CHAPTER-1

INTRODUCTION

1. INTRODUCTION

Gestures provide an very easy interface with the objects surrounding us to control its activities. It is a innovative interface to both human and computer. Thus, such gesture-based interfaces cannot only substitute the common interface devices, but can also be exploited to extend their functionality.

1.1 OBJECTIVE

Our objective is to build a robotic vehicle which is controlled by the gestures made through android mobile and simulate the same using PROTEUS 7.9.

1.2 MOTIVATION

Our motivation to work on this project came from a disabled person who was driving his wheel chair by hand with quite a lot of difficulty and for the people who cannot move from one place to other. So we wanted to make a device which would help such people to drive their chairs without even touching the wheels of their chairs and control any electronic equipment from single place. Here we built one of such electronic appliance, robotic vehicle which is controlled by smartphone.

1.3 GESTURE CONTROLLED ROBOT

Gesture recognition technologies are much younger in the world of today. At this time there is much active research in the field and little in the way of publicly available implementations. Several approaches have been developed for sensing gestures and controlling robots.

A Gesture Controlled robot is a kind of robot which can be controlled by hand gestures and not by using buttons. The user just needs a android mobile on his hand which includes a sensor which is an accelerometer inbuilt in android phones. Movement of the hand in a specific direction will transmit a command to the robot through Bluetooth which will then move in a specific direction.

At the receiving end an RF Receiver module here Bluetooth module will receive the encoded data and transmits it to microcontroller. This data is then processed by a microcontroller and passed onto a motor driver to rotate the motors in a special configuration to make the robot move in the same direction as that of the hand with android phone tilts.

CHAPTER-2

LITERATURE REVIEW

2. LITERATURE REVIEW

There are many ways to control an electronic appliances such as robotic vehicle or home automation systems. Till now we knew that a RC car can be controlled through remote or by wired connection. After referring to those, we are here with a new method of controlling these type of electronic appliances by our body movement. This can be achieved by using the smartphone. Smartphone is the primary need of every person in this digital world. Everyone uses smartphone for communication, games, internet and for many other purposes. But we can use the same for our home automation. Here we are controlling a robotic vehicle with a smartphone. Our gesture controlled robot works on the accelerometer present in the android phone. It senses the movements of phone and sends the signals to the microcontroller present in the robot through wireless Bluetooth communication by UART (Universal Asynchronous Receiver and Transmitter). The microcontroller receives the signals. These signals are passed to the motor driver IC which triggers the motors in different configurations to make the robot move in a specific direction. Here we use PWM (Pulse Width Modulation) method to control the motors.

2.1 ROBOT

A Robot is usually an electro-mechanical machine that can perform tasks automatically. Some robots require some degree of guidance, which may be done using a remote control or with a computer interface. Robots can be autonomous, semi-autonomous or remotely controlled. Robots have evolved so much and are capable of mimicking humans that they seem to have a mind of their own.

2.2 HUMAN MACHINE INTERACTION

An important aspect of a successful robotic system is the Human-Machine interaction. In the early years the only way to communicate with a robot was to program which required extensive hard work. With the development in science and robotics, gesture based recognition came into life. Gestures originate from any motion or state but commonly originate from the face or hand. Gesture recognition can be considered as a way for computer to understand human body language. This has minimized the need for text interfaces and GUIs (Graphical User Interface).

2.3 GESTURE

A gesture is an action that has to be seen by someone else and has to convey some piece of information. Gesture is usually considered as a movement of part of the body or any object to express an idea or meaning.

2.4 PULSE WIDTH MODULATION (PWM)

Pulse-width modulation is a modulation technique that controls the width of the pulse, formally the pulse duration, based on modulator signal information. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors.

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load.

The PWM switching frequency has to be much faster than what would affect the load, which is to say the device that uses the power. Typically switching has to be done several times a minute in an electric stove, 120 Hz in a lamp dimmer, from few kilohertz (kHz) to tens of kHz for a motor drive and well into the tens or hundreds of kHz in audio amplifiers and computer power supplies.

The term *duty cycle* describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on.

The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle.

PWM has also been used in certain communication systems where its duty cycle has been used to convey information over a communications channel.

2.5 UART (Universal Asynchronous Receiver/Transmitter)

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

There are two primary forms of serial transmission: Synchronous and Asynchronous. Depending on the modes that are supported by the hardware, the name of the communication sub-system will usually include a 'A' if it supports Asynchronous communications, and a 'S' if it supports Synchronous communications. Both forms are described below.

Some common acronyms are:

UART Universal Asynchronous Receiver/Transmitter

USART Universal Synchronous-Asynchronous Receiver/Transmitter

Synchronous Serial Transmission

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows when to "read" the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

A form of Synchronous transmission is used with printers and fixed disk devices in that the data is sent on one set of wires while a clock or strobe is sent on a different wire. Printers and fixed disk devices are not normally serial devices because most fixed disk interface standards send an entire word of data for each clock or strobe signal by using a separate wire for each bit of the word.

Asynchronous Serial Transmission

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. (This requirement was set in the days of mechanical tele printers and is easily met by modern electronic equipment.)

After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver “looks” at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on. The sender does not know when the receiver has “looked” at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent. Because asynchronous data is "self-synchronizing", if there is no data to transmit, the transmission line can be idle.

CHAPTER-3

HARDWARE DEVELOPMENT

3. HARDWARE DEVELOPMENT

This chapter of the thesis concentrates on the use of Arduino, Bluetooth Module, L298N Motor driver as the hardware parts of the project. Arduino which controls the action of the robot through signals sent from it. L298N is a motor driver shield which can control two DC motors and drive up to 2 A per channel. A Bluetooth modem is a medium that enables Arduino to connect to the android device or smartphone. Thus, the modem is the bridge for the commands sent from a smartphone to the Arduino. In addition, to power up the gear motors with appropriate signals, there is a need for a circuit to be designed

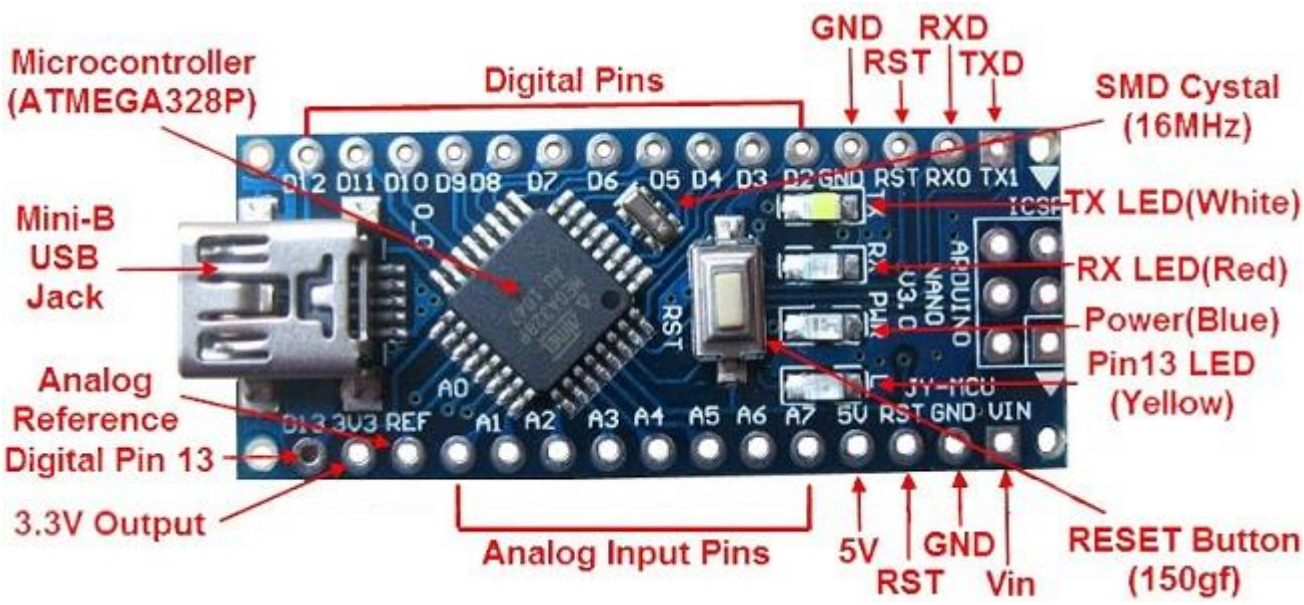
3.1 ARDUINO BOARD (ATMEGA 328p MCU)

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments."

An Arduino microcontroller is a simple yet sophisticated device, which has taken the world of electronics by storm. Because of its versatility in innovation, the product has gained several accolades from the electronics professionals. Further, it has made the newbies in electronics very enthusiastic about their possible future in electronics computing. The product's simplicity has allowed even the novel users to innovate different objects

ARDUINO NANO v3.0

The hardware used for the thesis is Arduino Nano V3.0. Arduino Nano is a microcontroller based on ATmega328. The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.0) . It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Nano was designed and is being produced by Gravitech.



Specifications:

Microcontroller	ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 Ma
Flash Memory	32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is

automatically selected to the highest voltage source. The FTDI FT232RL chip on the Nano is only powered if the board is being powered over USB. As a result, when running on external (non-USB) power, the 3.3V output (which is supplied by the FTDI chip) is not available and the RX and TXLEDs will flicker if digital pins 0 or 1 are high.

Memory

The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); the ATmega328has 32 KB, (also with 2 KB used for the bootloader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano 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:

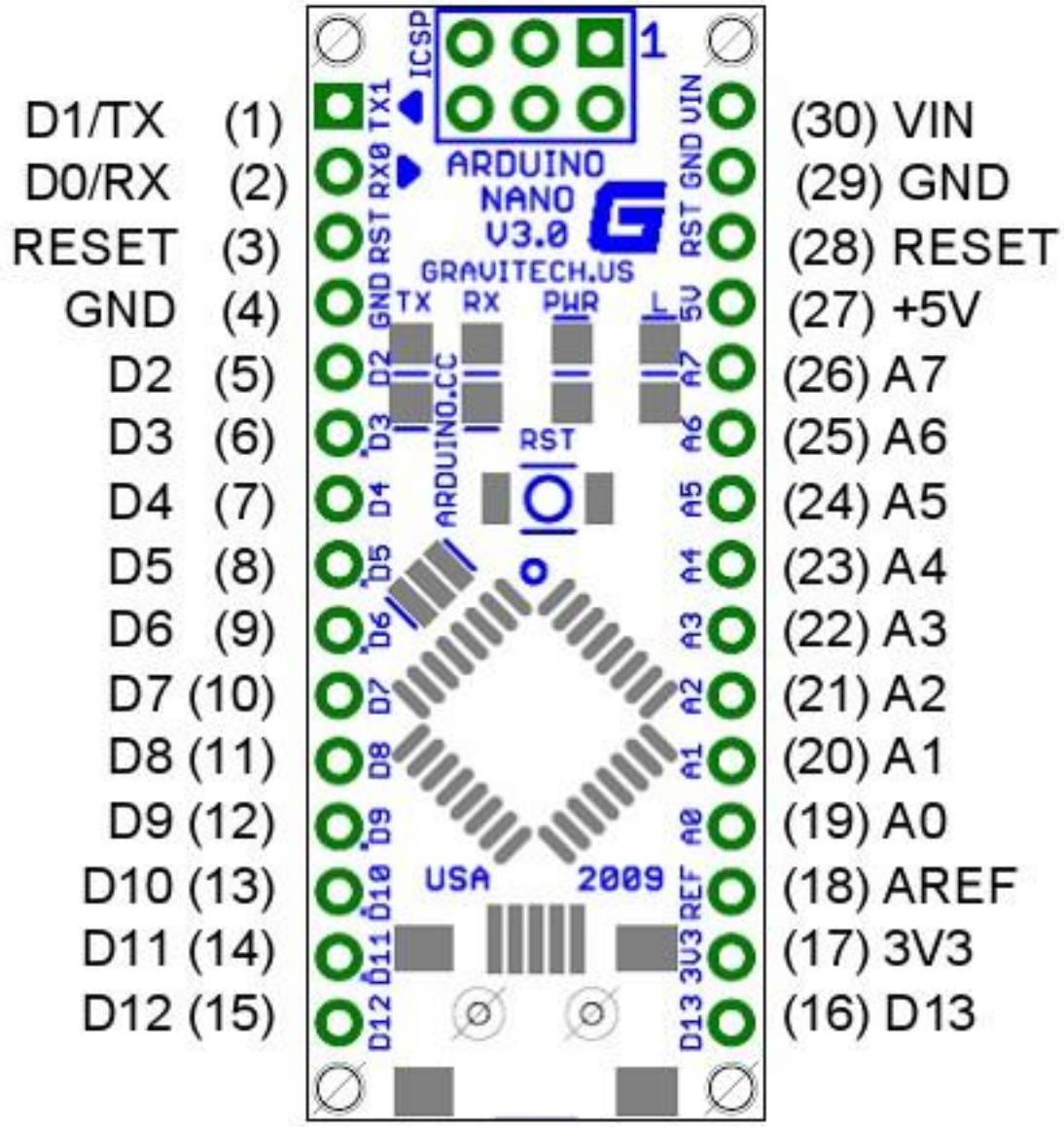
- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or 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, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- 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 Nano has 8 analog inputs, each of which provides 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 analogReference() function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- I^2C: 4(SDA) and 5(SCL). Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website).

