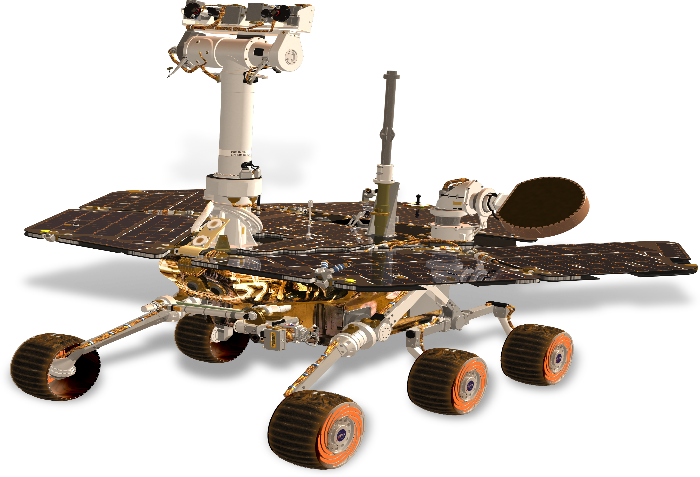


Index

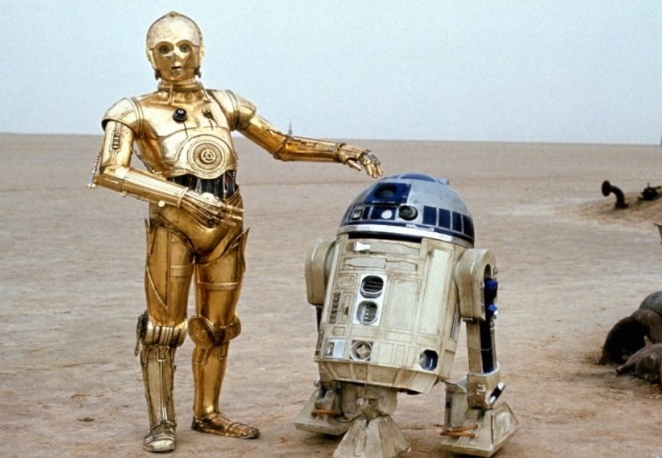
|  |  |
| --- | --- |
| **Chapter** | **Page No.** |
| Robotics | 3 |
| Sensors | 8 |
| Introduction to Arduino | 12 |
| Interfacing with Arduino | 17 |
| Robot Algorithms | 21 |



1

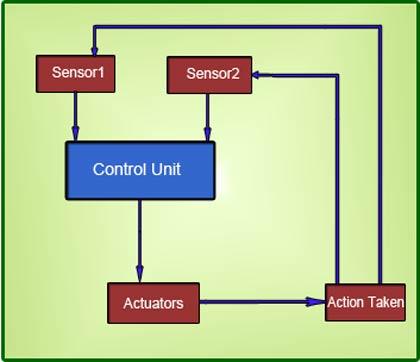
Robotics

**Introduction**

What does one visualize while thinking of a **Robot**? Does a robot roll as R2D2 (from Star Wars) or walk upright as C3P0? Does it have expressionistic face or a mechanical one? Does it function as vacuum or picks objects for you? As there are many uses of **Robotics**, it is reasonable to think that there are various designs too. To know how these complex yet intelligent machines move, one needs to look at their engineering, physics closely.

**Figure 1.1 R2D2(at left) and C3P0 (at right) From Star Wars**

Other than the sci-fi movies, the **Robots** can be seen around us assembling the cars, in bottling factory etc. The robots have been in the industry for last two decades because of their continuous working ability in an atmosphere, where humans are not even able to stand for minutes, without any supporting equipment, like space.

Basic structure of robots is very much like humans. How do humans sense? For example, a human sees something and sends neural signals to the brain via neurons and reacts accordingly. The development of all these senses artificially is achieved through 'Sensors'. Sensors are the transducers which receive the physical changes of the environment and convert them into electrical or electronic signals. These analog signals are converted into digital by using analog-to-digital convertors. Control system functions as a brain in robotic systems.

