ABSTRACT

Autonomous robots are used everywhere nowadays and robotics is also gaining popularity all around the world. This motivated us to make an autonomous robot that can avoid obstacle from its path. There are many kind of autonomous robot present nowadays and the technique for making it autonomous and the ability of obstacle avoidance comes from various methods. Already existing methods different types of autonomous robots:-

1. Programmable
2. Non-Programmable
3. Adaptive
4. Intelligent

Our robotic car will be a programmable type which will follow a particular path and if there are any changes required in the path then program should be changed. The robotic car will also be able to detect any obstacle in its path and move accordingly such that the obstacle is avoided and the car reaches its final destination. The robot is made mobile with the help of dc motors. All these motors are controlled by Arduino board. It controls these motors according to the input program.

.

Table Of Contents

List Of Figures

CHAPTER 1

INTRODUCTION

An obstacle avoiding robot is an intelligent device, which can automatically sense and overcome obstacles on its path. Obstacle Avoidance is a robotic discipline with the objective of moving vehicles on the basis of the sensorial information. The use of these methods front to classic methods (path planning) is a natural alternative when the scenario is dynamic with an unpredictable behaviour. In these cases, the surroundings do not remain invariable, and thus the sensory information is used to detect the changes consequently adapting moving. It will automatically scan the surrounding for further path.

An obstacle avoiding robot uses a proximity sensor module, besides other parts. In this case, this robot uses a proximity sensor developed by ourselves. The robot is controlled by a program that is embedded into a micro controller. The logics produced by the micro controller are further processed by an interface module, in this case, also developed by ourselves. The interface module translates micro controller's logics into voltage and current that can practically drive the two motors. This article provides a report on the project activity, consisting of summary of the design, summary of the development process and report on the running test of the robot. Following the test and program fine-tuning, it has been proven that the robot model operated well just as programmed.

1.1 AIMS AND OBJECTIVE

A simple project on Obstacle Avoiding Car is designed here. Robotics is an interesting and fast growing field. Being a branch of engineering, the applications of robotics are increasing with the advancement of technology. The concept of Mobile Car is fast evolving and the number of mobile robots and their complexities are increasing with different applications. There are many types of mobile robot navigation techniques like path planning, self – localization and map interpreting. An Obstacle Avoiding Car is a type of autonomous mobile robot that avoids collision with unexpected obstacles.

In this project, an Obstacle Avoiding Robot is designed. It is an Arduino based robot that uses Ultrasonic range finder sensors to avoid collisions.

Obstacle avoidance is a primary requirement of any autonomous mobile robot.Obstacle avoidance Robot is design to allow robot to navigate in unknown environment by avoiding collisions.Obstacle avoiding robot senses obstacles in the path,avoid it and resumes its running.

1.2 BACK GROUND OF THE PROJECT

The software application and the hardware implementation help the micro controller read the output of the sensors.The project proposes a autonomous robotic vehicle, In which no remote is used for controlling the robotic actions. It intelligently detects obstacles present on its path through the sensors, avoid it and take decision on the basis of internal code that we set for it. The detail information is given in the following subtopics which will help you to understand the whole system and its design.

The Arduino Uno is a micro controller board based on the ATmega328 (data sheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

CHAPTER 2

PROJECT DESCRIPTION

2.1 FLOW CHART

The flow chart of the design is as shown in Fig 2.1.The brief description of each unit is explained below.