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.



Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega168 and ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. 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 FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

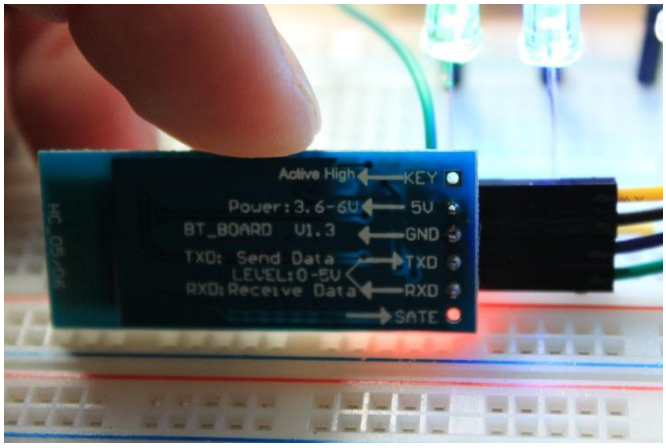
A Software Serial library allows for serial communication on any of the Nano's digital pins. The ATmega168 and ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see theATmega168 or ATmega328 datasheet.

Programming

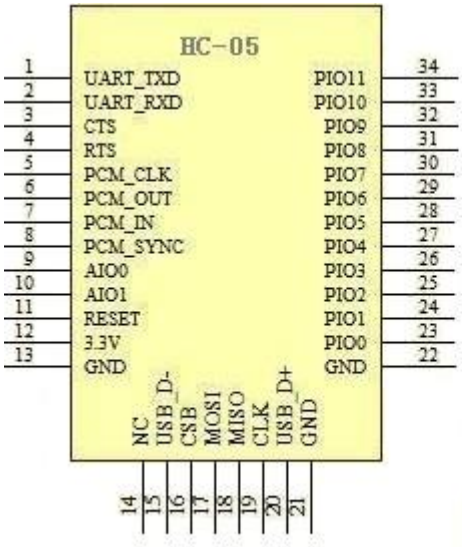
The Arduino Nano can be programmed with the Arduino software. Select "Arduino Diecimila, Duemilanove, or Nano w/ ATmega168" or "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board).

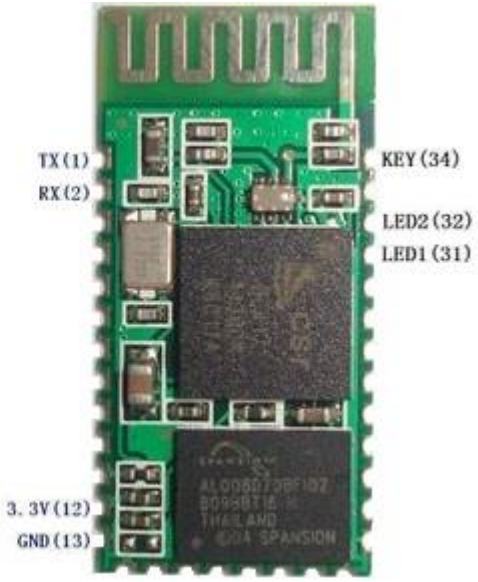
ATmega328 on the Arduino Nano comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

3.2 BLUETOOTH MODULE (HC-05)



HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.





Hardware features

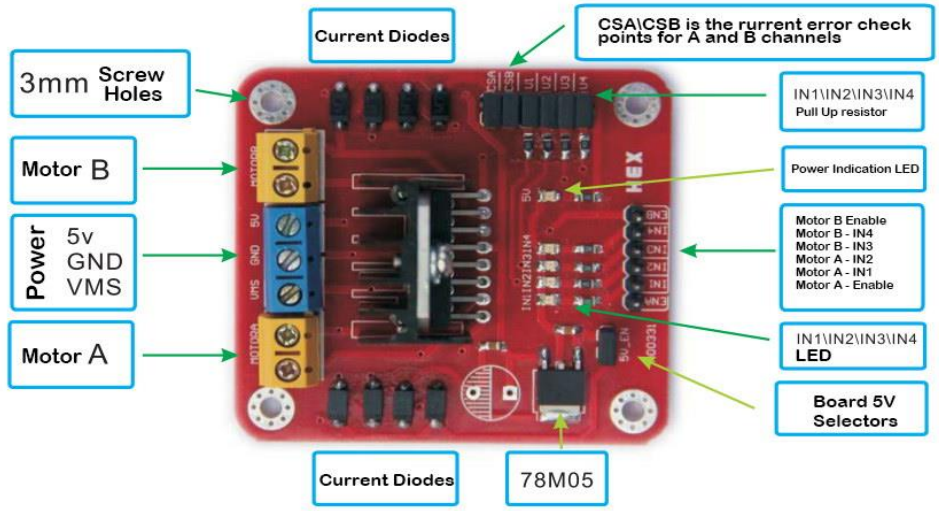
- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

3.3 L298N MOTOR DRIVER

The Motor Shield is based on the L298, which is a dual full-bridge driver designed to drive inductive loads such as relays, solenoids, DC and stepping motors. It lets you drive two DC motors or one stepper motor controlling the speed and direction of each one independently.



Operating Voltage 4V to 35V

Motor controller L298N, Drives 2 DC motors or 1 stepper motor

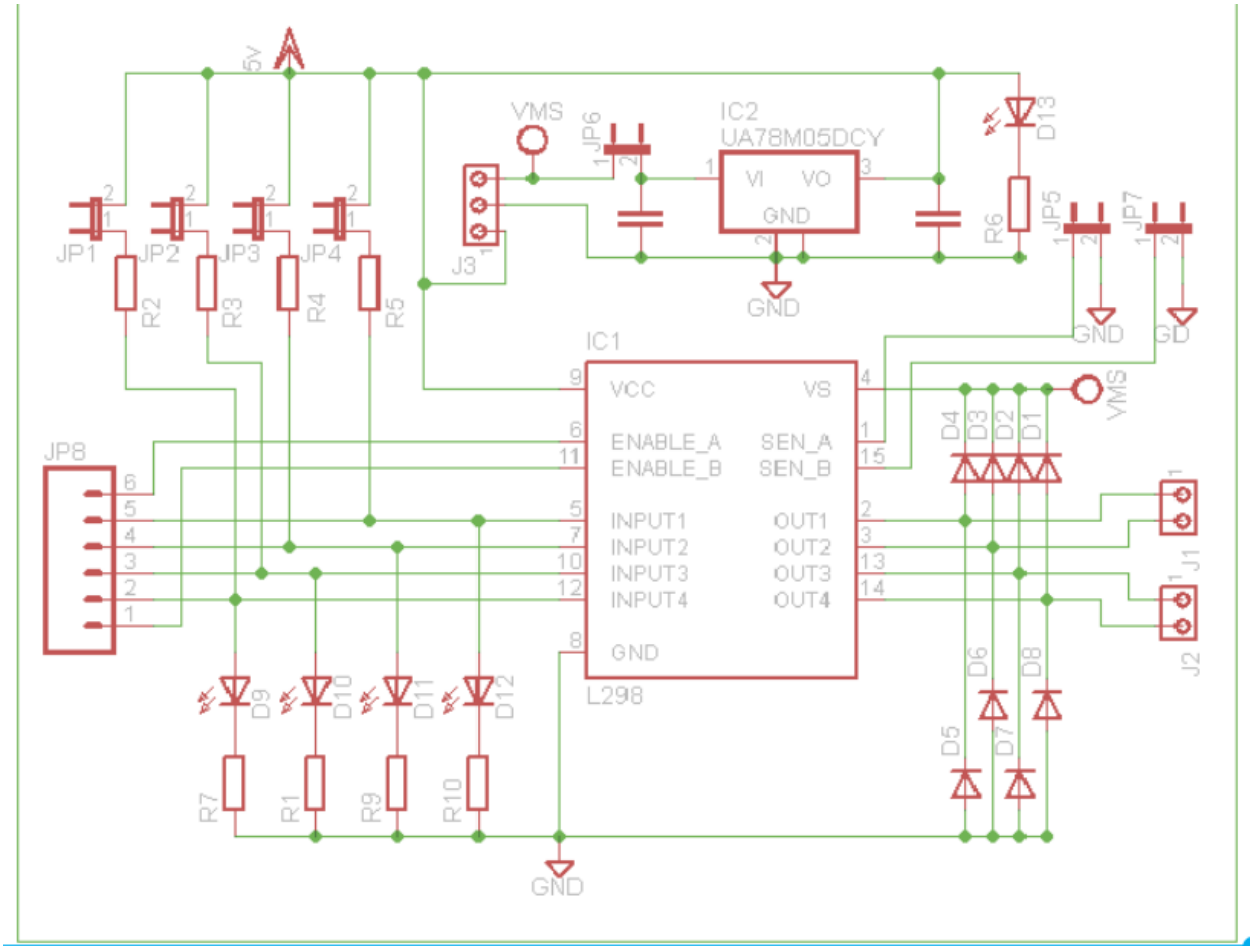
Max current 2A per channel or 4A max

Free running stop and brake function

Summary

- Chip: ST L298N
- Logic power supply:5v
- Max power:25w
- Storage temperature:-25 to +135

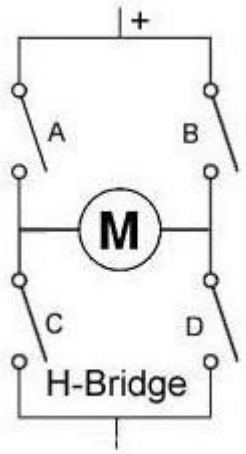
Schematic used for controlling DC motors



The L298 is an integrated monolithic circuit in a 15- lead Multi watt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

Turning a motor ON and OFF requires only one switch to control a single motor in a single direction. We can reverse the direction of the motor by simply reversing its polarity. This can be achieved by using four switches that are arranged in an intelligent manner such that the circuit not only drives the motor, but also controls its direction. Out of many, one of the most common and

clever design is a H-bridge circuit where transistors are arranged in a shape that resembles the English alphabet "H".



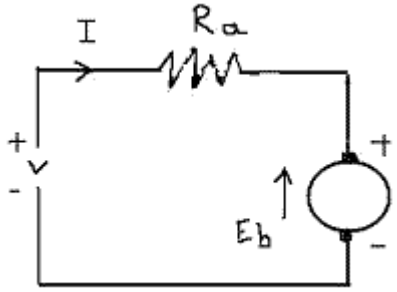
As seen in the image, the circuit has four switches A, B, C and D. Turning these switches ON and OFF can drive a motor in different ways.

- When switches A and D are on, motor rotates clockwise.
- When B and C are on, the motor rotates anti-clockwise.
- When A and B are on, the motor will stop.
- Turning off all the switches gives the motor a free wheel drive.
- Turning on A & C at the same time or B & D at the same time acts as a break

3.4 PERMANENT MAGNET DC MOTOR (PMDC MOTOR)

In a DC motor, an armature rotates inside a magnetic field. Basic working principle of DC motor is based on the fact that whenever a current carrying conductor is placed inside a magnetic field, there will be mechanical force experienced by that conductor. All kinds of DC motors work in this principle only. Hence for constructing a dc motor it is essential to establish a magnetic field. The magnetic field is obviously established by means of magnet. The magnet can by any types i.e. it may be electromagnet or it can be permanent magnet. When permanent magnet is used to create magnetic field in a DC motor, the motor is referred as **permanent magnet dc motor** or **PMDC motor**. Have you ever uncovered any battery operated toy, if you did, you had obviously found a battery operated motor inside it. This battery operated motor is nothing but a **permanent magnet dc motor** or **PMDC motor**. These types of motor are essentially simple in construction. These motors

are commonly used as starter motor in automobiles, windshield wipers, washer, for blowers used in heaters and air conditioners, to raise and lower windows, it also extensively used in toys. As the magnetic field strength of a permanent magnet is fixed it cannot be controlled externally, field control of this type of dc motor cannot be possible. Thus permanent magnet dc motor is used where there is no need of speed control of motor by means of controlling its field. Small fractional and sub fractional kW motors now constructed with permanent magnet.