**Figure 1.2 Basic Structure of a Robot**

As strange as it might seem, there really is no standard definition for a robot. However, there are some essential characteristics that a robot must have and this might help you to decide what is and what is not a robot. It will also help you to decide what features you will need to build into a machine before it can count as a robot.

A robot has these essential characteristics:

* **Sensing** First of all your robot would have to be able to sense its surroundings. It would do this in ways that are similar to the way that you sense your surroundings. Giving your robot sensors: light sensors (eyes), touch and pressure sensors (hands), chemical sensors (nose), hearing and sonar sensors (ears), and taste sensors (tongue) will give your robot awareness of its environment.
* **Movement** A robot needs to be able to move around its environment. Whether rolling on wheels, walking on legs or propelling by thrusters a robot needs to be able to move. To count as a robot either the whole robot moves, like the Sojourner or just parts of the robot moves, like the Canada Arm.
* **Energy** A robot needs to be able to power itself. A robot might be solar powered, electrically powered, battery powered. The way your robot gets its energy will depend on what your robot needs to do.
* **Intelligence** A robot needs some kind of "smarts." This is where programming enters the pictures. A programmer is the person who gives the robot its 'smarts.' The robot will have to have some way to receive the program so that it knows what it is to do.

**So, what is the Definition of robot?**

Well it is a system that contains sensors, control systems, manipulators, power supplies and software all working together to perform a task. Designing, building, programming and testing robots is a combination of physics, mechanical engineering, electrical engineering, structural engineering, mathematics and computing. In some cases, biology, medicine, chemistry might also be involved. A study of robotics means that students are actively engaged with all these disciplines in a deeply problem-posing problem-solving environment.

**Laws of Robotics**

Isaac Asimov popularized the term robotics. Asimov was a visionary who envisioned in the 1930’s the positronic brain for controlling robots. He stated the following three Laws of Robotics:

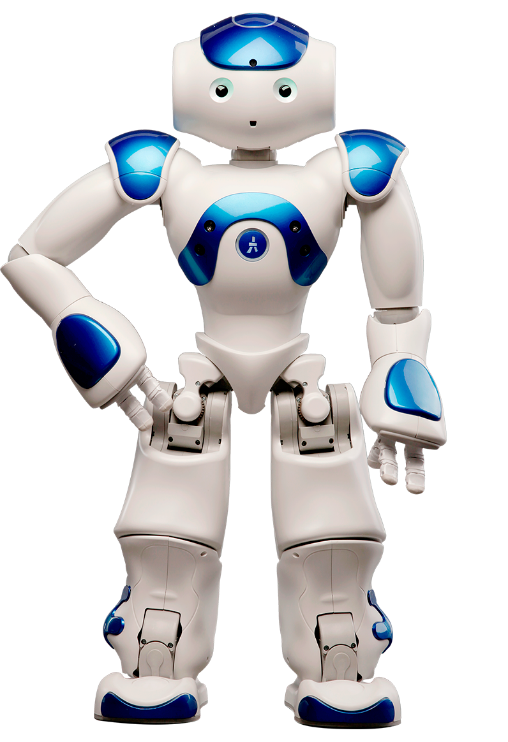
1. A robot may not harm a human through action or inaction, allow a human to come to harm
2. A robot must obey the orders given by human beings, except when such orders conflict with the FIRST LAW.
3. A robot must protect its own existence if it does not conflict with the first or second laws.

**Design Goals of Robots**

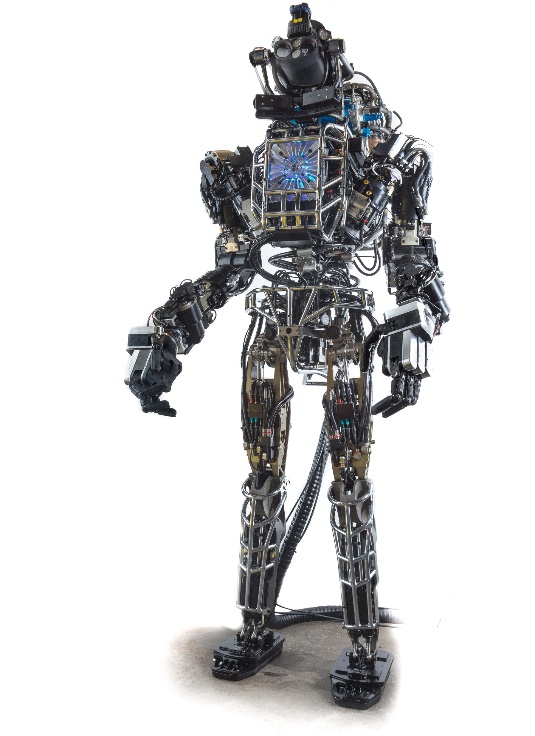
Basically, there are four design goals of robotics, those are as follows:

* **Sensor Rich-** It should have multiple sensors for detecting environmental changes around it.
* **Controllability-** It should be controllable.
* **Reliability-** It should be reliable to use in various situations.
* **Efficiency-** It should be effective under the various parameters also.

**Some Examples of Robots**

****

**Figure 1.3 NAO Robot**

* **NAO** is an autonomous humanoid robot developed by Aldebaran Robotics in Paris.
* The Nao was used in RoboCup 2008 and 2009, and the NaoV3R was chosen as the platform for the SPL at RoboCup 2010.
* The NAO Academics Edition was developed for universities and laboratories for research and education purposes and was released publicly in 2011
* **Atlas** is a bipedal humanoid robot primarily developed by the American robotics company Boston Dynamics.
* Atlas is intended to aid emergency services in search and rescue operations, performing tasks such as shutting off valves, opening doors and operating powered equipment in environments where humans could not survive.

**Figure 4 Atlas Robot**

2

Sensors

**What is Sensor?**

* A sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.
* Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim r brighten by touching the base.
* Applications include cars, machines, aerospace, medicine, manufacturing and robotics.

**Types of Sensors**

There are many types of sensors available in the market but here we will study two types of sensors:

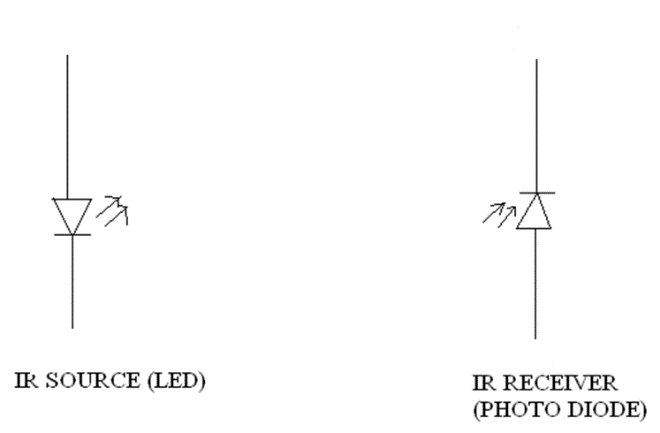
* IR Sensor
* Ultrasonic Sensor

**IR Sensor**

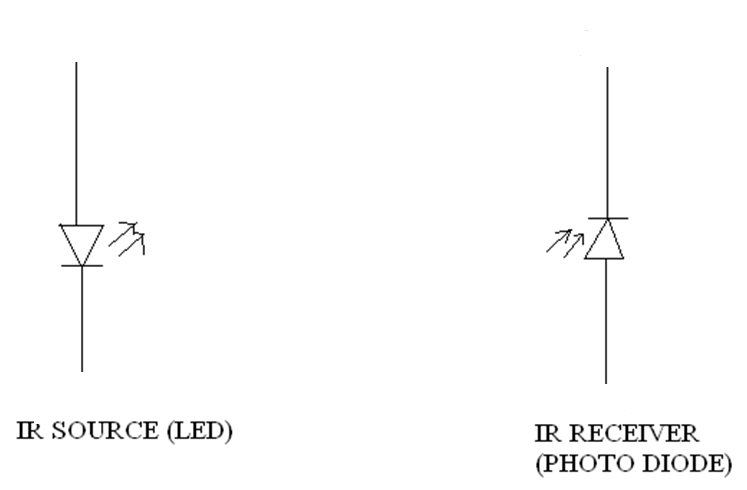
IR sensor basically consist an IR LED and a Photodiode, this pair is generally called IR pair or Photo coupler. IR sensor work on the principal in which IR LED emits IR radiation and Photodiode sense that IR radiation. Photodiode resistance changes according to the amount of IR radiation falling on it, hence the voltage drop across it also changes and by using the voltage comparator (like LM358) we can sense the voltage change and generate the output accordingly.