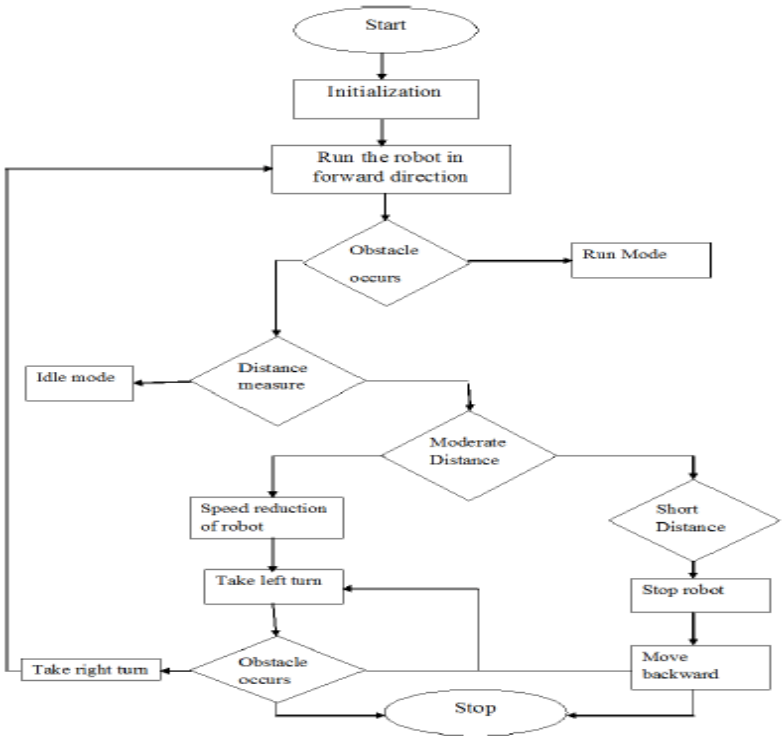

Fig 2.1: Flowchart for Obstacle Avoiding Car

2.2 CIRCUIT DIAGRAM

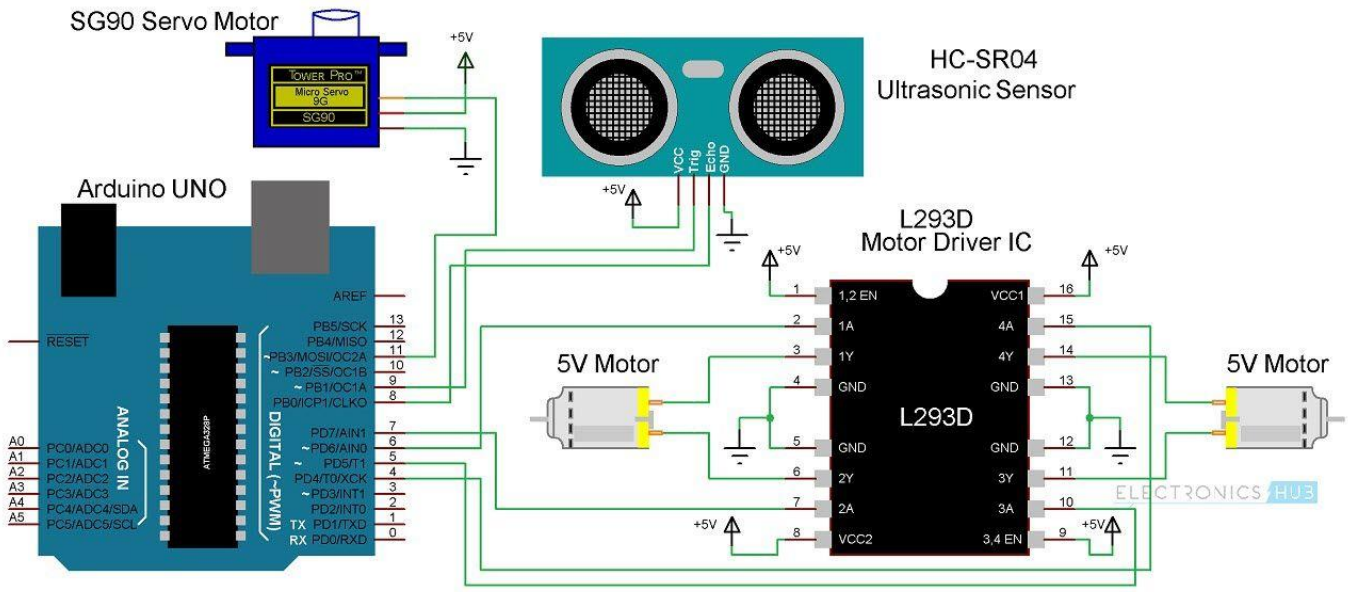

Fig 2.2 Circuit Diagram.

2.3 WORKING PROCEDURE

The implementation of obstacle avoidance strategy for robot involves the writing and compilation of program using Arduino software. Arduino is a popular programmable board used to create projects.It consists of a simple hardware platform on which micro controller is placed as well as a free code editor which has a “one click compile or upload” feature. Hence it is designed for the peoplesin such a way that they can use it without necessarily being an expert programmer. Arduino offers an open-source electronic prototyping platform that is easy to use and flexible for peoples who are beginners in robotics field with both the software and hardware perspective.

Sensors are connected with the Arduino board using breadboard. Micro controller is able to sense the environment through receiving input from sensors. It is also able to control its surrounding through controlling motors and other actuators.The Arduino programming language that is based on the processing are used to program the micro controller found on the board. Due to its open source environment, we can able to easily write and upload codes to the I/O board. Arduino environment is written in Java hence it can be run on Linux, Mac OSX and Windows

platforms. The output of the comparator is given to the micro controller, which then moves actuators in left or right direction by giving power through DC motor.

.

CHAPTER 3
MICROCONTROLLER

3.1. A brief history of ARDUINO UNO:

Arduino/Genuino Uno is a micro controller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

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.

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.

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).

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 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. □

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details. □

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. □ SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library. □

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. The Uno has 6 analog inputs, 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 analogReference() function.

Additionally, some pins have specialized functionality: □

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board: □

AREF. Reference voltage for the analog inputs. Used with analogReference(). □

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board. See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other micro controllers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by: □ On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. □ On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and

Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following halfsecond or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

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.

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.

Fig 3.1: Arduino Uno Circuit Board

KEY PARAMETERS

Parameter	Value
Micro Controller	ATmega328
Operating Voltage	5V
Flash memory	32 Kb

SRAM	2 Kb
EEPROM	1 Kb
Pin count	14 Digital I/O pins, 6 Analog Input pins.
Input Voltage(recommended)	7-12 V
Input Voltage(limits)	6-20 V
Clock Speed	16 MHz
External interrupts	-
USB Interface	No

3.2 PIN DIAGRAM:

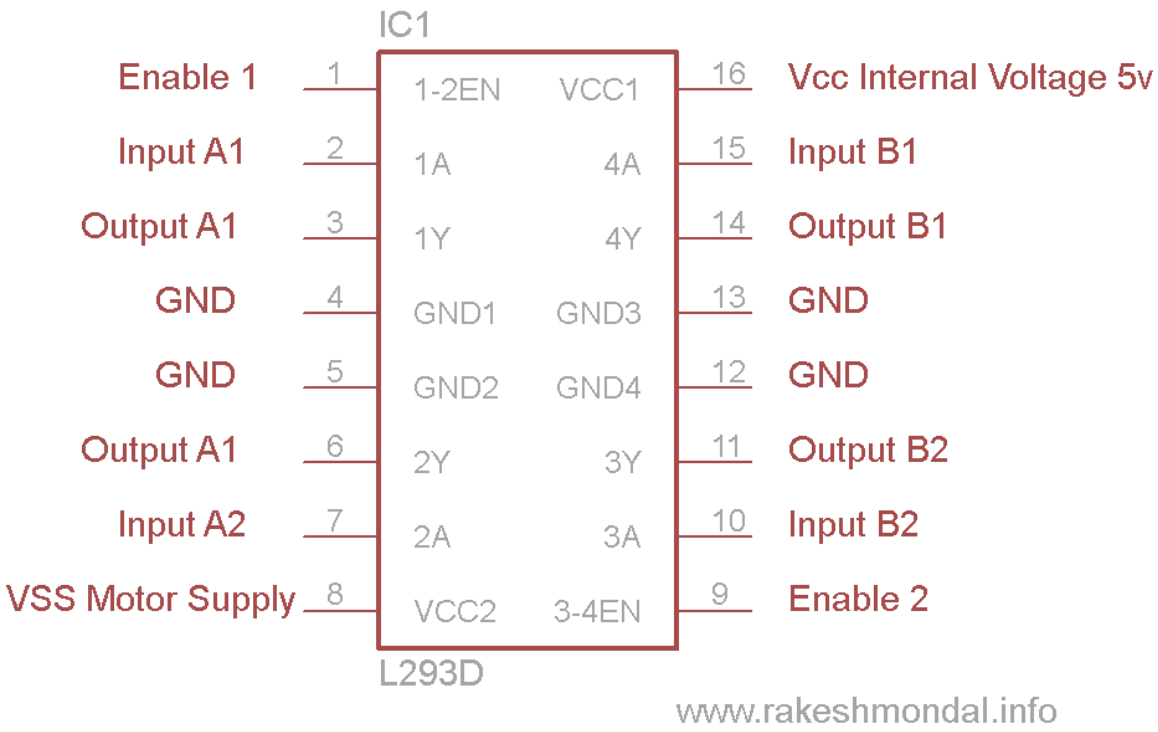

Fig 3.2 Pin Diagram of L293D

3.3 PIN DESCRIPTION

The table below gives a description for each of the pins, along with their function.