Equivalent Circuit of Permanent Magnet DC Motor or PMDC Motor

As in PMDC motor the field is produced by permanent magnet, there is no need of drawing field coils in the equivalent circuit of permanent magnet dc motor. The supply voltage to the armature will have armature resistance drop and rest of the supply voltage is countered by back emf of the motor.

$$V = IR + E_b$$

Applications of Permanent Magnet DC Motor or PMDC Motor

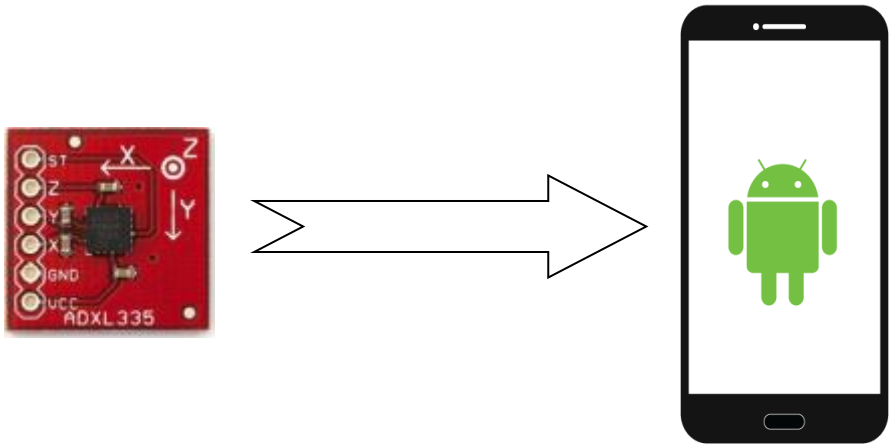
PMDC motor is extensively used where small dc motors are required and also very effective control is not required, such as in automobiles starter, toys, wipers, washers, hot blowers, air conditioners, computer disc drives and in many more.

3.5 ANDROID SMARTPHONE

Smartphone consists of inbuilt Bluetooth and accelerometer to sense the movement of smart phone. An Accelerometer is an electromechanical device that measures acceleration forces. These forces may be static, like the constant force of gravity pulling at your feet, or they could be dynamic caused by moving or vibrating the accelerometer. It is a kind of sensor which record acceleration

and gives an analog data while moving in X,Y,Z direction or may be X,Y direction only depending on the type of the sensor.

ACCELEROMETER SMARTPHONE

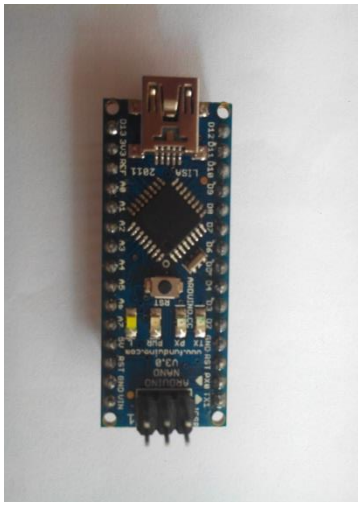


3.6: ROBOT CHASSIS, WHEELS, BATTERY ETC..

These are some of the other components used to build a robotic vehicle. It consists of robotic chassis, wheeles, gears, screws, headers, PCB board, battery etc… All these components are assembled to build a robotic vehicle.

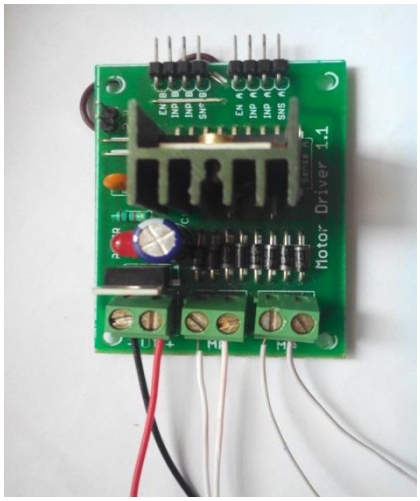
3.7 ASSEMBLING THE HARDWARE

Arduino nano v3.0



Bluetooth HC-05

L298N Motor Driver Board

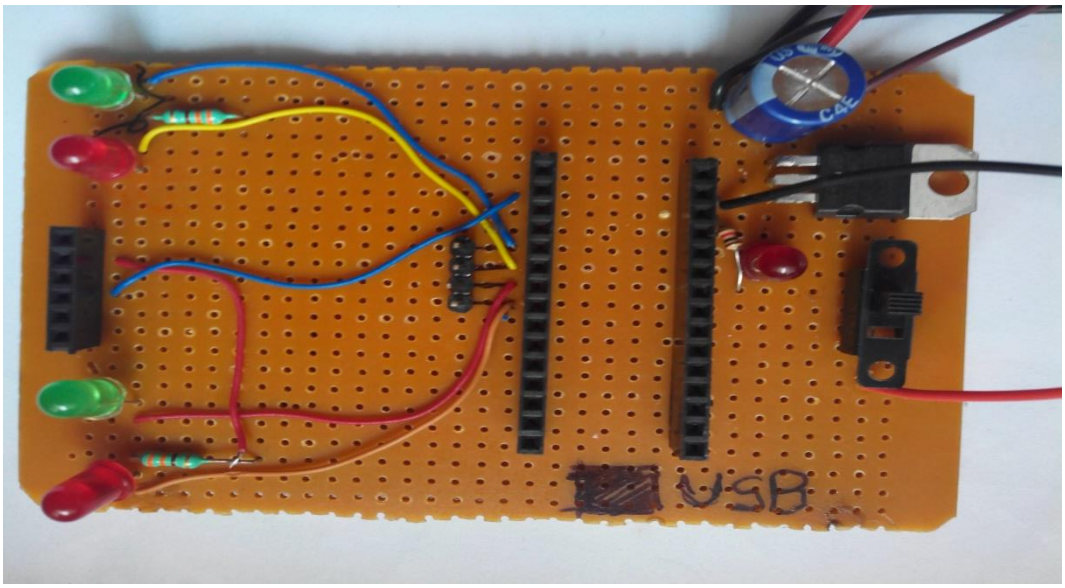


Robotic Chassis with Wheels & Motors

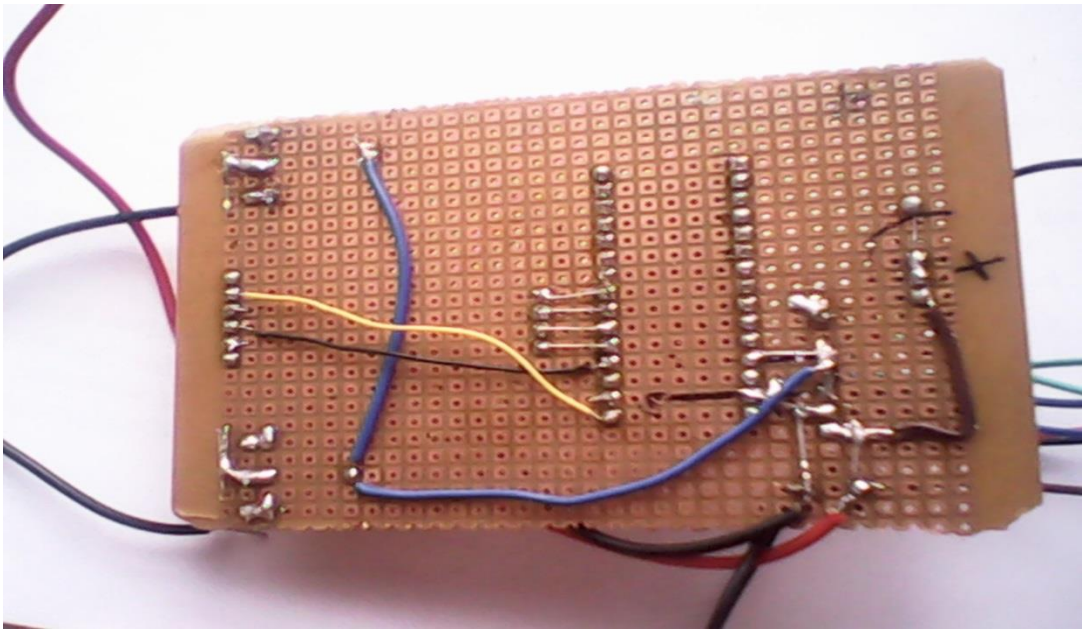
Battery

- Take a Base Board and solder the headers, LEDs, Voltage Regulator circuit to give a supply of 5V to the entire circuit.
- Use a resistors for limiting the voltage and current for diodes and for other low power components