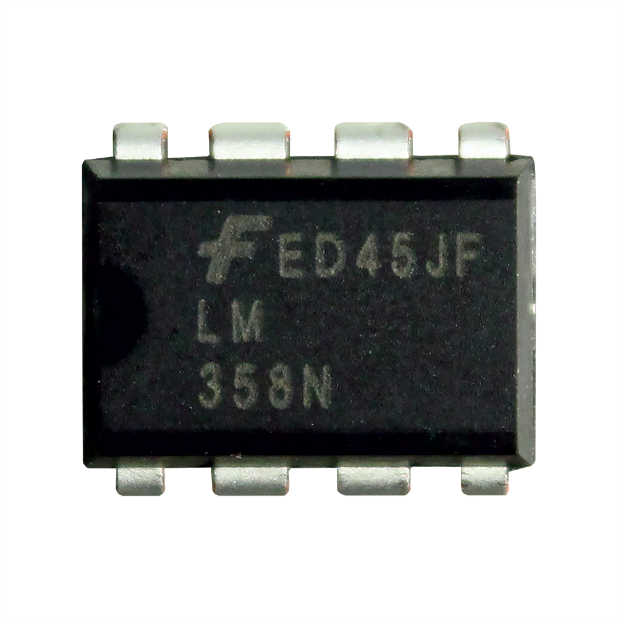
It is composed of following devices:

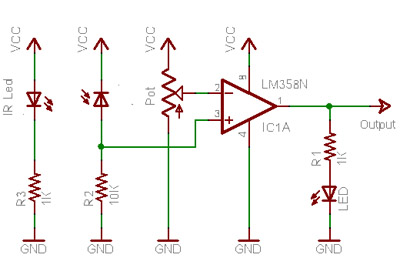
* IR LED



* IR Receiver(Photodiode)



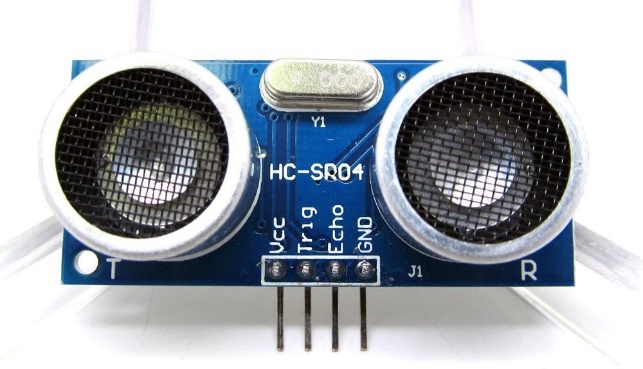
* Voltage Comparator
* A comparator is a device which compares two voltages or currents and switches its output to indicate which is larger.
* Comparator is On-Amp.

**Circuit Diagram**

**Figure 2.1 IR Sensor Circuit Diagram**

* The LM35 is an integrated circuit sensor that can be used to measure the temperature with an electrical output proportional to the temperature (in **o**C).
* The scale factor is 10mV/ **o**C.

**Ultrasonic Sensor**



**Figure 2.2 Ultrasonic Sensor**

Ultrasonic transmitter emitted an ultrasonic wave in one direction and started timing when it launched. Ultrasonic spread in the air and would return immediately when it encountered obstacles on the way. At last, the ultrasonic receiver would stop timing when it received the reflected wave. As Ultrasonic spread velocity is 340m / s in the air, based on the timer record **t**, we can calculate the distance (s) between the obstacle and transmitter, namely: s=340t/ 2, which is so- called time difference distance measurement principle.