Pin Number	Description	Function
1	Enable 1,2	This pin enables the input pin Input 1(2) and Input 2(7)
2	Input 1	Directly controls the Output 1 pin. Controlled by digital circuits
3	Output 1	Connected to one end of motor
4	Ground	Ground pins are connected to ground of circuit (0V)
5	Ground	Ground pins are connected to ground of circuit (0V)

6	Output 2	Connected to one end of motor
7	Input 2	Directly controls the Output 2 pin. Controlled by digital circuits
8	Vcc2(Vs)	Connected to Voltage pin for running motors.
9	Enable 3,4	This pin enables the input pin Input 3(10) and Input 4(15)
10	Input 3	Directly controls the Output 3 pin. Controlled by digital circuits
11	Output 3	Connected to one end of Motor 2
12	Ground	Ground pins are connected to ground of circuit (0V)
13	Ground	Ground pins are connected to ground of circuit (0V)
14	Output 4	Connected to one end of motor
15	Input 4	Directly controls the Output 4 pin. Controlled by digital circuits
16	Vcc2(Vss)	Connected to +5V to enable IC function

The L293D is a famous 16-Pin Motor Driver IC. As the name suggests it is mainly used to drive motors. A single **L293D IC** is capable of running two DC motors at the same time; also the direction of these two motors can be controlled independently. So if you have motors which has operating voltage less than 36V and operating current less than 600mA, which are to be controlled by digital circuits like Op-Amp, 555 timers, digital gates or even Micron rollers like Arduino, PIC, ARM etc.. this IC will be the right choice for you.
Using this L293D motor driver IC is very simple. The IC works on the principle of **Half H-Bridge**, let us not go too deep into what H-Bridge means, but for now just know that H bridge is a set up which is used to run motors both in clock wise and anti clockwise direction. As said earlier this IC is capable of running two motors at the any direction at the same time.

All the Ground pins should be grounded. There are two power pins for this IC, one is the Vss(Vcc1) which provides the voltage for the IC to work, this must be connected to +5V. The other is Vs(Vcc2) which provides voltage for the motors to run, based on the specification of your motor you can connect this pin to anywhere between 4.5V to 36V, here I have connected to +12V.

The Enable pins (Enable 1,2 and Enable 3,4) are used to Enable Input pins for Motors respectively. Since in most cases we will be using both the motors both the pins are held high by default by connecting to +5V supply. The input pins Input 1,2 are used to control the motor 1 and Input pins 3,4 are used to control the Motor 2. The input pins are connected to the any Digital circuit or micro controller to control the speed and direction of the motor. You can toggle the input pins based on the following table to control your motor.

Whenever a robotics hobbyist talk about making a robot, the first thing comes to his mind is making the robot move on the ground. And there are always two options in front of the designer whether to use a DC motor or a stepper motor. When it comes to speed, weight, size, cost... DC motors are always preferred over stepper motors. There are many things which you can do with your DC motor when interfaced with a micro controller. For example you can control the speed of motor; you can control the direction of rotation. In this part of tutorial we will learn to interface and control of a DC motor with a micro controller. Usually H-bridge is preferred way of interfacing a DC motor. These days many IC manufacturers have H-bridge motor driver available in the market like L293D is most used H- Bridge driver IC. H-bridge can also be made with the help of transistors and MOSFETs etc. rather of being cheap, they only increase the size of the design board, which is sometimes not required so using a small 16 pin IC is preferred for this purpose.

Differential drive By using two motors we can move our robot in any direction. This steering mechanism of robot is called as differential drive. Let's check how it works Left Motor Right Motor Robot Movement Straight Straight Straight Stop Straight Left Reverse Straight Sharp left Straight Stop Right Straight Reverse Sharp.

CHAPTER 4

ULTRASONIC SENSOR

The sonic waves emitted by the transducer are reflected by an object and received back in the transducer. After having emitted the sound waves, the ultrasonic sensor will switch to receive mode. The time elapsed between emitting and receiving is proportional to the distance of the object from the sensor.

Ultrasonic sensor are devices that use electrical–mechanical energy transformation to measure distance from the sensor to the target object. Ultrasonic waves are longitudinal mechanical waves which travel as a sequence of compressions and rarefactions along the direction of wave propagation through the medium.

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

In order to generate the ultrasound you need to set the Trig on a High State for 10s. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

Fig 4.1 Ultrasonic Sensor

First you have to define the Trig and Echo pins. In this case they are the pins number 9 and 10 on the Arduino Board and they are named trigPin and echoPin. Then you need a Long variable, named “duration” for the travel time that you will get from the sensor and an integer variable for the distance.

In the setup you have to define the trigPin as an output and the echoPin as an Input and also start the serial communication for showing the results on the serial monitor.

In the loop first you have to make sure that the trigPin is clear so you have to set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 µs. Using the pulseIn() function you have to read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the puls() function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously. At the end we will print the value of the distance on the Serial Monitor.

Ultrasonic sensors emit short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo.

As the distance to an object is determined by measuring the time of flight and not by the intensity of the sound, ultrasonic sensors are excellent at suppressing background interference.

Virtually all materials which reflect sound can be detected, regardless of their colour. Even transparent materials or thin foils represent no problem for an ultrasonic sensor.microsonic ultrasonic sensors are suitable for target distances from 20 mm to 10 m and as they measure the time of flight they can ascertain a measurement with pinpoint accuracy. Some of our sensors can even resolve the signal to an accuracy of 0.025 mm.