The Board with passive components, voltage regulator etc

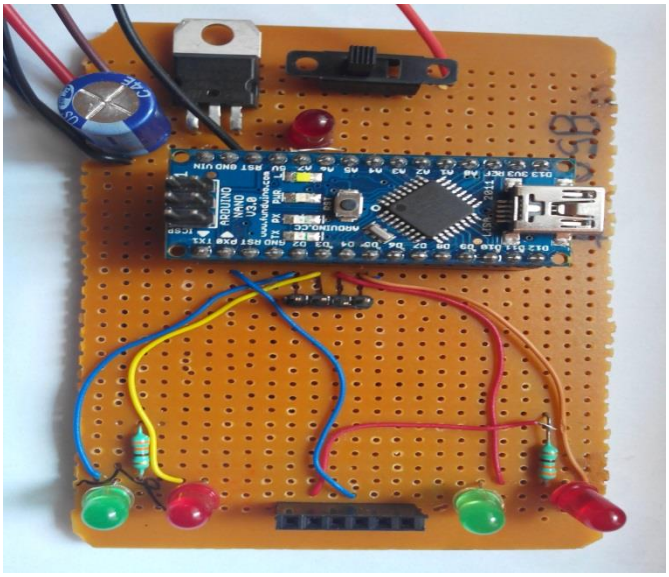


Front View

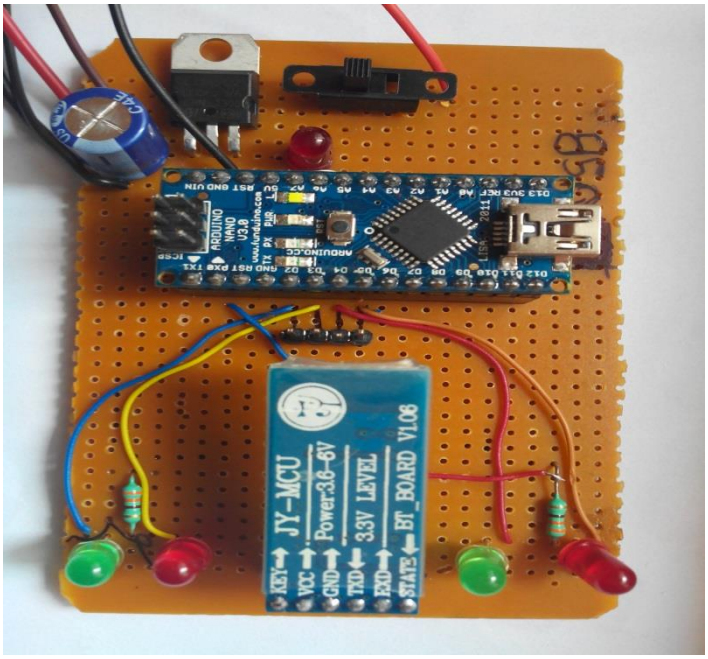


Back view

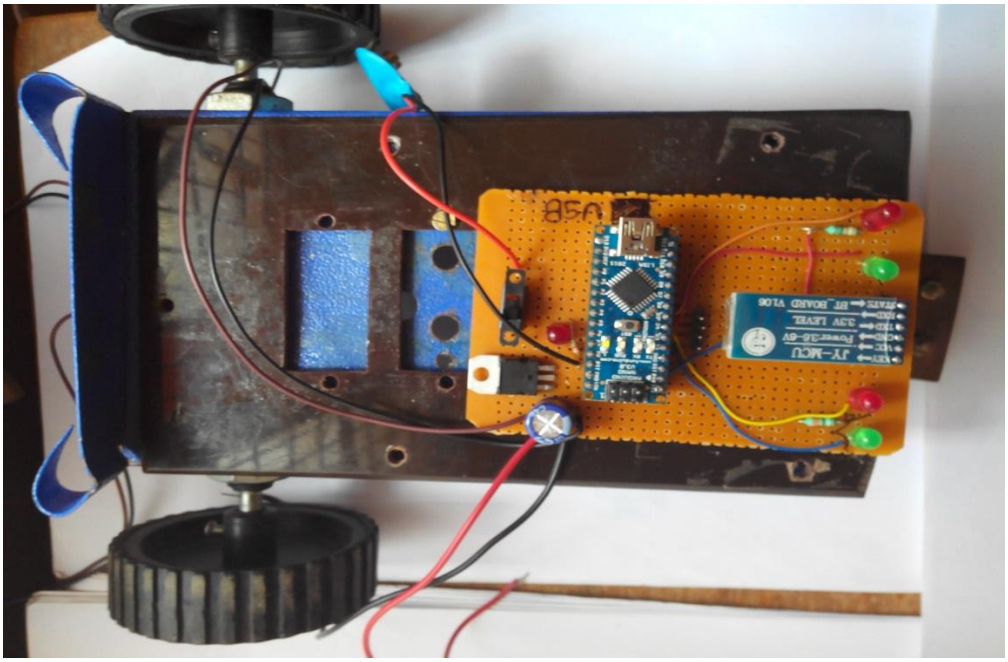
- *Arduino nano attached into the circuit board*



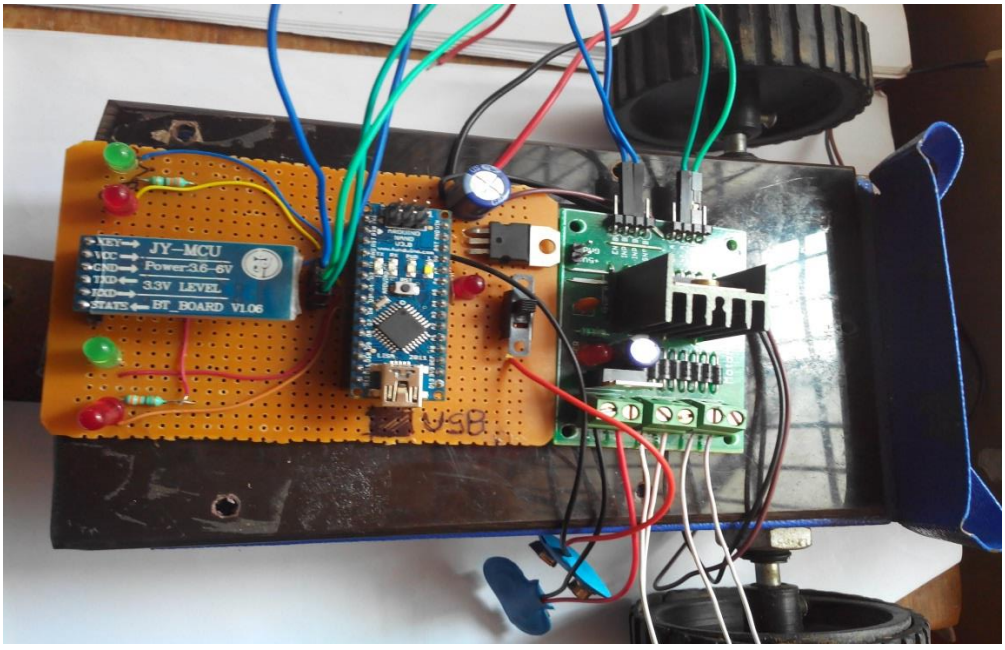
- *Bluetooth Module HC-05 and Arduino Nano attached into the circuit board*



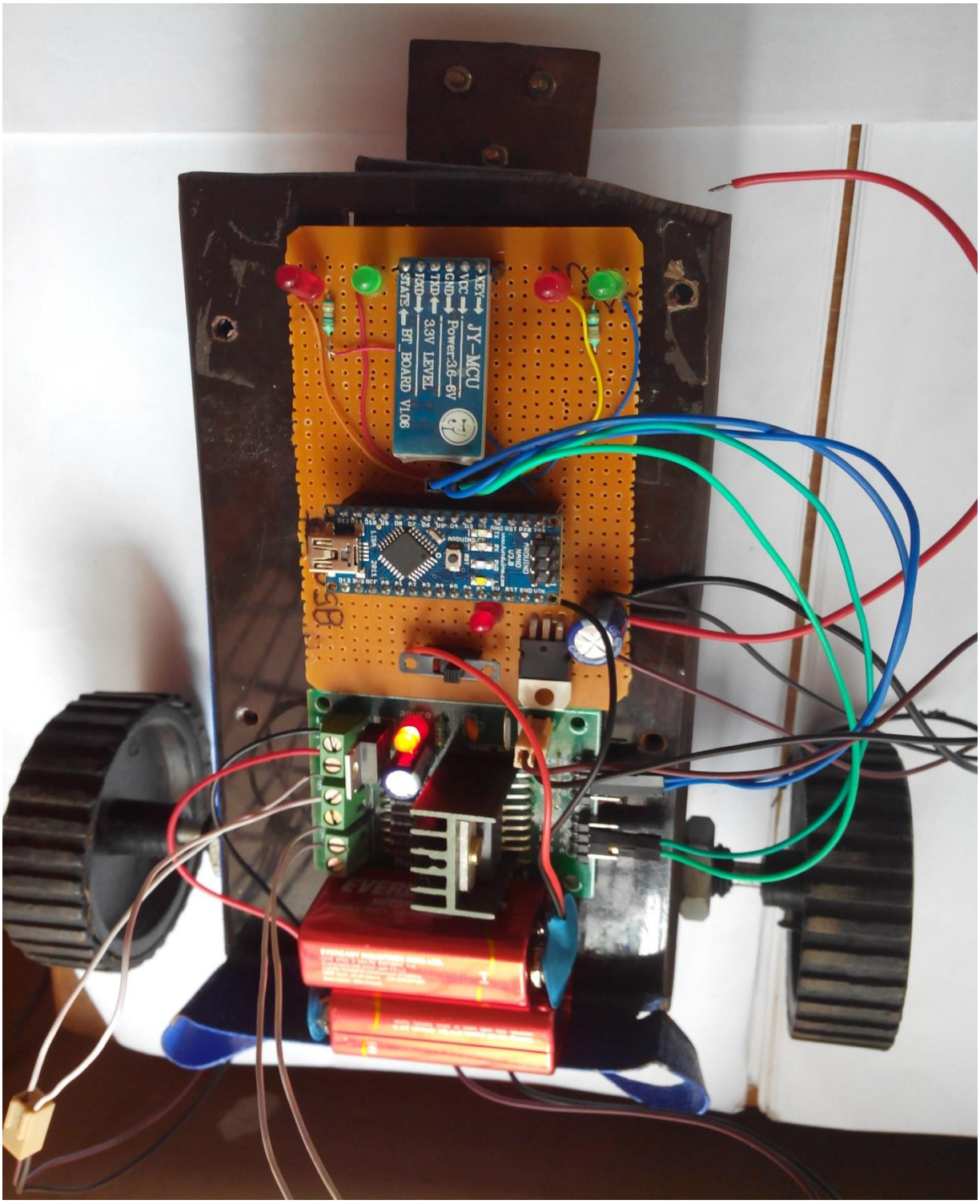
- *The Controller Board with all working components attached to the Robotic Chassis*



- *Robotic chassis with Controller board and Motor Driver Circuit*



- ***Complete Assembled board with all components in working condition***



CHAPTER-4

SOFTWARE

4. SOFTWARE

We targeted to choose a software platform and language that is easy to understand and program. So we chose simple high level language for our project. And we have used a simulation software to build and debug the entire model

4.1 ARDUINO IDE

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring 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, 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".

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnablecyclic executive program:

- 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 on and 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 at the bottom, to make it a valid C++ program. The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board.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.

Development

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested that the name "Arduino" be exclusive to the official product and not be used for derivative works without permission. The official policy document on the use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the "Arduino" name by using "-duino" name variants.

The board can be programmed from the Arduino software, which is available for different platforms such as Windows, Mac OSX and Linux. It is open source software, which is designed using a Java environment and is also based on processing and avr- gcc.

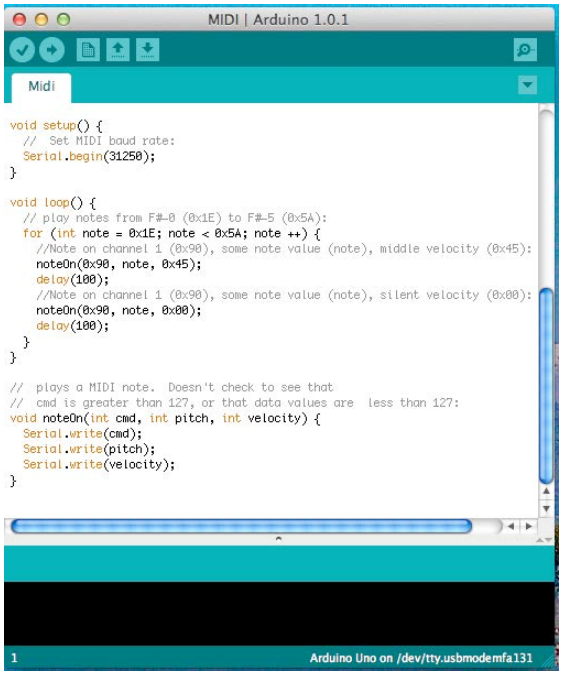
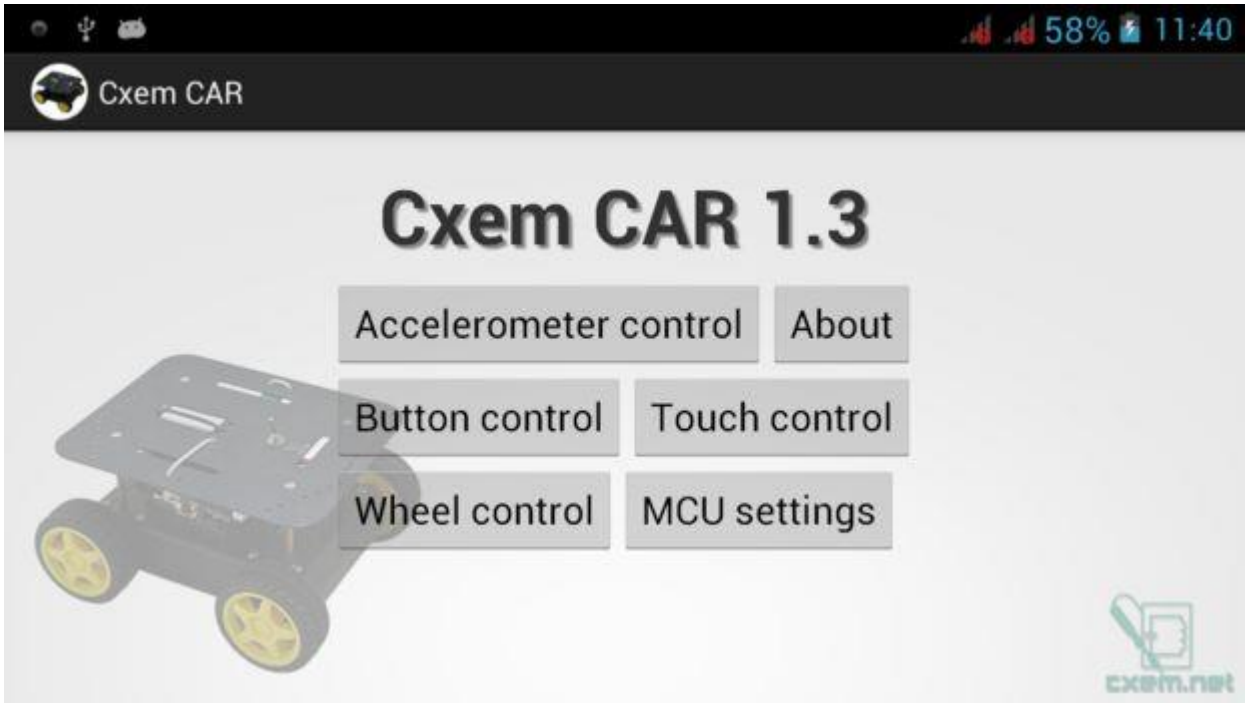


Figure 14. Arduino Software

The software allows users to write their code in C and upload to the board. The boot loader allows the uploading without the need of external hardware programmer. Thus the software is very easy to use and efficient.