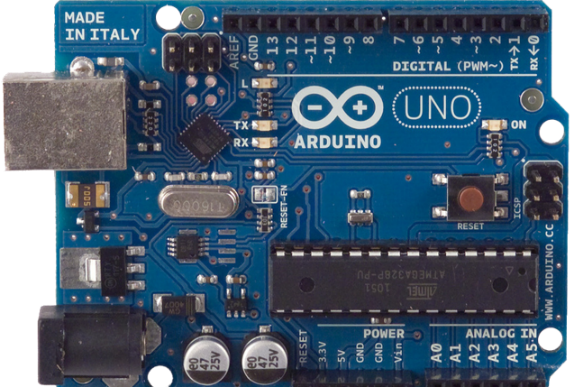
|  |  |
| --- | --- |
| Electrical Parameters | HC-SR04 Ultrasonic Module |
| Operating Voltage | DC-5V |
| Operating Current | 15mA |
| Operating Frequency | 40KHZ |
| Farthest Range | 4m |
| Nearest Range | 2cm |
| Measuring Angle | 15 Degree |
| Dimensions | 45\*20\*15mm |

Introduction To Arduino

3

A

rduino is an open source development platform designed for hobbyists and anyone interested in electronics. You Don’t have to be a programming Rockstar to work on Arduino.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload code to the physical board.

**Figure 3.1 An Arduino Board**

Arduino is quite popular with people just starting out with electronics, and for good reason. Unlike others, Arduino does not need a separate programmer hardware to load new code on to the board as well as Arduino uses a simplified version of C++, making it easier to learn to program.

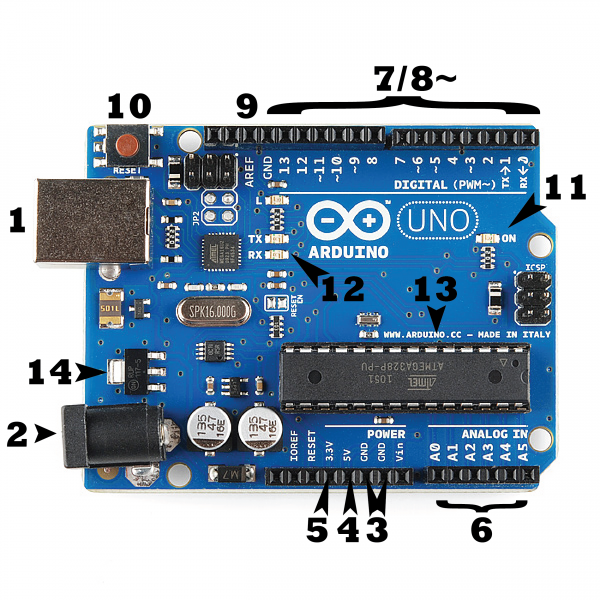
**What does It do?**

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects.

For everything from robots and a heating pad hand warming blanket to honest fortune-telling machines, and even a Dungeons and Dragons dice-throwing gauntlet, the Arduino can be used as the brains behind almost any electronics project.

**What’s on the board?**

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduinos have most of these components in common:

**Power (USB / Barrel Jack)**

**Figure 3.2 Arduino Uno board**

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labelled **(1)** and the barrel jack is labelled **(2)**.

The USB connection is also how you will load code onto your Arduino board.

**NOTE:** Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

**Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)**

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labelled on the board and used for different functions.

* **GND (3)**: Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
* **5V (4) & 3.3V (5)**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
* **Analog (6)**: The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
* **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

**PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM).

* but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
* **AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

**Reset Button**

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn’t repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn’t usually fix any problems.

**Power LED Indicator**

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’ **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong. Time to re-check your circuit!

**TX RX LEDs**

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

**Main IC**

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC’s from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC’s, reading the datasheets is often a good idea.

**Voltage Regulator**

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it’s for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don’t hook up your Arduino to anything greater than 20 volts.