Ultrasonic sensors can see through dust-laden air and ink mists. Even thin deposits on the sensor membrane do not impair its function.

Sensors with a blind zone of only 20 mm and an extremely thin beam spread are making entirely new applications possible today: Fill level measurement in wells of microtiter plates and test tubes, as well as the detection of small bottles in the packaging industry, can be implemented with ease. Even thin wires are reliably detected.

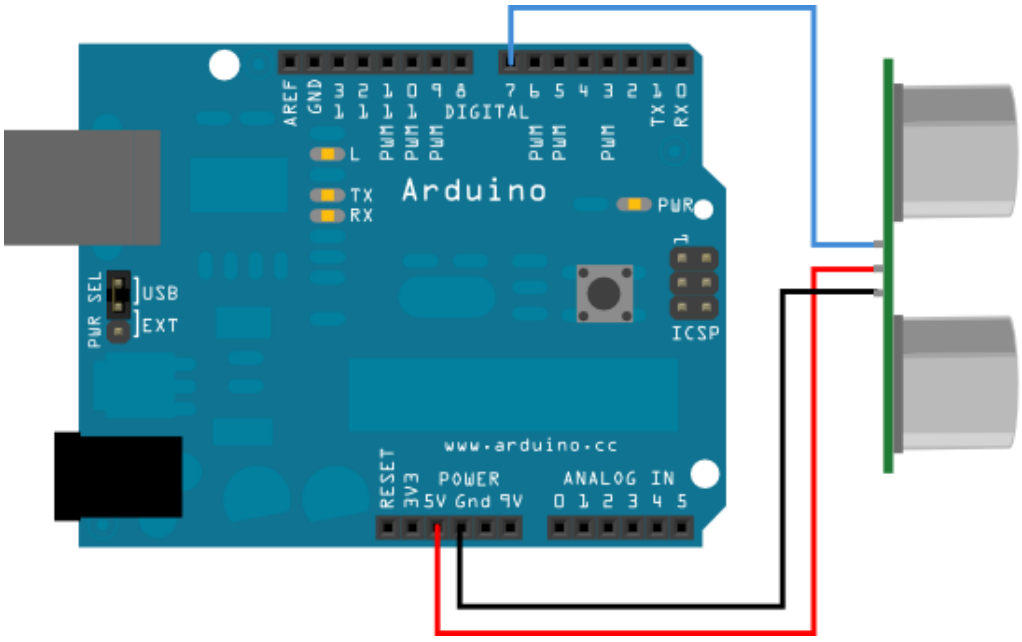

Fig 4.2 Ultrasonic Senor Arduino Connection

Understanding Ultrasonics Ultrasonic Sensing/Control Basics Ultrasonic signals are like audible sound waves, except the frequencies are much higher. Our ultrasonic transducers have piezoelectric crystals which resonate to a desired frequency and convert electric energy into acoustic energy and vice versa. The illustration shows how sound waves, transmitted in the shape of a cone, are reflected from a target back to the transducer. An output signal is produced to perform some kind of indicating or control function. A minimum distance from the sensor is required to provide a time delay so that the "echoes" can be interpreted. Variables which can effect the operation of ultrasonic sensing include: target surface angle, reflective surface roughness or changes in temperature or humidity. The targets can have any kind of reflective form – even round objects.

When used for sensing functions, the ultrasonic method has unique advantages over conventional sensors:

Discrete distances to moving objects can be detected and measured.

Less affected by target materials and surfaces, and not affected by color. Solid-state units have virtually unlimited, maintenance free life. Can detect small objects over long operating distances.

Resistance to external disturbances such as vibration, infrared radiation, ambient noise, and EMI radiation.

4.2 ULTRASONIC SENSOR TYPES

The following diagrams summarize the distinctions between proximity and ranging ultrasonic sensors:

Proximity Detection

An object passing anywhere within the preset range will be detected and generate an output signal. The detect point is independent of target size, material, or degree of reflectivity.

Ranging Measurement

Precise distance(s) of an object moving to and from the sensor are measured via time intervals between transmitted and reflected bursts of ultrasonic sound. The example shows a target detected at six inches from sensor and moving to 10 inches. The distance change is continuously calculated and outputted.

Target Angle

This term refers to the "tilt response" limitations of a given sensor. Since ultrasonic sound waves reflect off the target object, target angles indicate acceptable amounts of tilt for a given sensor. If an application requires a target angle beyond the capabilities of a single sensor, two sensors can be teamed up to provide an even broader angle of tilt.

Beam Spread

This term is defined as the area in which a round wand will be sensed if passed through the target area. This is the maximum spreading of the ultrasonic sound as it leaves the transducer.

Distance Hysteresis Option

Some standard proximity sensors can be furnished with a “hysteresis control” capability. Many hysteresis sensors have a fixed detect point equal to the sensor’s minimum range, and an adjustable turn off point up to the sensor’s maximum range . Consult factory for hysteresis option.

Thru-Beam Option

The TBT/TBR-600-40 is primarily a Thru-Beam sensor consisting of one transmit and one receive transducer, in separate and self-contained housings.

High Gain Option

The RPS-300 and 325 can be furnished with a high gain capability. This is denoted by the letter (G) in the part number. (Example: RPS-300G-14)

Analog Output Option

Analog outputs of 0-10 VDC and 4-20 mA are available in a variety of models. See the selection card and appropriate data sheet for details. Note: In some models, analog output is an option indicated by adding (-500) to the part number. (Example: RP S-100-14-500)

Analog Ranging Card

For converting proximity sensors into RANGING sensors
When connected to any RPS-100/300/325/326 proximity sensor, an RPS-500 ANALOG RANGING CARD measures discrete distances to a target within a selected range, then outputs one or more signals to a controller or computer.

DESCRIPTION RPS-500

OUTPUTS: Analog voltage (0-10 VDC) and current (4-20 mA), linearly proportional to target distance. Has adjustable zeroing and span control, plus inverting switch. Outputs are isolated up to 2500V. Powered by 120VAC. ANALOG RANGING CARD RPS-500.

When used for sensing functions, the ultrasonic method has unique advantages over conventional sensors:

- Discrete distances to moving objects can be detected and measured.
- Less affected by target materials and surfaces, and not affected by color. Solid-state units have virtually unlimited, maintenance free life. Can detect small objects over long operating distances.
- Resistance to external disturbances such as vibration, infrared radiation, ambient noise, and EMI radiation.