4.2 ANDROID APPLICATION

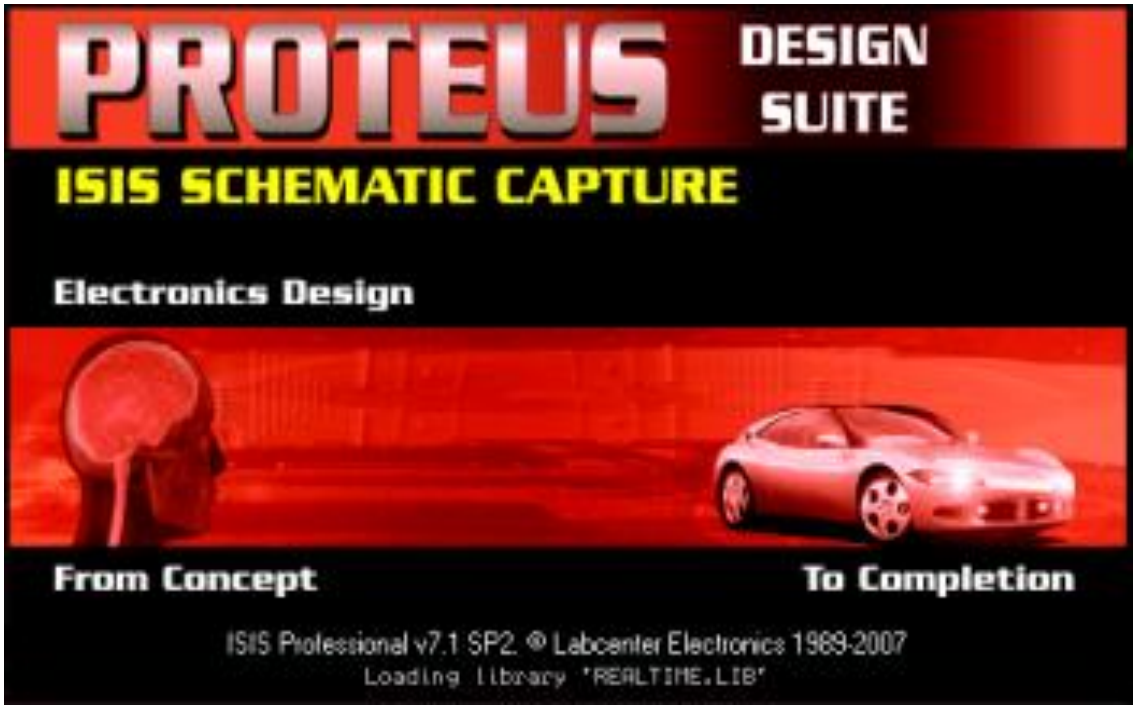
The application for Android was written in Eclipse IDE.. Android version on your device must be > 3.0. The application contains several activities. Main activity is a home screen with buttons running different operating modes and settings. There are 3 control modes Bluetooth-car: from accelerometer, screen buttons and touch-control.



To establish a connection with the Robotic Car's Bluetooth module, you must set MAC-address in the application settings. But first, you must configure the pair the devices on Android-device: open Settings -> Bluetooth and click "Search for devices". When the phone finds our Bluetooth-module, click them and enter password for pairing (usually "1234")
To know Bluetooth module MAC-address possible from any application, such as Bluetooth Terminal. To do this, click "Connect a device - Secure" and in the resulting window, click the button "Scan for devices". Software will be scans the Bluetooth devices and displays them MAC-address

4.3 PROTEUS PROFESSIONAL 7.8

Proteus PCB design combines the ISIS schematic capture and ARES PCB layout programs to provide a powerful, integrated and easy to use suite of tools for professional PCB Design. All Proteus PCB design products include an integrated shape based auto router and a basic SPICE simulation capability as standard.



System components

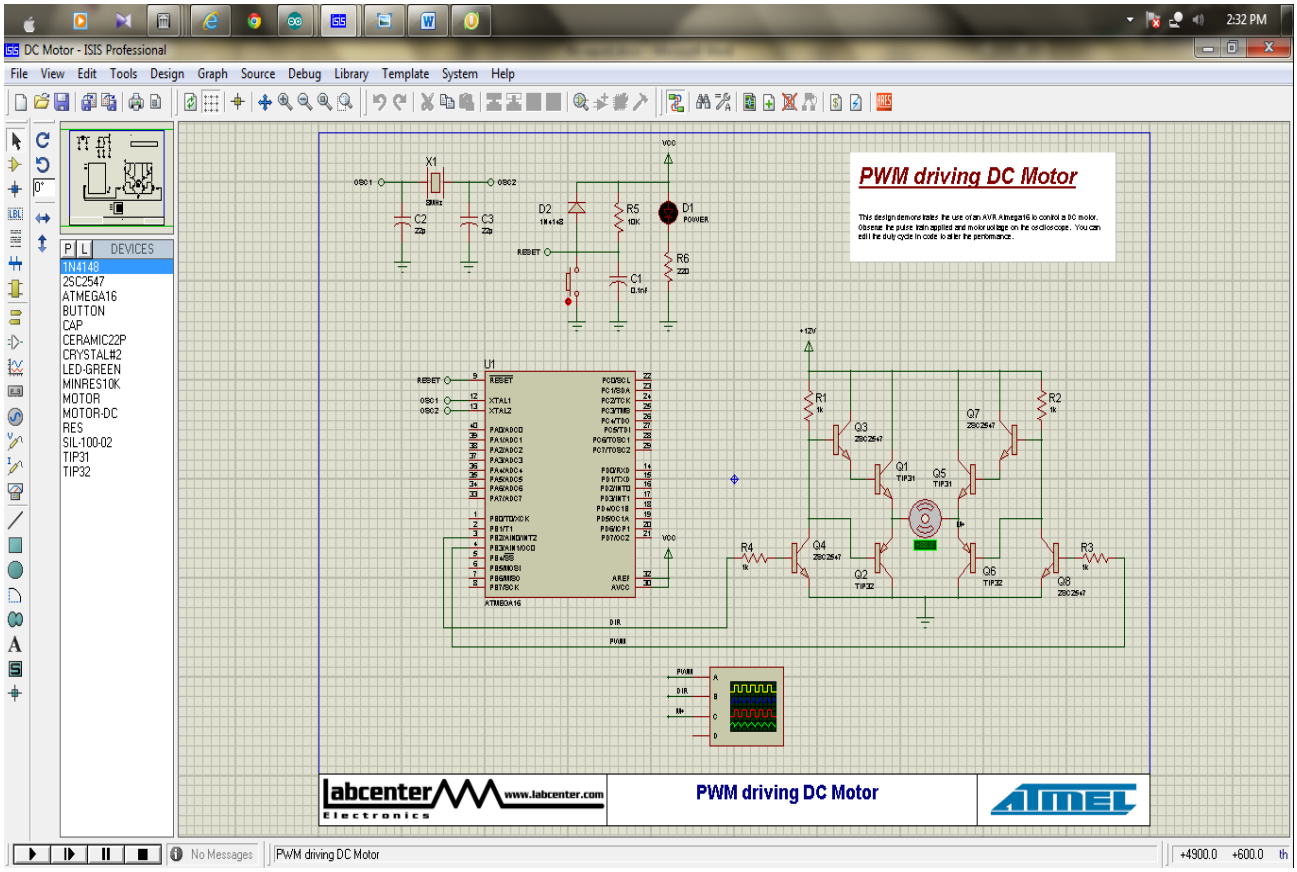
- **ISIS Schematic Capture** - a tool for entering designs.
- **PROSPICE Mixed mode SPICE simulation** - industry standard SPICE3F5 simulator combined with a digital simulator.
- **ARES PCB Layout** - PCB design system with automatic component placer, rip-up and retry auto-router and interactive design rule checking.
- **VSM** - Virtual System Modelling lets co-simulate embedded software for popular microcontrollers alongside hardware design.
- **System Benefits** Integrated package with common user interface and fully context sensitive help.

Proteus VSM for Arduino™ AVR

Proteus VSM for Arduino™ AVR® provides an integrated environment for development, testing and virtually prototyping your embedded system designs based around the popular Arduino™ platform. The unique nature of schematic based microcontroller simulation with Proteus facilitates rapid, flexible and parallel development of both the system hardware and the system firmware. This design synergy allows engineers to evolve their projects more quickly, empowering them with the flexibility to make hardware or firmware changes at will and reducing the time to market.

ISIS Schematic Capture

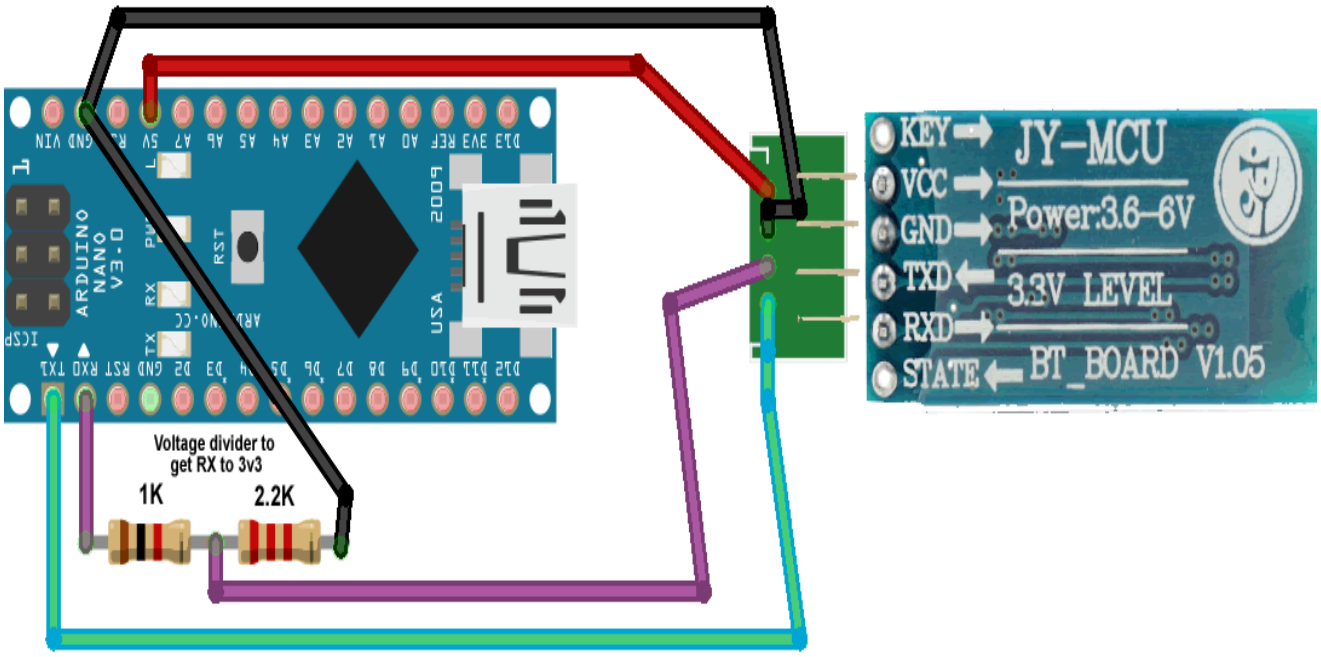
ISIS lies at the heart of the Proteus system, and is far more than just another schematics package. It combines a powerful design environment with the ability to define most aspects of the drawing appearance. Whether your requirement is the rapid entry of complex designs for simulation and PCB layout, or the creation of attractive schematics for publication, ISIS is the tool for the job.



