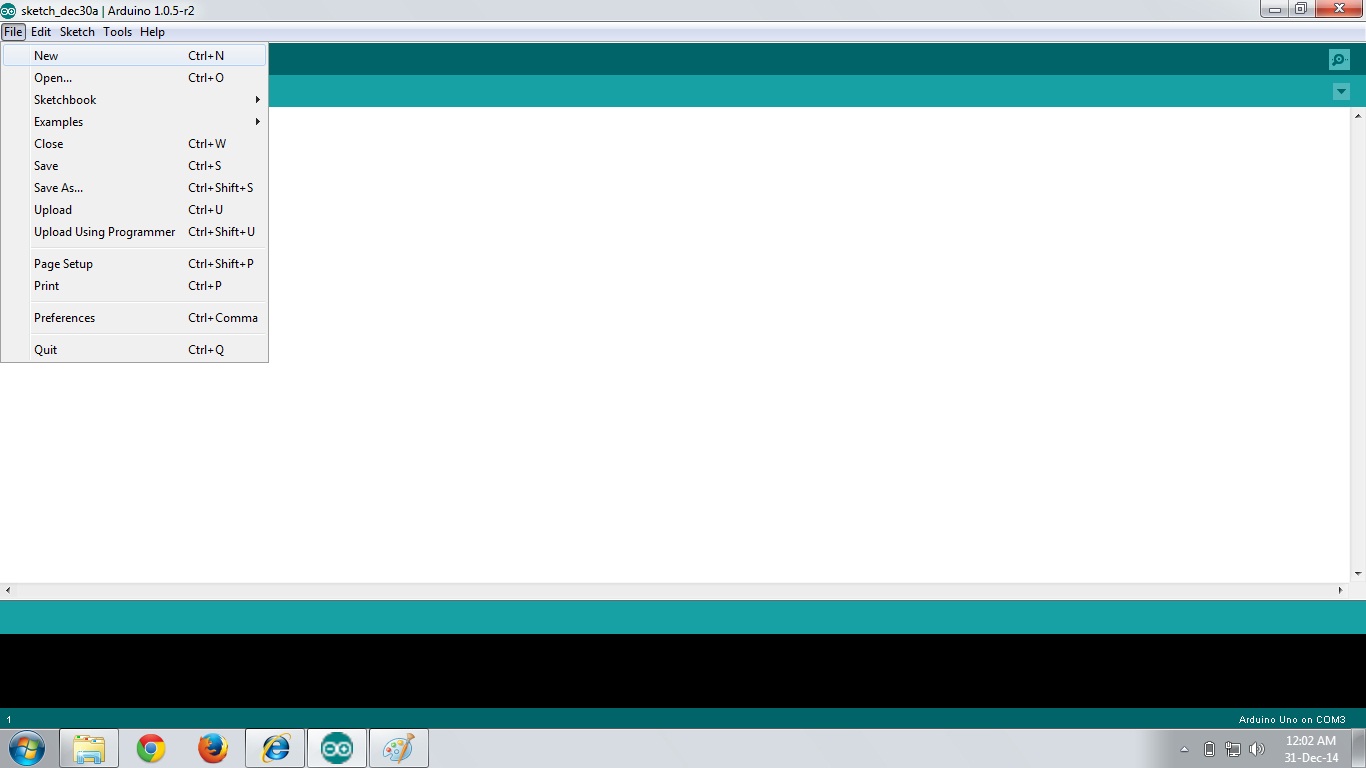
**To Setup the wireless network in transmitting and receiving data Using PC ,the following steps are followed.**