4.3 ULTRASONIC SENSOR WORKING

A transmitter sends out a signal (an ultrasonic wave) that humans cannot hear. And a receiver receives the signal after it has reflected off nearby objects. The sensor times how long it takes for its signals to come back and relays that information to the LEGO brick/computer, which calculates how far away objects are. Ask the students: Does this process sound familiar? Do you know of any animal that are able to sense in a similar way? (Answer: Bats, dolphins, porpoises and some whales use echolocation, sometimes called biosonar.)

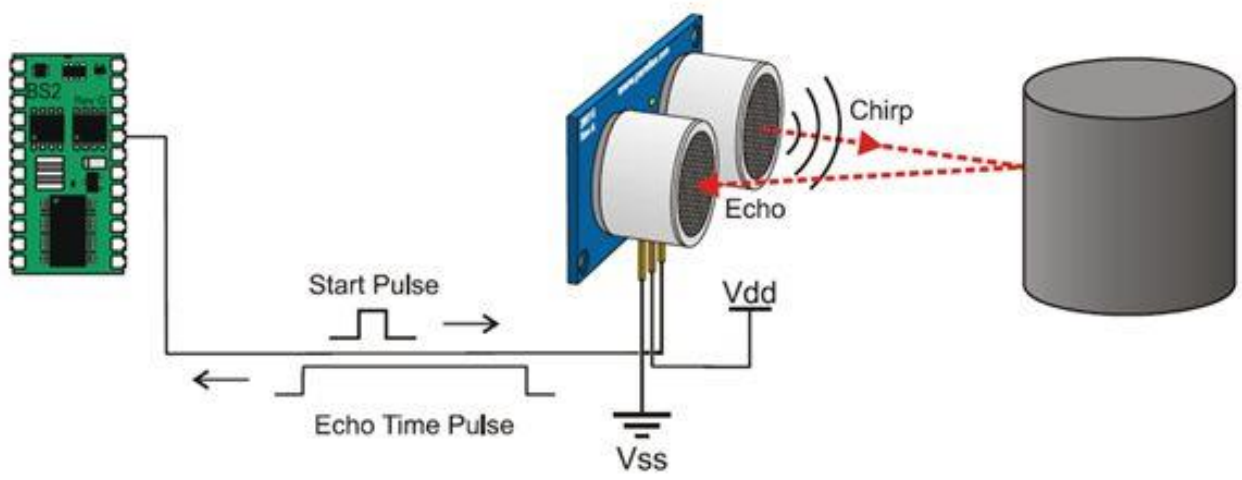

Fig 4.3 Ultrasonic Senor Arduino Circuit

Go into the details of the same concepts, but with some specifics on measuring distance. The ultrasonic sensor can measure distances in centimeters and inches. It can measure from 0 to 2.5 meters, with a precision of 3 cm. It provides good readings in sensing large-sized objects with hard surfaces, but has more difficulties reading reflections from soft, curved, thin or small objects.
The ultrasonic sensor sends out sound from one side (one of the "eyes") and receives it back using the sensor on the other side. Together, the ultrasonic sensor's transmitter and receiver look like a pair of eyes, but it is not a sight sensor. Instead, those "eyes" are really more like a speaker and a microphone (a sound sensor).

For the mini-activities, divide the class into pairs and give each pair a LEGO EV3 ultrasonic sensor, motor, intelligent brick and two cables. Direct student pairs to follow the instructions to experiment with the EV3 ultrasonic sensor.

In the first activity, they spend ~10 minutes using the ultrasonic sensor connected to the brick to study how the sensor works and what it does. Make sure they do the "Try Me" option and write down their observations from their experimentation.

In the second activity, they add the motor to the setup and spend ~15 minutes programming the LEGO robot so that it plays one music when an object is close to it (less than 10 inches), and keeps the motor turning.

If the object is farther than 10 inches, have the program stop. On a separate sheet of paper, have students provide step-by-step explanations describing how the program works. The programming solution is provided .

CHAPTER 5

DC MOTORS

5.1 DC MOTOR DESCRIPTION

A DC Motor is a type of electric motor that converts DC electrical power to mechanical power i.e. a DC supply is converted to rotation or movement. DC motors are one of the commonly used motors in different applications like electronic toys, power tools, portable fans, etc.

DC Motors are further classified in to different types like series, shunt and compound and each type is used in different areas of applications. Some DC motors are also used in Robotic and Industrial applications for their easy control and precision.

Since DC motors are generally associated with small to medium applications, where the system mainly consists of a Microcontroller as the main processing unit, controlling and driving a DC motor is very important. This is because, driving a motor directly using the microcontroller is not advised (sometimes not possible) as the current from the Microcontroller is very small (usually less than 30mA.

Fig 5.1 DC Motor

5.2 DC MOTOR WORKING

The aim of this project is to design an Arduino based system for controlling a DC Motor. All the connections are made as per the circuit diagram mentioned above. The working of the project is very simple and is explained here.

Two buttons are used in this project, one each for forward and reverse direction of the motor. The two buttons are connected to Pins 13 and 12 of Arduino which are internally pulled-up (using code). The other terminals of the buttons are connected to ground and hence when the button is pressed, the microcontroller detects LOW (logic 0).

The output of the POT is an analog signal and hence it is connected to analog pin of the Arduino. Based on the analog voltage value from the POT, the speed of the motor is varied.

For this to happen, we need to use the concept of PWM in the circuit. The inputs to the motor driver IC must be in the form of a PWM signal and hence are connected to Pins 11 and 10 of Arduino respectively, which are capable of generating PWM signals.

When the system is powered ON, Arduino waits for the button to be pressed. If the forward direction button is pressed, the Arduino drives input 1 of motor driver IC (Pin 2) with PWM signal and a logic low to input 2 (Pin 3). Hence, the motor starts rotating in forward direction.

Similarly, if the reverse direction button is pressed, Arduino drives input 2 (Pin 3) of L293D Motor Driver IC with the PWM signal and input 1 (pin 2) of L293D is given a logic low. Hence, the motor starts rotating in reverse directions.