CHAPTER-5

IMPLEMENTATION

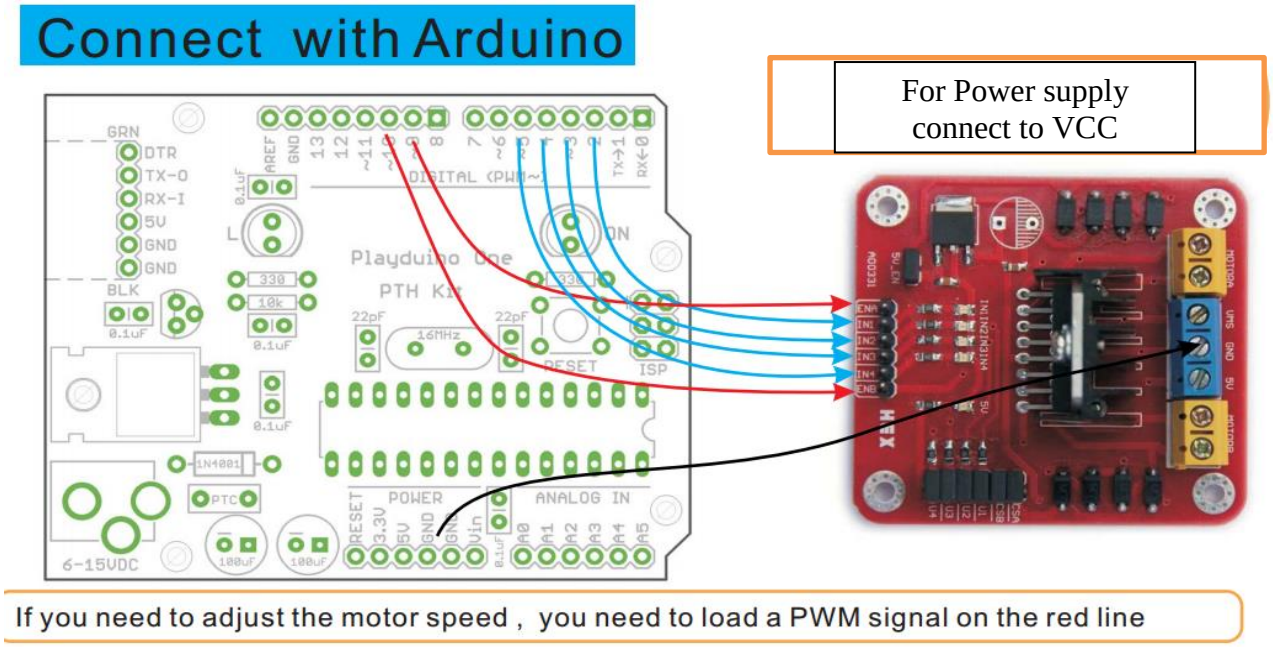
5.1 INTERFACING HC-05 WITH ARDUINO



Here the Arduino Nano is connected with Bluetooth module HC-05 for transmitting data which is received from android phone Bluetooth. The data is transmitted through UART serial communication means HC-05 to Arduino. In the circuit we used a jumper (in the scheme Jmp1), because with a connected Bluetooth module is impossible be load sketch to the Arduino. Between BT pin RX (2) and Arduino pin TX may require level shifter. So we used voltage divider to convert 5V from arduino to 3.3V

The TX pin of Arduino is connected to RX pin of bluetooth module. The RX of Arduino is connected to 1K resistor and then to TX of Bluetooth module. The 5V pin of arduino powers the Bluetooth by connecting it to VCC and GND to GND of arduino and Bluetooth.

5.2 INTERFACING ARDUINO WITH L298N DRIVER

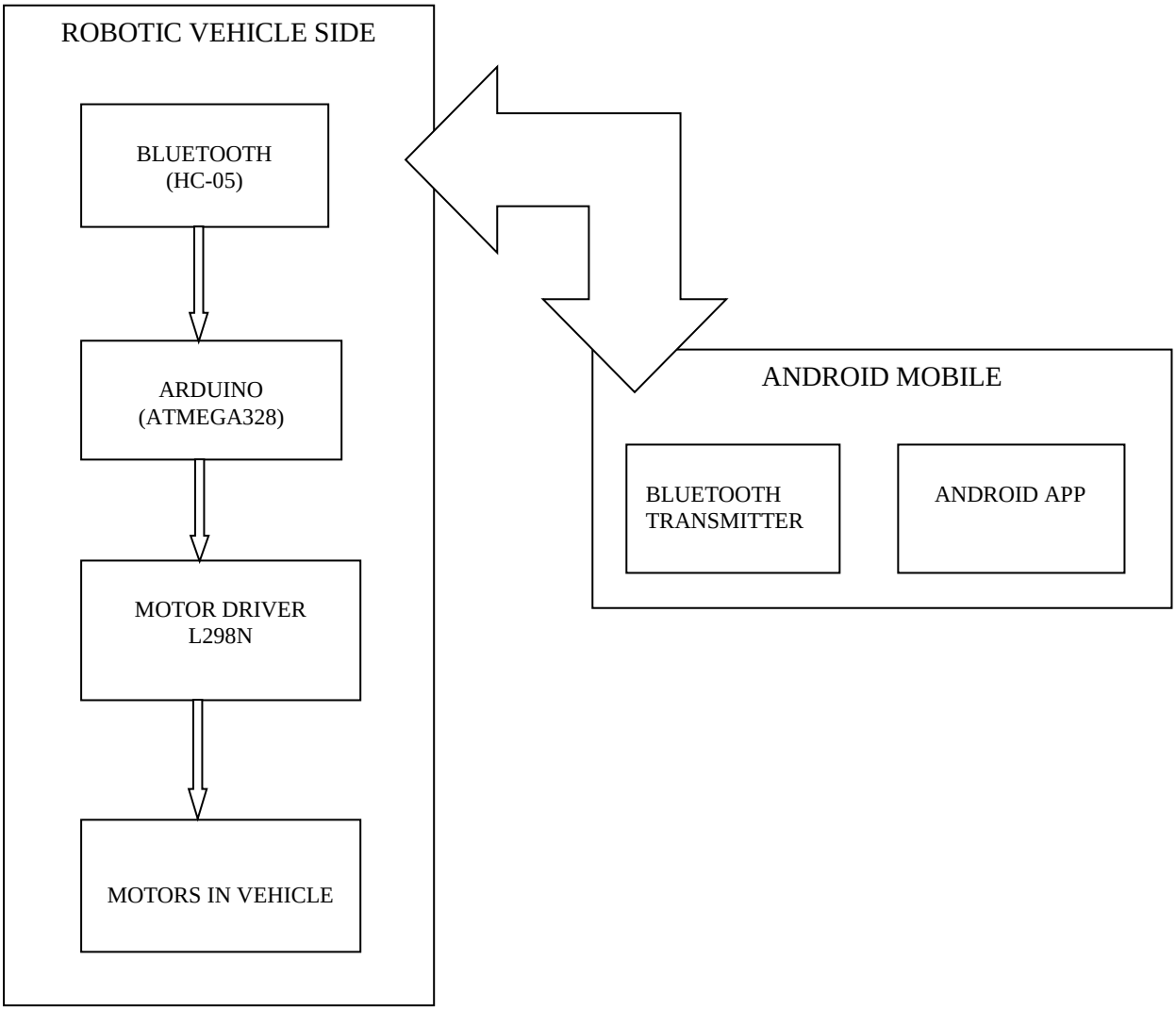


To drive the motors we are using L298N dual-full bridge motor driver. Form this driver we can drive 2 DC motors or single stepper motor independently.

The digital pins of arduino D2,D3,D4 and D5 are connected to input pins IN1,IN2,IN3 and IN4 of motor driver respectively to control the directions of motor 1 and 2. IN1 and IN2 drives motor1. IN3 and IN4 drives motor 2. The EN1 and EN2 pins connected to Arduino's pin 9 and 10, which supports PWM signals. These pins control the speed of the motors 1 and 2.

The motor driver can be powered upto 45V. By giving proper controlling inputs to IN1, IN2, IN3 and IN4 from arduino. The motor runs if IN1 and IN2 are HIGH and LOW respectively or vice-versa. It works on H-bridge. The H-bridge is explained above. When both are HIGH or LOW. The motor circuit behaves as a generator and stops. This motor driver controls each motors independently.

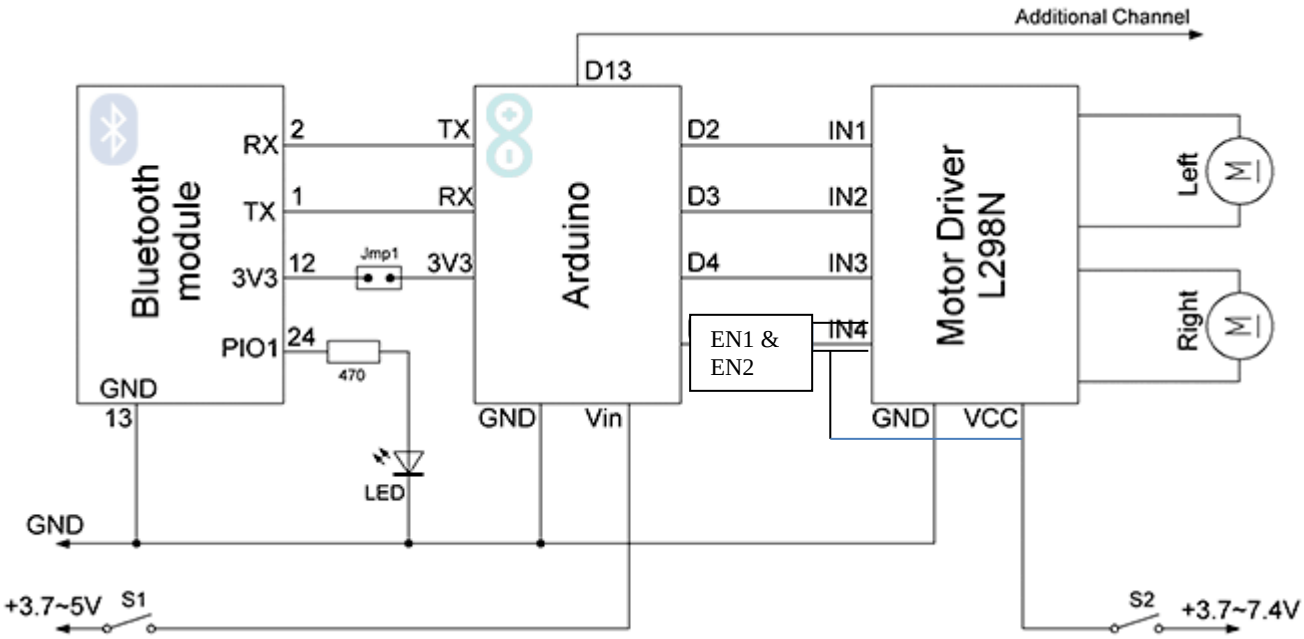
5.3 BLOCK DIAGRAM



The block diagram of project is shown above. The moments or gestures of mobile is sensed by the accelerator through android app. That data is sent through Bluetooth present inside the android phone. This Bluetooth acts as transmitter.

In the Robotic Vehicle side Bluetooth receiver HC-05 receives the data sent from mobile. This HC-05 is serially connected to Arduino. The data is transferred to Arduino serially. Now the control action takes place in the microcontroller Atmega 328. These control signals sent to motor driver L298N which drives the motor. These motors will be attached to the vehicle to run. The microcontroller controls the motors through motor driver.