Interfacing with Arduino

4

**Setting up Environment**

1. Download Arduino IDE from <https://www.arduino.cc>



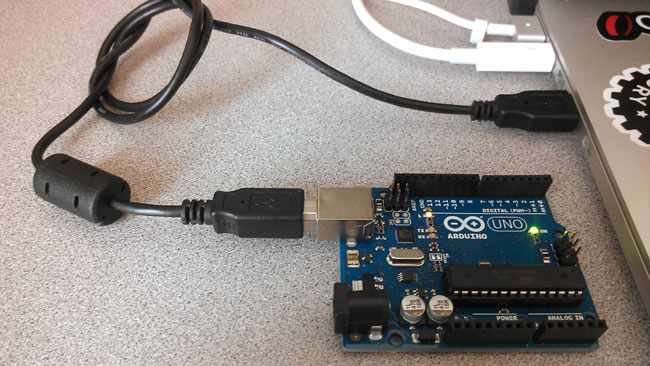
Figure 4.1 Arduino IDE

Select the IDE for your type of Operating System, as I am using windows on my PC so I selected windows installer.

1. Install IDE in your PC

For windows, Simply install IDE using the windows installer or .exe file provided, for mac and Linux just follow the readme file given in the package.

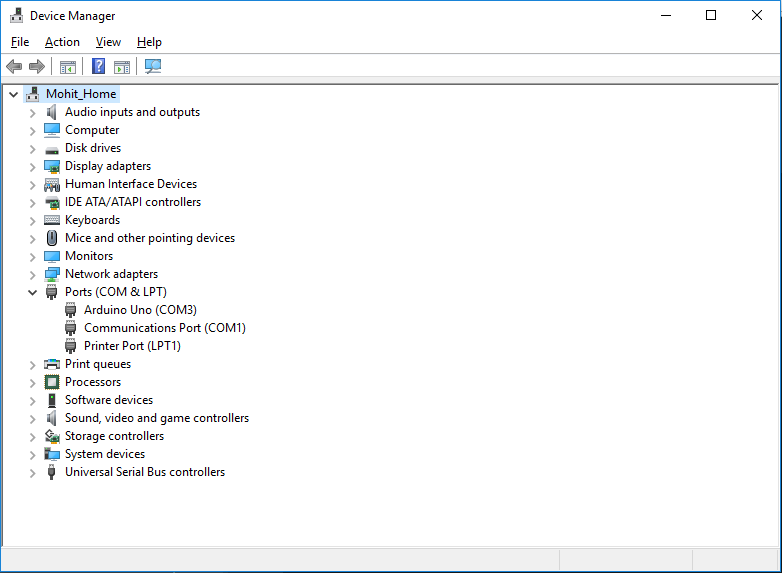
**Connecting Arduino to PC**



**Figure 4.2 Arduino Connected to pc**

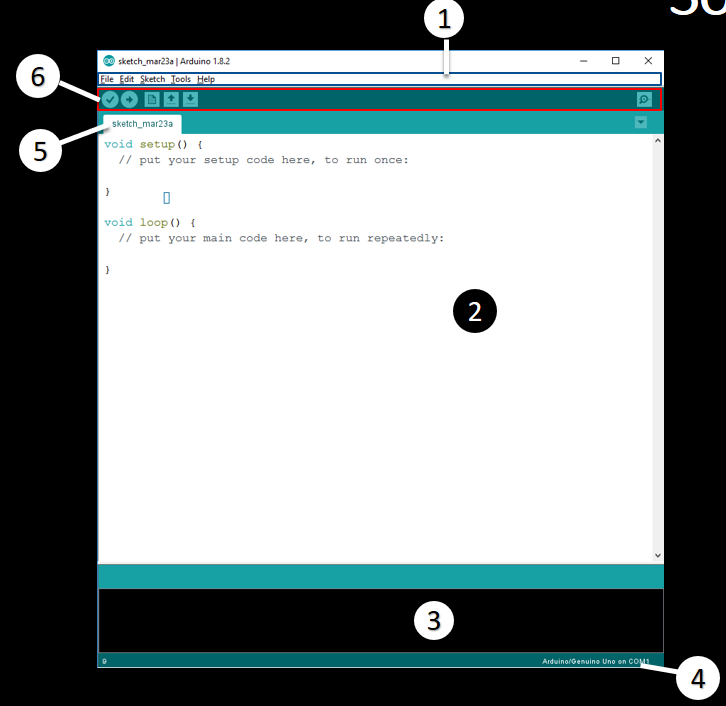
Now, connect your Arduino to your computer as shown in the above image and wait for the drivers to install.

Check the port number on which Arduino is connected using device manager in your windows pc.