The speed of the motor in either direction can be controlled using the POT as it controls the duty cycle of the output PWM signal.

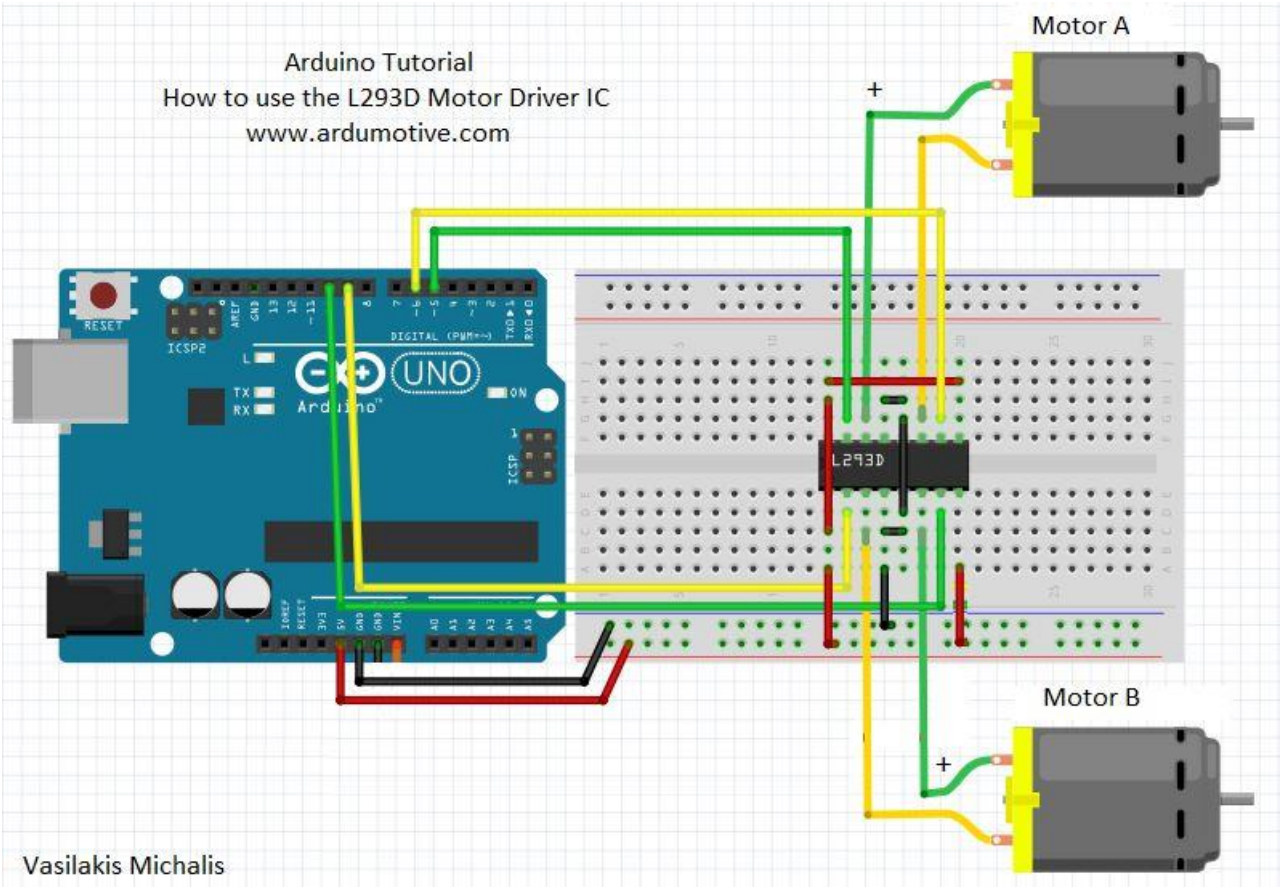

Fig 5.2 DC Motor Arduino Circuit

CHAPTER 6

LIPO BATTERY

6.1 LIPO BATTERY DESCRIPTION

Lithium Polymer batteries (henceforth referred to as “LiPo” batteries), are a newer type of battery now used in many consumer electronics devices. They have been gaining in popularity in the radio control industry over the last few years, and are now the most popular choice for anyone looking for long run times and high power.

LiPo batteries offer a wide array of benefits. But each user must decide if the benefits outweigh the drawbacks. For more and more people, they do. In my personal opinion, there is nothing to fear from LiPo batteries, so long as you follow the rules and treat the batteries with the respect they deserve.

This guide was written after many hours of research. It is as accurate as I can make it without actually being a chemical engineer (though, in researching this article, I did talk to a few of them). That said, this guide isn't intended to be taken as definitive. It is a living document, and as common knowledge regarding LiPo batteries changes, so too will this guide.

Let's first talk about the differences between LiPo batteries and their Nickel-Cadmium and Nickel-Metal Hydride counterparts.

Pros

Much lighter weight, and can be made in almost any size or shape.Much higher capacities, allowing them to hold much more power.Much higher discharge rates, meaning they pack more punch.

Cons

Much shorter lifespan; LiPos average only 150–250 cycles.The sensitive chemistry can lead to fire if the battery gets punctured.Need special care for charging, discharging, and storage.

They way we define any battery is through a ratings system. This allows us to compare the properties of a battery and help us determine which battery pack is suitable for the need at hand. There are three main ratings that you need to be aware of on a LiPo battery.

Fig 6.1 LIPO Battery

Voltage / Cell Count

A LiPo cell has a nominal voltage of 3.7V. For the 7.4V battery above, that means that there are two cells in series (which means the voltage gets added together). This is sometimes why you will hear people talk about a "2S" battery pack - it means that there are **2** cells in **S**eries. So a two-cell (2S) pack is 7.4V, a three-cell (3S) pack is 11.1V, and so on.
In the early days of LiPo batteries, you might have seen a battery pack described as "2S2P". This meant that there were actually four cells in the battery; two cells wired in series, and two more wired into the first two batteries in parallel (parallel meaning the capacities get added together).

This terminology is not used much nowadays; modern technology allows us to have the individual cells hold much more energy than they could only a few years ago. Even so, it can be handy to know the older terms, just in case you run into something with a few years on it.