WIRING DIAGRAM



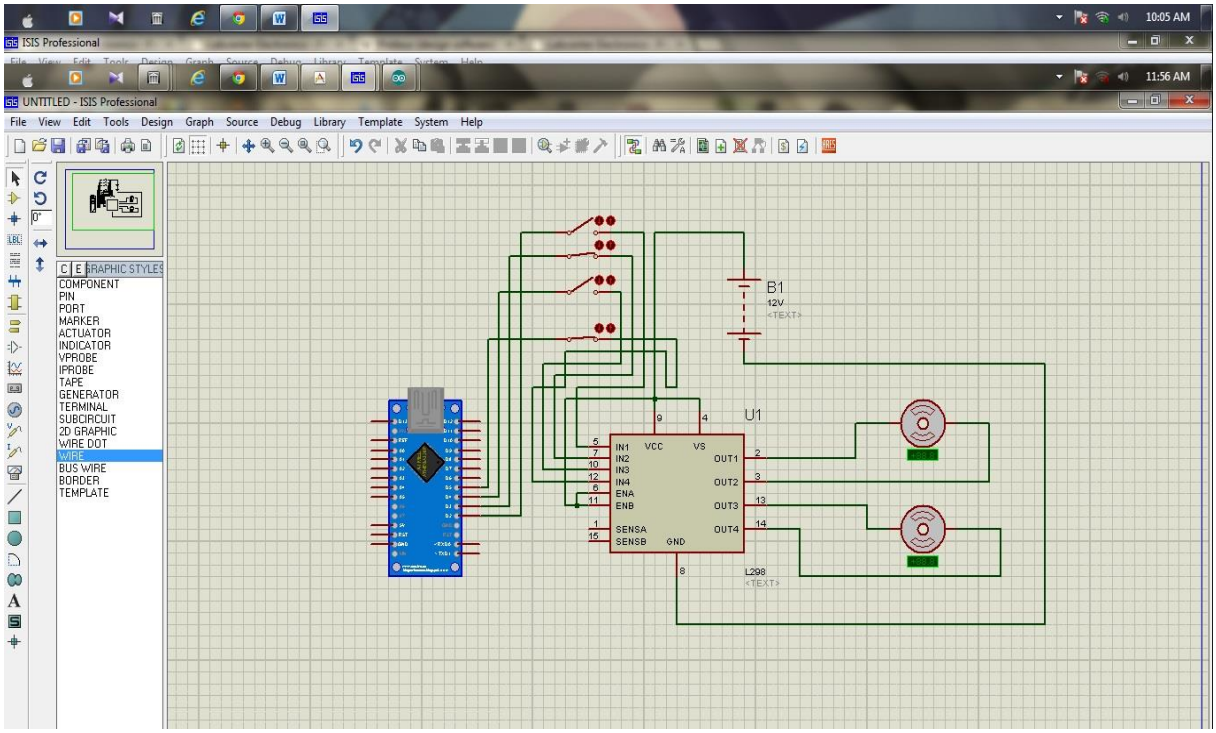
The wiring diagram is as follows. Thr arduino and motor driver are powered by separate 5V and 9V battries respectively. Bluetooth module is powered through arduino 5V supply. And for data transmission the voltage divider circuit is used from 5V to 3.3V using 1K and 2K resistors. The LED is connected to PIO1 pin to ground with 470ohm resistor. The RX of the HC-05 is connected to TX of the arduino and TX of the Bluetooth is connected to RX of the Arduino. The –ve terminals of both arduino and Bluetooth are connected to GND.

The connection of Motor driver L298N with Arduino is as follows. The Motor driver is powered through 9-12V battery. The other terminal of the motor driver is connected to ground. The Input pins IN1, IN2, IN3 and IN4 are connected to the digital pins D2, D3, D4 and D5. The motor 1 and 2 terminals on the motor driver are connected to the DC motors present in the robotic vehicle.

5.4 SIMULATION

We have simulated our entire circuit for designing and debugging the model. We have used PROTEUS PROFFESIONAL 7.9 Simulation tool. ISIS Schematic capture simulates the circuit

Simulation Sketch:



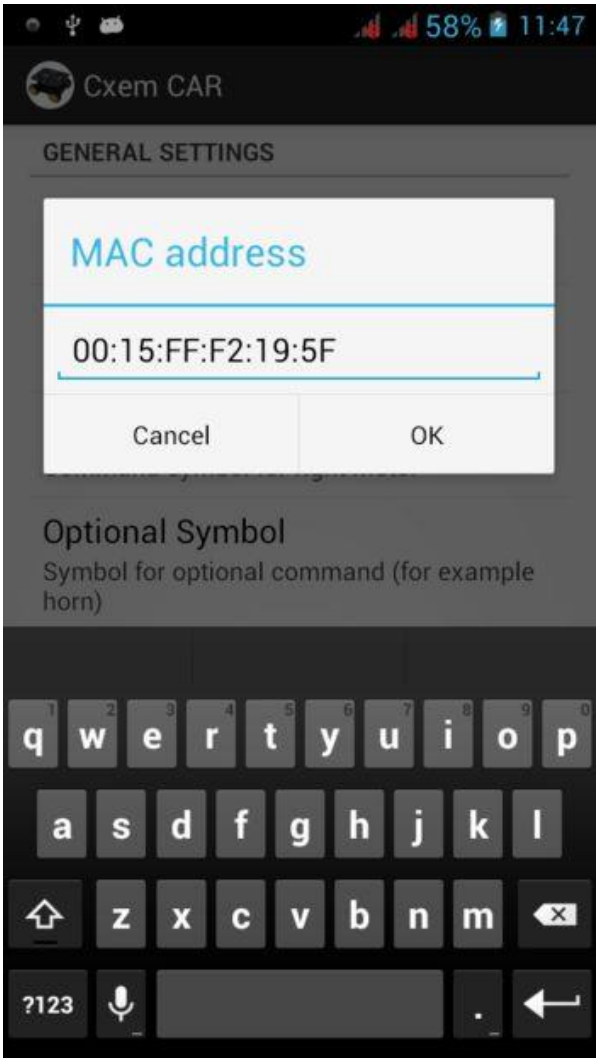
5.5 WORKING

All the connections are made as shown in the above figure. The circuit is powered by battery for both arduino and motor driver. And install the android app CXEM-CAR on android mobile.

Step-1: Establishing connection between android mobile and Bluetooth module (HC-05)

- Switch on the Bluetooth in android mobile
- Search for ‘new devices’

- Once HC-05 module is found pair with it and note the MAC-Address of the Bluetooth module (Default pairing code 0000 or 1234)



Step-2: Interfacing with Android application CXEM-CAR

- Open the application CXEM-CAR
- Go to 'Settings'
- Delete the default MAC-Address and enter the address of the Bluetooth module HC-05
- After pairing, the application shows that the 'HC-05 is connected' and the LED connected to the PIO1 of HC-05 pin glows, this ensures the connection between the app and the Bluetooth module

Step-3: Setting control for robotic vehicle

- Now the connection is established between the robotic vehicle and android app through Bluetooth
- In the settings set the control values if u want. Default settings are made to run the robotic vehicle in this app
- We can vary the PWM value from 0-255 or there is a option in the app to change the controls.

Step-4: Controlling

- Click on the 'Accelerometer Control' button on the home screen
- It shows 'Tilt your device' option and shows X, Y, Xpwm, Ypwm, Send: L, R' MotorL and MotorR with some varying values.
- Now by tilting the android device we are able to control the Robotic vehicle.
- By tilting Android mobile forward the robotic vehicle moves in forward direction.
- By tilting it backward the vehicle moves in backward direction.
- By tilting it towards left the vehicle moves to left direction.
- By tilting it towards right the vehicle moves in right direction.
- Thus we are able to control the vehicle in all the directions using the Android application.

CHAPTER-6

RESULT OF PROJECT

Case-1:

INTERFACING ARDUINO WITH BLUETOOTH MODULE AND ANDROID MOBILE:

- By interfacing arduino with bluetooth the data is transferred from bluetooth module HC-05 to Arduino through serial communication
- The Led attached with bluetooth module and arduino is controlled by using our android application
- This ensures the proper data transfer and interface between android mobile and arduino

Case-2:

INTERFACING ARDUINO WITH MOTOR DRIVER L298N AND MOTORS

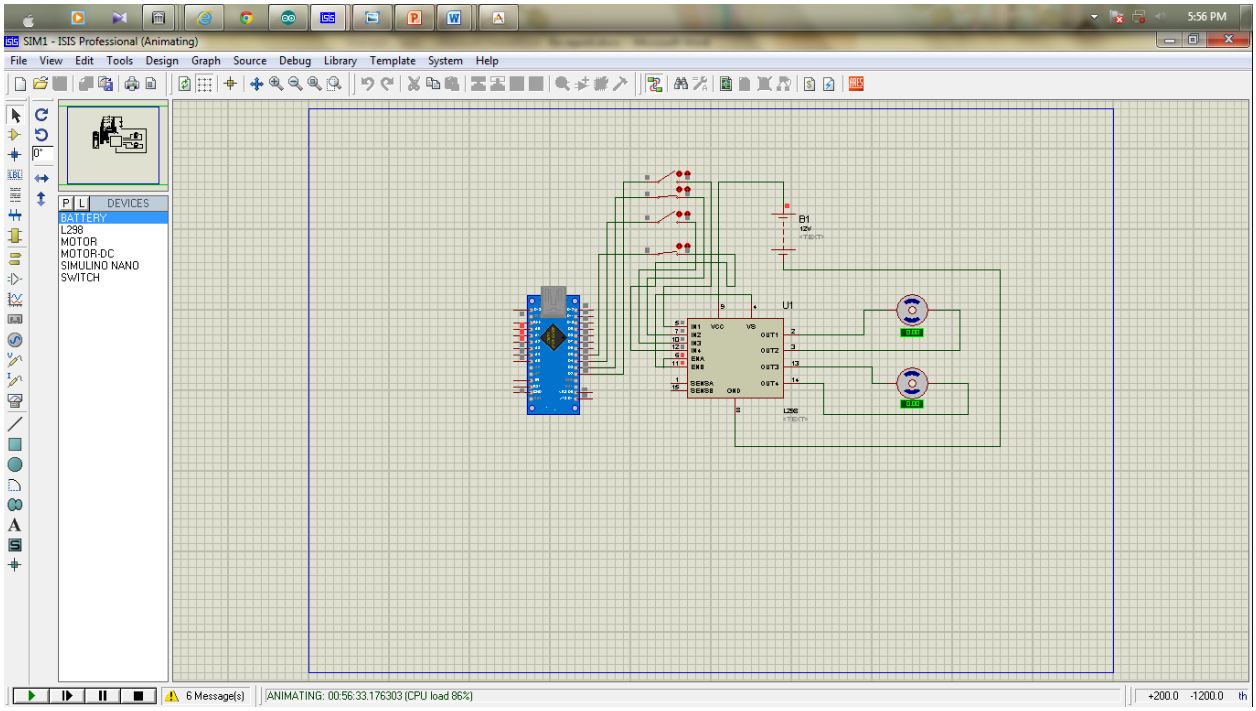
- By interfacing arduino with motor driver as shown above, to ensure a proper control of motors as we need
- The motors attached to the motor drives rotates forward, backward and with the use of two motors we are able to move the robot to left and right directions by using this H-Bridge motor driver circuit

Case-3:

SIMULATION RESULT

From the simulation of our circuit.

- When switch A is ON and switch B is OFF the motor A rotates in clockwise direction and same with respect to switch C and D for Motor B. Hence our vehicle moves forward.
- When switch A is OFF and switch B is ON the motor rotates in anticlockwise and same with respect to switch C and D for Motor B. Hence our vehicle moves backward.
- When both A and B are OFF the motor A stops rotating, with C in ON condition and D in OFF condition. The vehicle moves in right direction
- If its vice-versa of above case then the vehicle will be moving towards left direction
- If both A and B are OFF or ON and if its same with C and D. Both motor stops and the vehicle will not move



*Motor A is left Motor and B is right Motor in our case.

Case-4:

GESTURE CONTROL OF ROBOTIC VEHICLE

- The block diagram and wiring diagram shows the full details of our working model
- Now by tilting the android device we are able to control the Robotic vehicle.
- By tilting Android mobile forward the robotic vehicle moves in forward direction.
- By tilting it backward the vehicle moves in backward direction.
- By tilting it towards left the vehicle moves to left direction.
- By tilting it towards right the vehicle moves in right direction.
- Thus we are able to control the vehicle in all the directions using the Android application.

CHAPTER-7

APPLICATIONS AND FUTURE ENHANCEMENT

APPLICATIONS

This technique of controlling robotic vehicle through gestures can be implemented for many applications:

- Through the use of gesture recognition, remote control with the wave of a hand of various devices is possible.
- Gesture controlling is very helpful for handicapped and physically disabled people to achieve certain tasks, such as control electronic appliances.
- Gestures can be used to control interactions for entertainment purposes such as gaming to make the game player's experience more interactive or immersive.
- It can be used in various industries for picking various objects where human intervention is not desired.
- On a large scale, it can be used to develop robots with military applications. It can be used to target enemy without any human being crossing the territory.
- It is robust, sensitive and fast moving, hence can be applied in rescue operations.
- With tremendous smart phone in markets, it is bound to have many more applications in near future.