* 1. **Open the arduino IDE and click on file -> new**

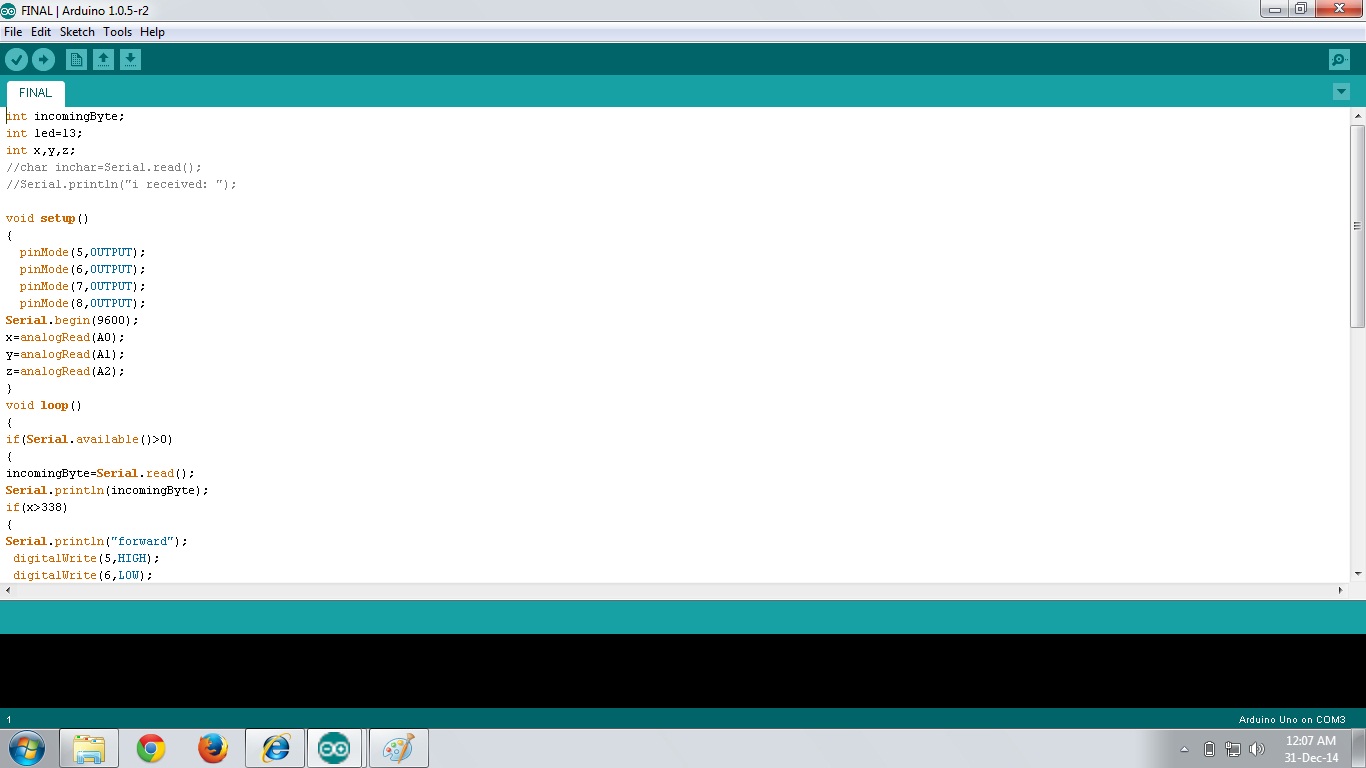
****

* 1. **Open and type the entire program click on file -> debug and file -> run**

Chapter-5

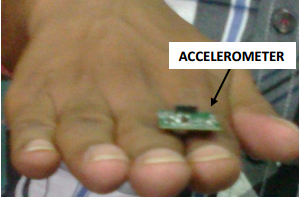
****

* 1. **See the output in the IDE channel for IDE program**

****

**Hand Gesture feedback and output detection**

Chapter-5

****

**The above figure shows the accelerometer mounted on hand**

****

**Different gestures to control the robot, clockwise forward , backward , left and right**

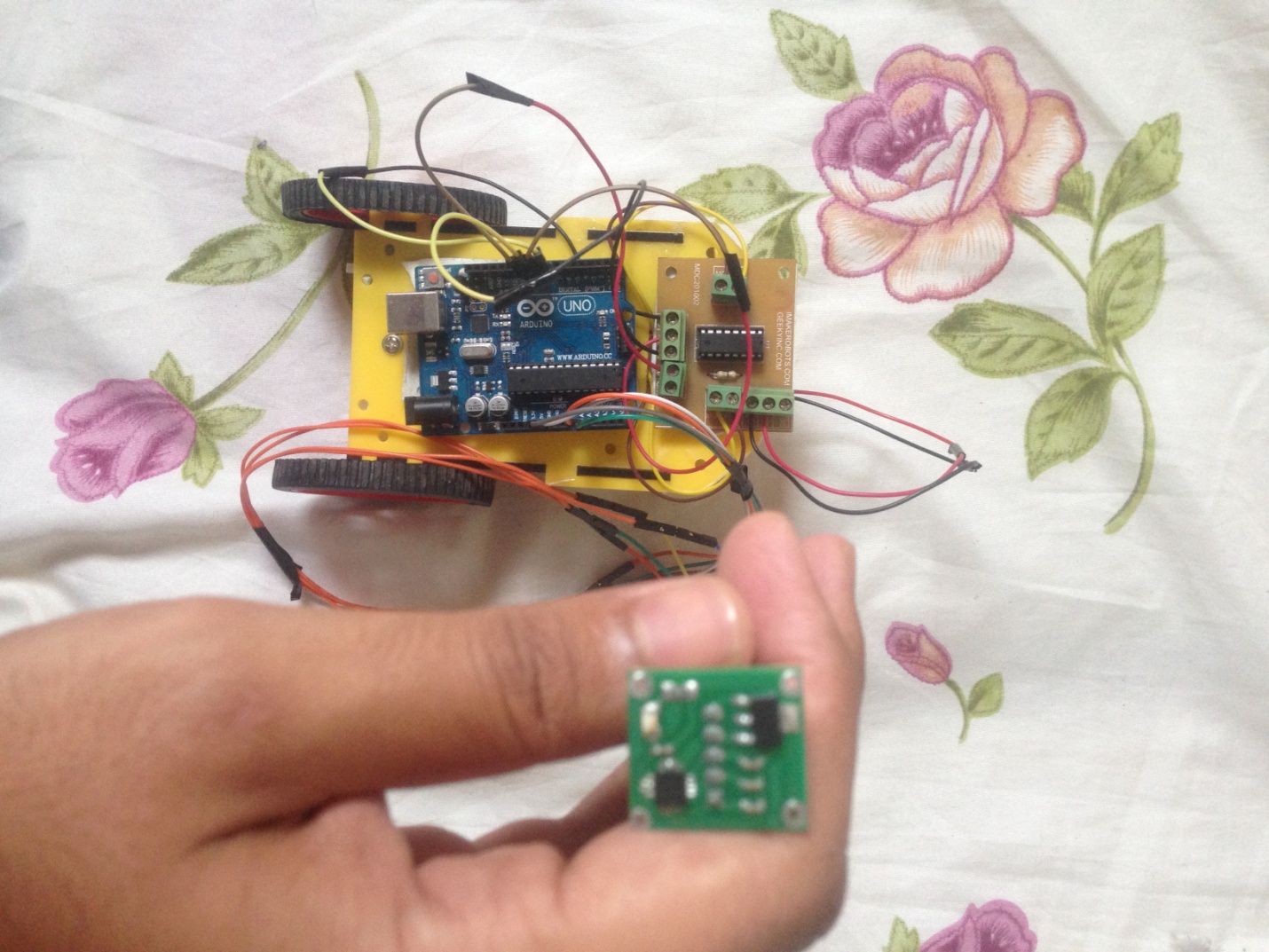
****

Chapter-5

**The accelerometer mounted upon the foot**

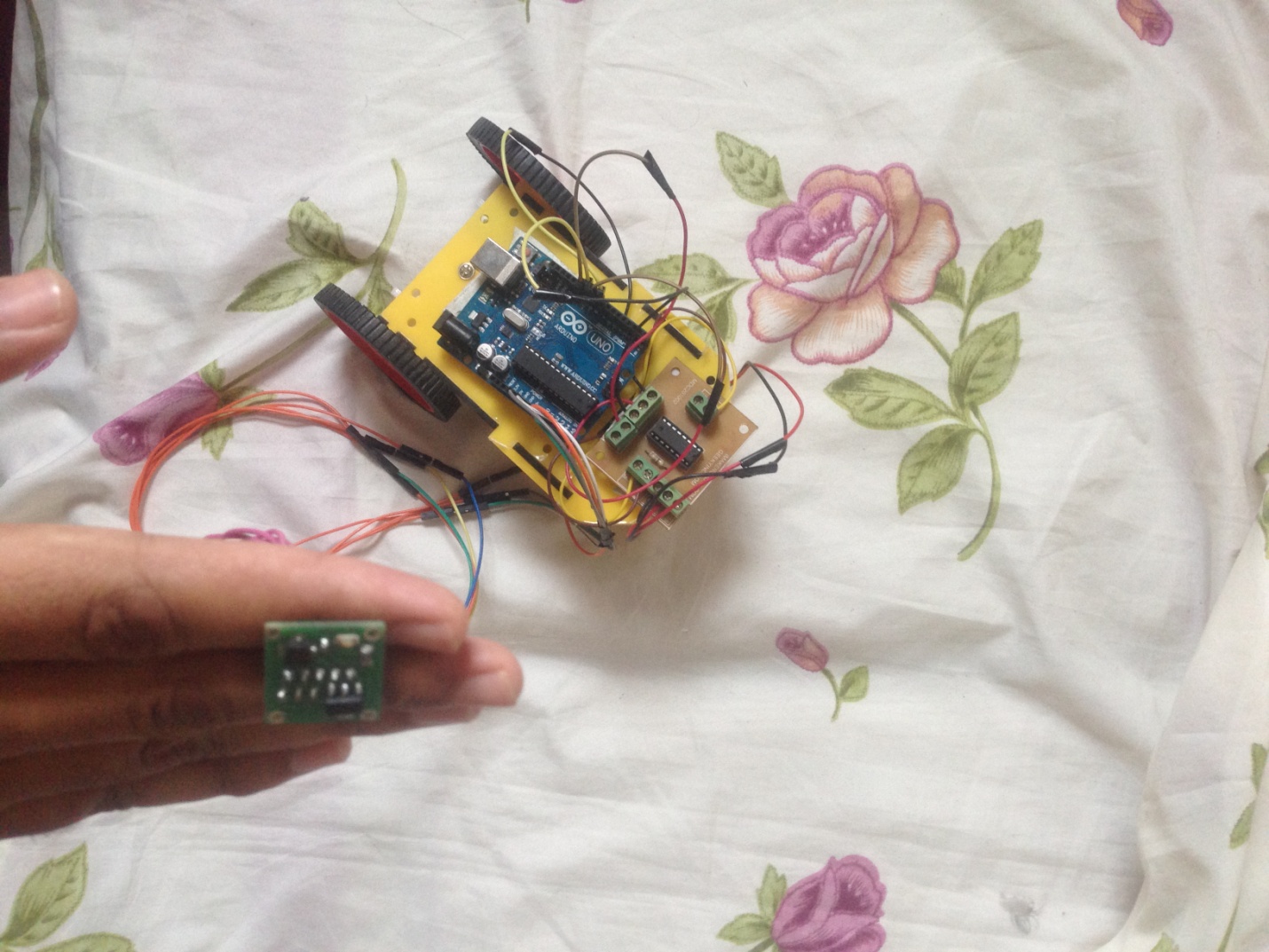