The voltage of a battery pack is essentially going to determine how fast your vehicle is going to go. Voltage directly influences the RPM of the electric motor (brushless motors are rated by kV, which means 'RPM per Volt'). So if you have a brushless motor with a rating of 3,500kV, that motor will spin 3,500 RPM for every volt you apply to it. On a 2S LiPo battery, that motor will spin around 25,900 RPM. On a 3S, it will spin a whopping 38,850 RPM. So the more voltage you have, the faster you're going to go.

Capacity

The capacity of a battery is basically a measure of how much power the battery can hold. Think of it as the size of your fuel tank. The unit of measure here is milliamp hours (mAh). This is saying how much drain can be put on the battery to discharge it in one hour. Since we usually discuss the drain of a motor system in amps (A), here is the conversion:

I said that the capacity of the battery is like the fuel tank - which means the capacity determines how long you can run before you have to recharge. The higher the number, the longer the run time. Airplanes and helicopters don't really have a standard capacity, because they come in many different sizes, but for R/C cars and trucks, the average is 5000mAh - that is our most popular battery here in the store. But there are companies that make batteries with larger capacities. Traxxas even has one that is over 12000mAh! That's huge, but there is a downside to large capacities as well. The bigger the capacity, the bigger the physical size and weight of the battery. Another consideration is heat build up in the motor and speed control over such a long run. Unless periodically checked, you can easily burn up a motor if it isn't given enough time to cool down, and most people don't stop during a run to check their motor temps. Keep that in mind when picking up a battery with a large capacity.

6.2 LIPO BATTERY WORKING

Voltage and Capacity had a direct impact on certain aspects of the vehicle, whether it's speed or run time. This makes them easy to understand. The Discharge Rating (I'll be referring to it as the C

Rating from now on) is a bit harder to understand, and this has lead to it being the most over-hyped and misunderstood aspects of LiPo batteries.

The C Rating is simply a measure of how fast the battery can be discharged safely and without harming the battery. One of the things that makes it complicated is that it's not a stand-alone number; it requires you to also know the capacity of the battery to ultimately figure out the safe amp draw (the "C" in C Rating actually stands for **C**apacity). Once you know the capacity, it's pretty much a plug-and-play math problem. Using the above battery, here's the way you find out the maximum safe continuous amp draw:
The resulting number is the maximum sustained load you can safely put on the battery. Going higher than that will result in, at best, the degradation of the battery at a faster than normal pace. At worst, it could burst into flames. So our example battery can handle a maximum continuous load of 250A.

Most batteries today have two C Ratings: a Continuous Rating (which we've been discussing), and a Burst Rating. The Burst rating works the same way, except it is only applicable in 10-second bursts, not continuously. For example, the Burst Rating would come into play when accelerating a vehicle, but not when at a steady speed on a straight-away. The Burst Rating is almost always higher than the Continuous Rating. Batteries are usually compared using the Continuous Rating, not the Burst Rating.

There is a lot of vitriolic comments on the Internet about what C Rating is best. Is it best to get the highest you can? Or should you get a C Rating that's just enough to cover your need? There isn't a simple answer. All I can give you is my take on the issue. When I set up a customer with a LiPo battery, I first find out what the maximum current his or her application will draw. Let's look at how that works
However, the ratings on the motor aren't the whole picture. The way the truck is geared, the terrain the truck is driving on, the size of the tires, the weight of the truck... all of these things have an impact on the final draw on the battery. It's very possible that the final draw on the battery is higher than the maximum motor draw. So having that little bit of overhead is crucial, because you can't easily figure out a hard number that the truck will never go over.

For most applications, a 20C or 25C battery should be fine. But if you're driving a heavy truck, or you're geared up for racing, or you have a large motor for 3D flying applications, you should probably start around a 40C battery pack. But since there is no easy way to figure this out, I encourage you to talk to your local hobby shop to have them help determine which battery pack is right for your application.

it's important to use a LiPo compatible charger for LiPos. As I said in the Introduction, LiPo batteries require specialized care. They charge using a system called CC/CV charging. It stands for **C**onstant **C**urrent / **C**onstant **V**oltage. Basically, the charger will keep the current, or charge rate, constant until the battery reaches its peak voltage (4.2v per cell in a battery pack). Then it will

maintain that voltage, while reducing the current. On the other hand, NiMH and NiCd batteries charge best using a pulse charging method. Charging a LiPo battery in this way can have damaging effects, so it's important to have a LiPo-compatible charger.

The second reason that you need a LiPo-compatible charger is balancing. Balancing is a term we use to describe the act of equalizing the voltage of each cell in a battery pack. We balance LiPo batteries to ensure each cell discharges the same amount. This helps with the performance of the battery. It is also crucial for safety reasons - but I'll get to that in the section on discharging.

While there are stand-alone balancers on the market, I recommend purchasing a charger with built-in balancing capabilities, using a balance board like the one pictured to the right. This simplifies the process of balancing, and requires one less thing to be purchased. And with the price of chargers with built-in balancers coming down to very reasonable levels, I can't think of a reason you would not want to simplify your charging set up. We'll talk more about chargers in the next section.

Most LiPo batteries come with a connector called a JST-XH connector on the balance tap. One of the big problems with this connector is it's lack of surface area; namely, one's inability to get a good grip on the connector. This makes it hard to unplug from a balance board, and a user usually just ends up pulling on the wires. This can break the connector, and potentially short out the battery. A unique product, called Balance Protector Clips (or AB Clips) is a great way to solve this problem. They clip around the balance connector, and give a user more space to grab on to the it. They are usually inexpensive, and a great way to prevent balance connector fatigue. To the left, you can see a balance connector with and without the Balance Protector Clips.

Most LiPo batteries need to be charged rather slowly, compared to NiMH or NiCd batteries. While we would routinely charge a 3000mAh NiMH battery at four or five amps, a LiPo battery of the same capacity should be charged at no more than three amps. Just as the C Rating of a battery determines what the safe continuous discharge of the battery is, there is a C Rating for charging as well. For the vast majority of LiPos, the Charge Rate is **1C**. The equation works the same way as the previous discharge rating, where 1000mAh = 1A. So, for a 3000mAh battery, we would want to charge at 3A, for a 5000mAh LiPo, we should set the charger at 5A, and for a 4500mAh pack, 4.5A is the correct charge rate.