FUTURE ENHANCEMENT

- It provides for more development of applications based on android operating system. Such as. Application based on sensors etc. This opens door for wide range of possible similar applications Remote starter for car and Automation of household tasks.
- Enhanced for home automation using smartphone
- Advanced motion: i.e. robot arm controlled by servo motor
- Obstacle avoidance: Install proximity sensor; develop algorithms to steer around / back up when obstacles detected
- Vision: Use camera to transmit frames back to Android application for display to user
- Bluetooth too low-bandwidth; switch to Wi-Fi

CHAPTER-8

CONCLUSION

CONCLUSION

The development of this project is challenging yet quite enjoyable and comfortable. The outcome of the work is a simple robot which is controlled by a smartphone and its movement. Although the thesis projects very little about the robot's use in real world, but with the help of guidelines and the abundance of resources the outcome could be very beneficial for many people in the world. People with physical limitations such as handicapped people could use the feature from this thesis to compensate their abilities. Most importantly for home automation and as a new gaming trend with smartphones. By using this technology of controlling a robotic vehicle using smartphone, makes life easier. Can also be used as spy vehicle by implementing a camera on this type of vehicles.

This method of controlling robotic vehicles using smartphone enhances it to use for controlling purpose more than just as a telecommunication device.

BIBLIOGRAPHY

- W.T. Freeman, C.D. Weissman, Television control by hand gesture, Proceedings of the International Workshop on Automatic Face-and Gesture-Recognition, Zurich, Switzerland, June 1995, pp. 179-183.
- J.S. Kim, C.S. Lee, K.J. Song, B. Min, Z. Bien, Real-time hand gesture recognition for avatar motion control, Proceedings of HCI'97, February 1997, pp. 96-101.
- T. Starner, A. Pentland,"Visual recognition of american sign language using hidden Markov model", Proceedings of the International Workshop on Automatic Face-and Gesture Recognition, Zurich, Switzerland, June 1995, pp. 189-194.
- J. Yang, Y. Xu, C.S. Chen,"Human action learning via hidden markov model", IEEE Trans Systems, Man, Cybernet. 27 (1), 1997, pp. 34-44.
- T.S. Huang, A. Pentland,"Hand gesture modeling, analysis, and synthesis", Proceedings of the International Workshop on Automatic Face-and Gesture-Recognition, Zurich, Switzerland, June 1995, pp. 73-79.
- Jakub Segan,"Controlling computer with gloveless gesture", In Virtual Reality System'93, 1993, pp. 2-6.
- E. Hunter, J. Schlenzig, R. Jain,"Posture estimation in reduced-model gesture input systems, Proceedings of the International Workshop on Automatic Face-and Gesture-Recognition, June 1995, pp. 290-295.

- M.J. Swain, D.H. Ballard,"Color indexing", International Journal of Computer Vision 7 (1), 1991, pp. 11-32
- http://en.wikipedia.org/wiki/LM317
- http://en.wikipedia.org/wiki/78xx
- http://www.ti.com/product/l293D
- http://www.engineersgarage.com/electronic- components/l293d-motor-driver-ic
- https://www.futurlec.com/Atmel/ATMEGA8L.shtml
- "Beginning Android Application Development", John Wiley & Sons, Inc. By WEI MENG-LEE. × "Android Game Development", John Wiley & Sons, Inc. By ROBERT GREEN And MARIO ZECHNER.
- "Gesture Controlled Robot PPT" <http://seminarprojects.com/s/hand-gesture-controlled-robot-ppt>
- "Gesture Controlled Tank Toy User Guide" <http://www.slideshare.net/neeraj18290/wireless-gesture-controlled-tank-toy-transmitter>
- "Robotic Gesture Recognition (1997)" by Jochen Triesch and Christoph Von Der Malsburg <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.5427>
- "Hand Gesture Controlled Robot" by Bhosale Prasad S., Bunage Yogesh B. and Shinde Swapnil V.
- < http://www.robotplatform.com/howto/L293/motor_driver_1.html>
- < http://en.wikipedia.org/wiki/Gesture_interface>
- < http://www.wisegeek.com/what-is-a-gear-motor.htm>
- <http://www.scribd.com/doc/98400320/InTech-Real-Time-Robotic-Hand-Control-Using-Hand-Gestures>
- < http://en.wikipedia.org/wiki/DC_motor>

- <http://electronics.stackexchange.com/questions/18447/what-is-back-emf-counter-electromotive-force>
- < http://en.wikipedia.org/wiki/Robots>
- <www.alldatasheet.com>
- <www.google.com>
- <www.wikipedia.com>

APPENDIX A: Abbreviations

- AC : Alternative Current
- AFH : Adaptive Frequency Hopping feature
- CMOS : Complementary Metal-Oxide Semiconductor
- DC : Direct Current
- EEPROM : Electrically Erasable Programmable ROM
- EN : Enable
- FTDI : Future Technology Devices International
- GND : Ground
- GNU : GNU NOT UNIX
- GUIs : Graphical User Interface
- IC : Integrated Circuit
- ICSP : In Circuit Serial Programming
- IDE : Integrated Development Environment
- IN : Input
- ISIS : Image and Scanner Interface Specification
- LED : Light Emitting Diode
- LSB : Least Significant Bit
- MAC : Media Access Control address
- MISO : Master In Slave Out
- MOSI : Master Out Slave In

- PCB : Printed Circuit Board
- PMDC : Permanent Magnet DC motor
- PWM : Pulse Width Modulation
- RX : Receiver
- SCK : Serial Clock
- SPI : Serial Peripheral Interface
- SPICE : Simulation Program with Integrated Circuit Emphasis
- SRAM : Static RAM
- TTL : Transistor Transistor Logic
- TX : Transmitter
- UART : Universal Asynchronous Receiver and Transmitter
- USART : Universal Synchronous-Asynchronous Receiver and Transmitter
- USB : Universal Serial Board
- Vin : Power input
- VSM : Virtual System Modeling.
- Wi-Fi : Wireless-Fidility

APPENDIX B: Code

PROGRAM WRITTEN IN ARDUINO IDE

```
#include "EEPROM.h"
#define D1 2          // direction of motor rotation 1
#define M1 3          // PWM left motor
#define D2 4          // direction of motor rotation 2
#define M2 5          // PWM right motor
#define HORN 13       // additional channel 1
//#define autoOFF 2500  // milliseconds after which the robot stops when the connection

#define cmdL 'L'      // UART-command for left motor
#define cmdR 'R'      // UART-command for right motor
#define cmdH 'H'      // UART-command for additional channel (for example Horn)
#define cmdF 'F'      // UART-command for EEPROM operation
#define cmdr 'r'      // UART-command for EEPROM operation (read)
```

```
#define cmdw 'w'      // UART-command for EEPROM operation (write)


char incomingByte;    // incoming data
char L_Data[4];       // array data for left motor
byte L_index = 0;     // index of array L
char R_Data[4];       // array data for right motor
byte R_index = 0;     // index of array R
char H_Data[1];       // array data for additional channel
byte H_index = 0;     // index of array H
char F_Data[8];       // array data for  EEPROM
byte F_index = 0;     // index of array F
char command;         // command


unsigned long currentTime, lastTimeCommand, autoOFF;


void setup() {
  Serial.begin(9600);       // initialization UART
  pinMode(HORN, OUTPUT);    // additional channel
  pinMode(D1, OUTPUT);      // output for motor rotation
  pinMode(D2, OUTPUT);      // output for motor rotation
  /*EEPROM.write(0,255);
  EEPROM.write(1,255);
  EEPROM.write(2,255);
  EEPROM.write(3,255);*/
  timer_init();             // initialization software timer
```

```
}

void timer_init() {
  uint8_t sw_autoOFF = EEPROM.read(0);   // read EEPROM "is activated or not stopping the car
        when losing connection"
  if(sw_autoOFF == '1'){                // if activated
    char var_Data[3];
    var_Data[0] = EEPROM.read(1);
    var_Data[1] = EEPROM.read(2);
    var_Data[2] = EEPROM.read(3);
    autoOFF = atoi(var_Data)*100;       // variable autoOFF ms
  }
  else if(sw_autoOFF == '0'){
    autoOFF = 999999;
  }
  else if(sw_autoOFF == 255){
    autoOFF = 2500;                     // if the EEPROM is blank, dafault value is 2.5 sec
  }
  currentTime = millis();               // read the time elapsed since application start
}

void loop() {
  if (Serial.available() > 0) {         // if received UART data
    incomingByte = Serial.read();       // raed byte
    if(incomingByte == cmdL) {          // if received data for left motor L
```

```
  command = cmdL;                  // current command
  memset(L_Data,0,sizeof(L_Data));   // clear array
  L_index = 0;                     // resetting array index
 }
 else if(incomingByte == cmdR) {      // if received data for left motor R
  command = cmdR;
  memset(R_Data,0,sizeof(R_Data));
  R_index = 0;
 }
 else if(incomingByte == cmdH) {      // if received data for additional channel
  command = cmdH;
  memset(H_Data,0,sizeof(H_Data));
  H_index = 0;
 }
 else if(incomingByte == cmdF) {      // if received data for EEPROM op
  command = cmdF;
  memset(F_Data,0,sizeof(F_Data));
  F_index = 0;
 }
 else if(incomingByte == '\r') command = 'e';   // end of line
 else if(incomingByte == '\t') command = 't';   // end of line for EEPROM op

 if(command == cmdL && incomingByte != cmdL){
  L_Data[L_index] = incomingByte;            // store each byte in the array
  L_index++;                                 // increment array index
```

```
  }
  else if(command == cmdR && incomingByte != cmdR){
    R_Data[R_index] = incomingByte;
    R_index++;
  }
  else if(command == cmdH && incomingByte != cmdH){
    H_Data[H_index] = incomingByte;
    H_index++;
  }
  else if(command == cmdF && incomingByte != cmdF){
    F_Data[F_index] = incomingByte;
    F_index++;
  }
  else if(command == 'e'){                    // if we take the line end
    Control4WD(atoi(L_Data),atoi(R_Data),atoi(H_Data));
    delay(10);
  }
  else if(command == 't'){                    // if we take the EEPROM line end
    Flash_Op(F_Data[0],F_Data[1],F_Data[2],F_Data[3],F_Data[4]);
  }
  lastTimeCommand = millis();                 // read the time elapsed since application start
 }
 if(millis() >= (lastTimeCommand + autoOFF)){     // compare the current timer with variable
      lastTimeCommand + autoOFF
  Control4WD(0,0,0);                          // stop the car
```

```
  }
}

void Control4WD(int mLeft, int mRight, uint8_t Horn){

  bool directionL, directionR;      // direction of motor rotation L298N
  byte valueL, valueR;              // PWM M1, M2 (0-255)

  if(mLeft > 0){
    valueL = mLeft;
    directionL = 0;
  }
  else if(mLeft < 0){
    valueL = 255 - abs(mLeft);
    directionL = 1;
  }
  else {
    directionL = 0;
    valueL = 0;
  }

  if(mRight > 0){
    valueR = mRight;
    directionR = 0;
  }
```

```
  else if(mRight < 0){
    valueR = 255 - abs(mRight);
    directionR = 1;
  }
  else {
    directionR = 0;
    valueR = 0;
  }

  analogWrite(M1, valueL);            // set speed for left motor
  analogWrite(M2, valueR);            // set speed for right motor
  digitalWrite(D1, directionL);       // set direction of left motor rotation
  digitalWrite(D2, directionR);       // set direction of right motor rotation

  digitalWrite(HORN, Horn);           // additional channel
}

void Flash_Op(char FCMD, uint8_t z1, uint8_t z2, uint8_t z3, uint8_t z4){

  if(FCMD == cmdr){           // if EEPROM data read command
    Serial.print("FData:");       // send EEPROM data
    Serial.write(EEPROM.read(0));     // read value from the memory with 0 address and print it to
        UART
    Serial.write(EEPROM.read(1));
    Serial.write(EEPROM.read(2));
```

```
    Serial.write(EEPROM.read(3));
    Serial.print("\r\n");        // mark the end of the transmission of data EEPROM
   }
   else if(FCMD == cmdw){         // if EEPROM data write command
    EEPROM.write(0,z1);            // z1 record to a memory with 0 address
    EEPROM.write(1,z2);
    EEPROM.write(2,z3);
    EEPROM.write(3,z4);
    timer_init();            // reinitialize the timer
    Serial.print("FWOK\r\n");       // send a message that the data is successfully written to
        EEPROM
   }
  }
```