****

**Different gestures to control the robot, clockwise forward, backward , left and right**

****

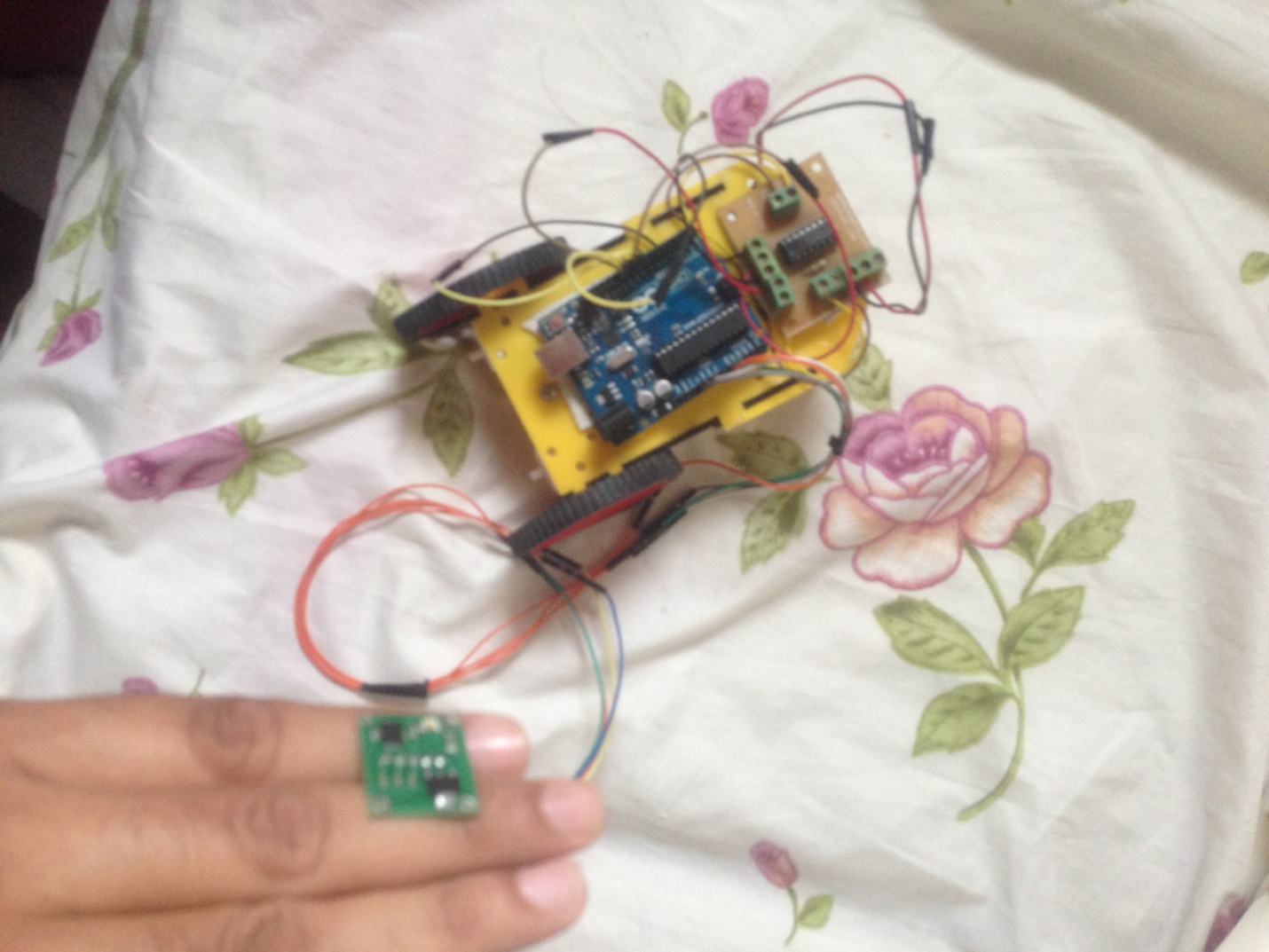
Chapter-5

**Final robot hardware**

****

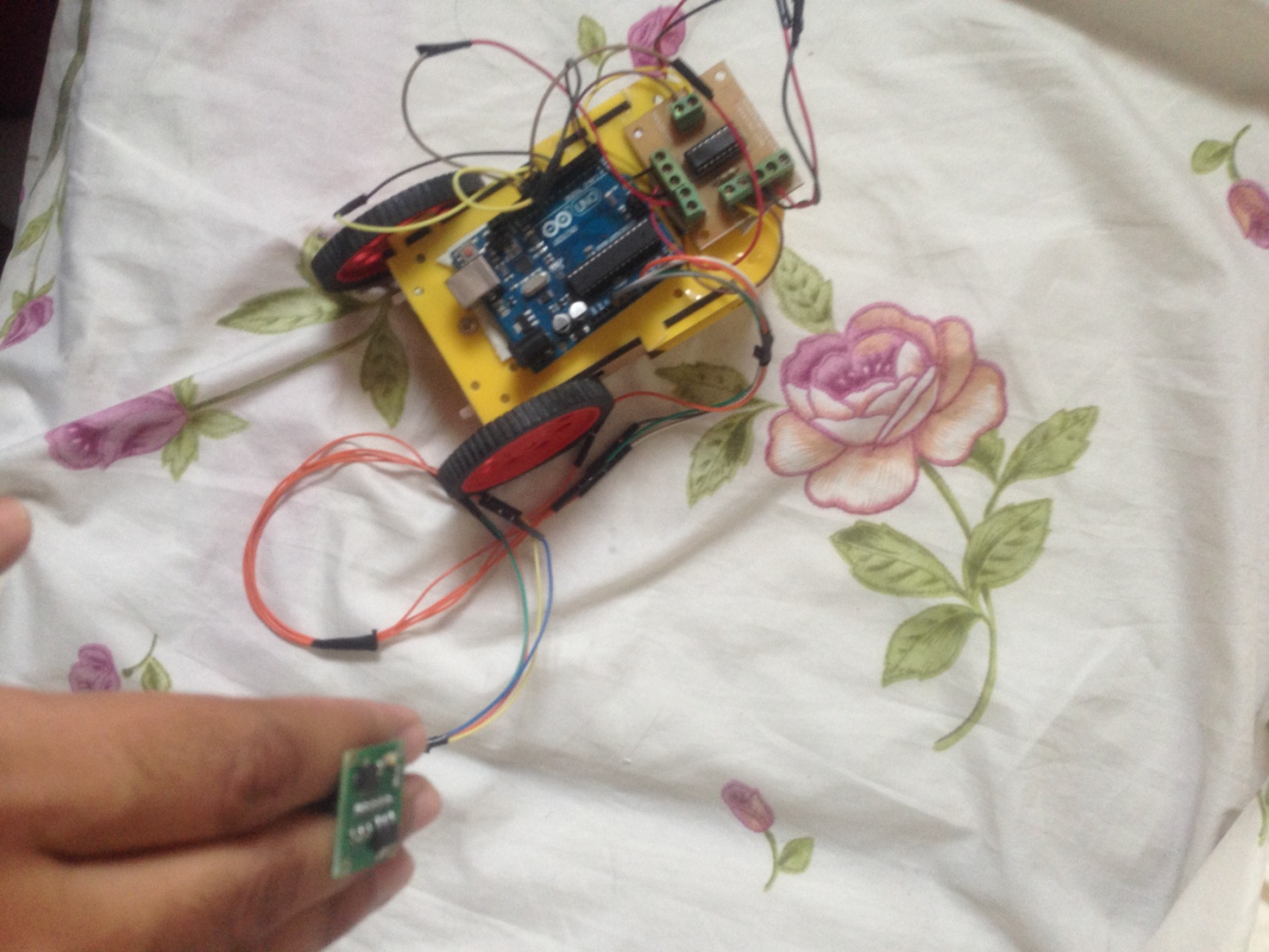
Chapter-5

**Robot moving left**

****

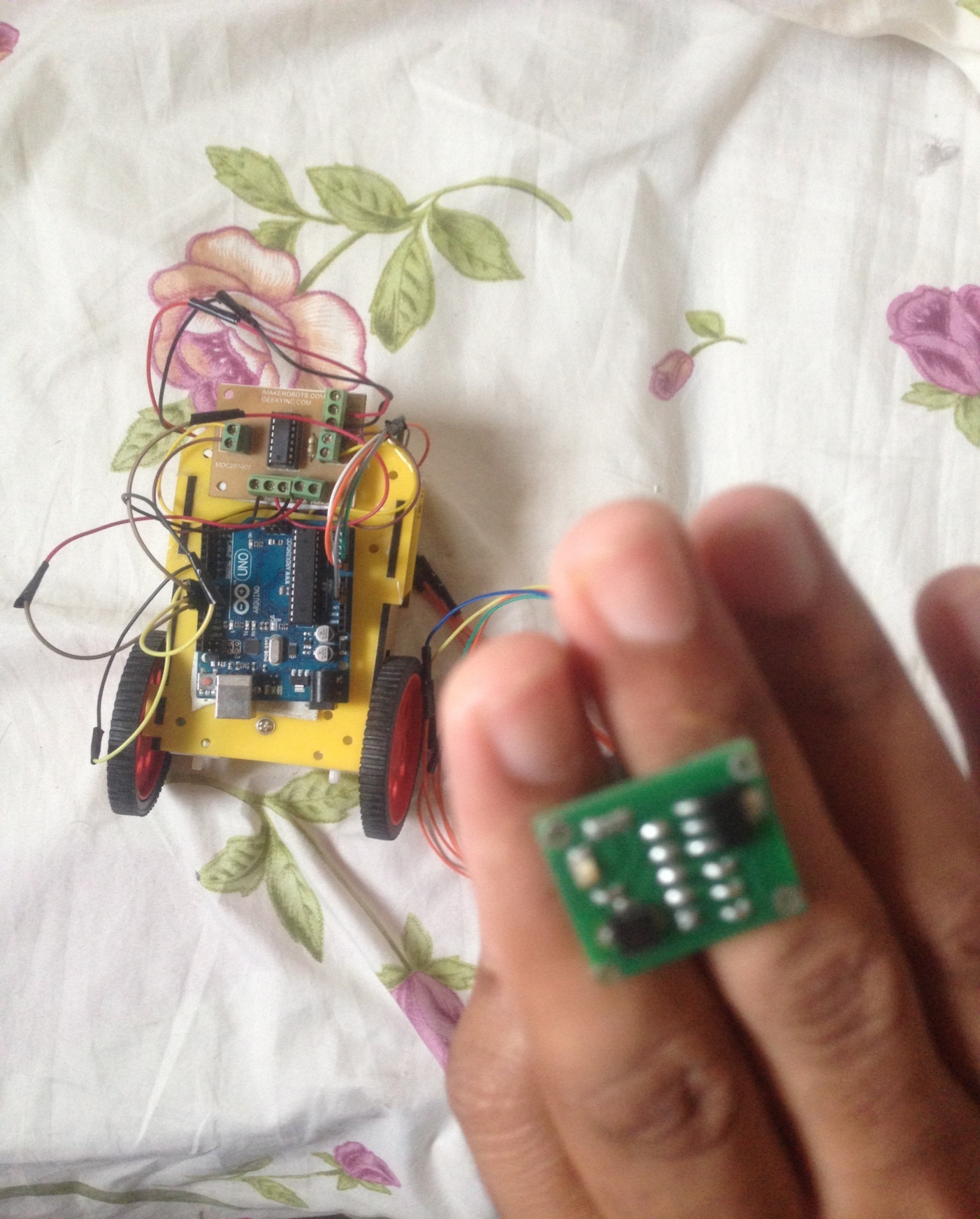
Chapter-5

**Robot moving right**

****

Chapter-5

**Robot moving front**

****

Chapter-5

**Robot moving backwards**

**6. CONCLUSION**

Chapter-6

The prime aim of the design was that the robot and platform starts the movement as soon as the operator makes a gesture or posture or any motion. The Robot was synchronized with the gestures (hand postures) of the operator and the platform part is synchronized with the gestures (leg