However, more and more LiPo batteries are coming out these days that advertise faster charging capabilities, like the example battery we had above. On the battery, the label says it has a "3C Charge Rate". Given that the battery's capacity is 5000mAh, or 5 Amps, that means the battery can be safely charged at a maximum of 15 Amps! While it's best to default at a 1C charge rate, always defer to the battery's labeling itself to determine the maximum safe charge rate.

Due to the potential for fire when using LiPo batteries, regardless of the likelihood, certain precautions should be taken. Always have a fire extinguisher nearby; it won't put out a LiPo fire (as I will further explain below, LiPo fires are chemical reactions and are very hard to put out). But a

fire extinguisher will contain the fire and stop it from spreading. I prefer a CO2 (Carbon Dioxide) extinguisher - it helps to remove oxygen from the burn site, and will also cool down the battery and surrounding items. Another safety precaution is to charge the LiPo in a fire-resistant container. Most people opt toward the LiPo Bags on the market today, like the one pictured to the left. They are a bit pricy, but are more portable than other solutions. Finally, **never** charge your LiPo batteries unattended! If something does happen, you needs to be around to react quickly. While you don't have to always be in the same room, you shouldn't leave the house, or go mow the lawn, or anything else that will prevent you from taking action should the battery catch fire.

Parallel vs. Series Charging

A wonderful gentleman from the Netherlands contacted me recently asking about parallel charging versus series charging. He wanted to know how best to charge six of his single-cell LiPo batteries at the same time. Parallel charging adapters are readily available, so that must be the best way, right?

Parallel charging can be very dangerous. Even experts from well-known battery manufacturers "consider parallel pack charging to be highly dangerous and should not be attempted even by experienced users". The problem with parallel charging (or even using your batteries in parallel) is that, when hooking up batteries in parallel, you are doubling the capacity of the batteries while, and this is important, maintaining the voltage of one of the individual batteries. What this means is that your charger, which normally monitors the battery while charging to prevent overcharging, cannot see all of the individual batteries' voltages - it can only see one.

Another problem with parallel charging is the inequality of the batteries. If the two batteries (and the cells contained therein) were from the exact same production lot, had the exact same chemical composition and age and charge history and everything else - in other words, if they were completely identical - parallel charging would be okay. But a consumer (that's you) will never be able to replicate those conditions, or even come close. The more those parameters differ, and considering the questionable balance charging techniques that many battery chargers use, the higher the chance of over charging and thermal runaway (more on that in the next section).

CHAPTER 7

PROGRAMMING

```
//float distance[3];

#define motor1 10

#define motor2 11

#define motor3 5

#define motor4 6

#define circle 1360   // Time in millis to complete a full circle

const float minDist = 20.0;

float dLeft, dRight, dCenter;

long duration;


int trigPin[3] = {12, 3, 8};  // Left, Right, Center

int echoPin[3] = {13, 4, 9};



float findDist(int trigPin, int echoPin)

{

  digitalWrite(trigPin , LOW);

  delayMicroseconds(2);
```

```
  digitalWrite(trigPin , HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin , LOW);

  float duration = pulseIn(echoPin, HIGH);
  return duration * 0.034 / 2;
}

void setup() {
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);
  pinMode(motor3, OUTPUT);
  pinMode(motor4, OUTPUT);

  for (int i = 0; i < 3; i++) {
    pinMode(trigPin[i], OUTPUT);
    pinMode(echoPin[i], INPUT);
  }
  Serial.begin(9600);
```

```
}


void loop() {
  dCenter = findDist(trigPin[2], echoPin[2]);
  dRight = findDist(trigPin[1], echoPin[1]);
  dLeft = findDist(trigPin[0], echoPin[0]);


  Serial.println(dCenter);


  if (dCenter < minDist + 5) {
    reverse(1000);
    turnAbout();
  } else if (dRight < minDist) {
    turnLeft(circle / 4);
  } else if (dLeft < minDist) {
    turnRight(circle / 4);
  } else {
    forward(100);
  }
}
```

```
void forward(int millisec) {
  Serial.println("Forward");
  analogWrite(motor1, 180);
  analogWrite(motor2, 0);
  analogWrite(motor3, 180);
  analogWrite(motor4, 0);
  delay(millisec);
}

void reverse(int millisec) {
  Serial.println("Reverse");
  analogWrite(motor1, 0);
  analogWrite(motor2, 180);
  analogWrite(motor3, 0);
  analogWrite(motor4, 180);
  delay(millisec);
}
void turnLeft(int millisec) {
  Serial.println("Left");
```

```
  analogWrite(motor1, 0);
  analogWrite(motor2, 180);
  analogWrite(motor3, 180);
  analogWrite(motor4, 0);
  delay(millisec);
}

void stopMove(int millisec) {
  Serial.println("Stop");
  analogWrite(motor1, 0);
  analogWrite(motor2, 0);
  analogWrite(motor3, 0);
  analogWrite(motor4, 0);
  delay(millisec);
}

void turnRight(int millisec) {
  Serial.println("Right");
  analogWrite(motor1, 180);
  analogWrite(motor2, 0);
```

```
  analogWrite(motor3, 0);
  analogWrite(motor4, 180);
  delay(millisec);
}

void turnAbout() {
  Serial.println("Reverse");
  analogWrite(motor1, 180);
  analogWrite(motor2, 0);
  analogWrite(motor3, 0);
  analogWrite(motor4, 180);
  delay(circle / 2);
}

   else
   {
     lcd.setCursor(0,1);
     lcd.print("PLZ Wear Helmet");
    digitalWrite(m1,LOW);//motor
    //digitalWrite(m2,HIGH);
```

```
    }

}
```

CHAPTER 11

FEATURES AND BENEFITS

11.1 Benefits:

- Detection of accident in remote area can be easily detected and medical services provided in short time.
- Simply avoiding drunken drive by using alcohol detector. it will reduces the probability of accident
- Operates on solar as well as battery supply.

If helmet was stolen then we can start the bike by the password

11.2 Application:

- It can be used in real time safety system.
- We can implement the whole circuit into small module later.
- Less power consuming safety system.
- This safety system technology can further be enhanced in car and also by replacing the helmet with seat belt.