As can be seen in figure 4.2, it should appear under Ports list, if you own a Chinese clone of Arduino, then look for the drivers over internet for its usb to serial ic.

**Figure 4.3 Arduino in Device Manager**

**Arduino IDE interface**



**Figure 4.4 Arduino IDE Interface**

1. Menu Bar
2. Text Editor
3. Debug OUTPUT Window
4. Connected Device Port
5. Project Title Tab
6. Button Bar

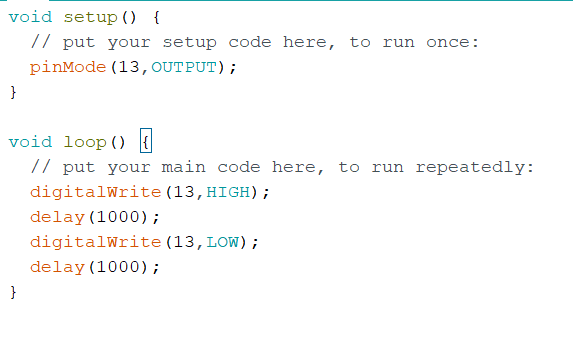
**Writing Your First Arduino Program**

Whenever we start to learn any programming language, we write a “Hello world!!!” program as our very first step to that programming language.

Similarly, in the world of microcontroller or hardware programming, we “Blink and LED” as our very first step to the world of hardware programming.

As we know that Arduino have an LED connected to D13 pin so, we will blink this LED as our first Arduino Program.

Open Arduino IDE and type the following code in it then connect your Arduino and press ctrl+U to upload code to Arduino.



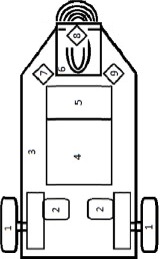
**Figure 4.5 LED Blink Code**

As an output, you will see led connected to pin D13 blinking once per second, as shown in figure 4.2.

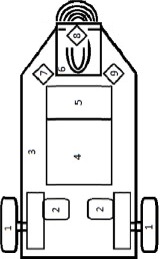
5

Algorithms

**Obstacle Avoiding Robot**

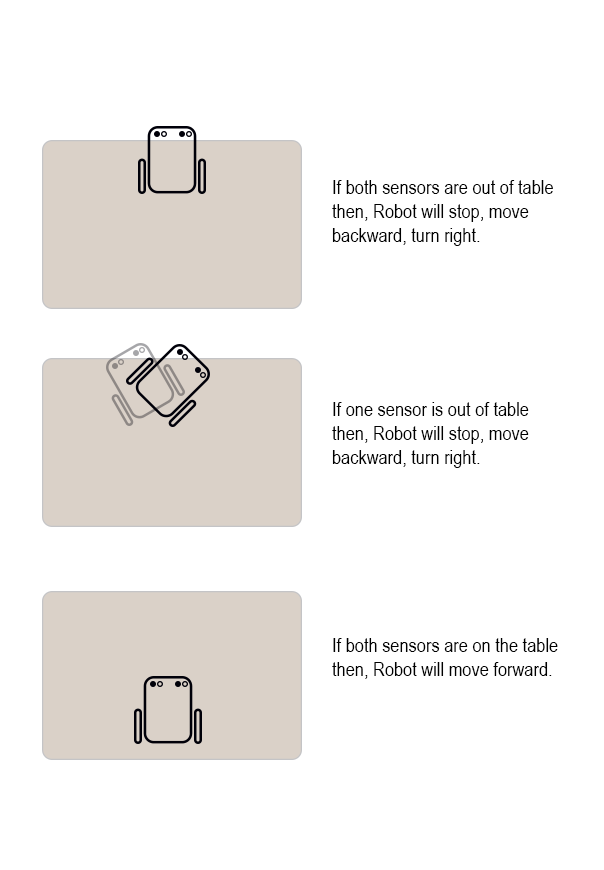


If the ultrasonic sensor of robot did not detect any obstaclein its path, then it allow the robot to continue its path.



If ultrasonic sensor detects any obstacle within the range specified by the programmer, then it will turn the robot right to avoid the obstacle.

**Edge Avoider Algorithm**

****

****