postures) of the operator. The goal of this paper was to develop methodologies that help users to control and program a robot, with a high-level of abstraction from the robot specific language i.e. to simplify the robot programming.

Many variants of these robots/robotic are available or designed as per the requirement. Few variants are Keypad Controlled, Voice Control, Gesture Control, etc. However, most of the industrial robots are still programmed using the typical teaching process which is still a tedious and time-consuming task that requires technical expertise. Therefore, the need for new and easier ways for programming the robots was realized.

The Robot was successfully able to regulate the output of the motors based on the movement of the Accelerometer and was able to move to the desired position.

**7. FUTURE ASPECTS**

Chapter-7

Robotic arms can be developed based on this application and they are the vital part of almost all the industries. In industries, a robotic arm performs various different tasks such as welding, trimming, picking and placing etc. Moreover the biggest advantage of these arms is that it can work in hazardous areas and also in the areas which cannot be accessed by human. For example in NASA’s mission to Mars, the Spirit and Opportunity drone. It is also used to implement highly precise medical treatments etc.

This Gesture Recognition methodology has become increasingly popular in a very short span of time. The low moderate cost and relative small size of the accelerometers are the two factors that make it an effective tool to detect and recognize human body gestures.

This recognition technique made it possible to implement an accelerometer based system to communicate with an industrial robotic arm wirelessly. In this particular project the robotic arm is powered with ARM7 based LPC1768 core. MEMS is a three dimensional accelerometer sensor which captures gestures of human-arm and produces three different analog output voltages in three dimensional axes. And two flex sensors are used to control the gripper movement.

**8. BIBLOGRAPHY AND REFERENCES:**

Chapter-8

* The 8051 Micro controller and Embedded Systems

-**Muhammad Ali Mazidi**

**-Janice Gillispie Mazidi**

* The 8051 Micro controller Architecture, Programming & Applications

-**Kenneth J.Ayala**

* Fundamentals Of Micro processors and Micro computers

**-B.Ram**

* Micro processor Architecture, Programming & Applications

**-Ramesh S. Gaonkar**

* Electronic Component

-**D.V. Prasad**

* Wireless Communications

-**Theodore S. Rappaport**

**References on the Web:**

Chapter-8

[www.national.com](http://www.national.com)

[www.atmel.com](http://www.atmel.com)

[www.microsoftsearch.com](http://www.microsoftsearch.com)

[www.geocities.com](http://www.geocities.com)

[www.ieeeexplore.com](http://www.ieeeexplore.com)