ECE3003

MICROCONTROLLERS AND ITS APPLICATIONS

J Component

A project report titled

AUTOMATED CAR PARKING SYSTEM

By

17BEC1002	DEBARPAN MAZUMDER
17BEC1003	ABHINAV BHATTACHARYA
17BEC1020	SOUBHIK MAZUMDAR
17BLC1109	ARKA KUMAR

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Submitted to

PROF.V.PRAKASH

March 2019

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering (SENSE)

"J" COMPONENT REVIEW-II REPORT

COURSE CODE / NAME	ECE3003 – MICROCONTROLLERS & ITS APPLICATIONS
PROGRAM / YEAR / SEM	B.Tech (ECE/ECM)/II Year / Fall 2018-19
DATE OF REVIEW	27/03/2019
J TITLE	AUTOMATED CAR PARKING SYSTEM

	REGISTER NO.	**NAME**
TEAM MEMBERS DETAILS	17BEC1002	DEBARPAN MAZUMDER
	17BEC1003	ABHINAV BHATTACHARYA
	17BEC1020	SOUBHIK MAZUMDAR
	17BLC1109	ARKA KUMAR

EVALUATION ITEMS	**Yes (√) / No (x)**
The project has achieved the objective set for this point?	
Level of Knowledge Gained While Completing the Project was satisfactory?	
Are the students having clear idea on their proposed and have they acquired to carry forward it?	
Are the contribution made by the individuals towards attaining objective of the project was satisfactory?	
Are the submitted report and presentation made by each team member was satisfactory?	

COURSE INCHARGE NAME	**Prof. V. PRAKASH**	**MARKS**	
REVIEWER'S NAME & SIGN			

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report entitled "**AUTOMATED CAR PARKING SYSTEM"** is a bonafide work of **DEBARPAN MAZUMDER(17BEC1002), ABHINAV BHATTACHARYA (17BEC1003),SOUBHIK MAZUMDAR (17BEC1020), ARKA KUMAR(17BLC1109)** carried out the "J"-Project work under my supervision and guidance for ECE 3003 Microcontrollers and its Applications.

PROF.V.PRAKASH

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Our team members were quite new to the field of Microcontrollers before the start of this project. Previously we were introduced to the study of sensors in our second semester. We had some basic ideas about the sensors like IR, Ultrasonic, LDR, etc. While thinking upon making something out of a microcontroller and an ultrasonic sensor we stumbled upon this idea of making an automated car parking system.

We have tried to design a device that could help driver from accidents when parking a car. The driver knows the range of the vehicle and any object behind. Moreover, the informing device should help to break the machine down automatically. This device uses ultrasound as range finder, microcontroller AT89S52 as processes calculation and displays result in LCD screen. The model proposed is a basic and cheap demonstration of the model we wanted to design.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof.V.Prakash,** School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A,** Dean of the Schools of Electrical Engineering (SELECT) and Electronics Engineering (SENSE), VIT University Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Programme Chair **Dr. Vetrivelan. P.(for B.Tech-ECE)** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

OBJECTIVE OF THE PROJECT:

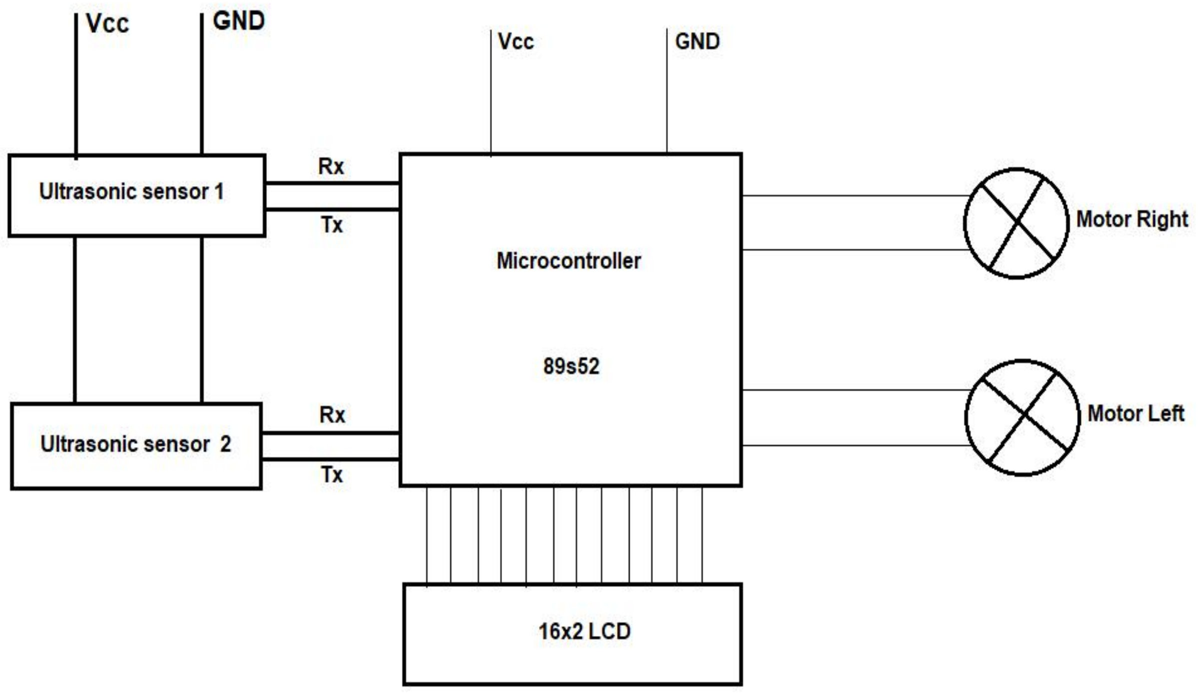
- The objective of the project is to study, identify and apply the properties of ultrasonic sensor and use them to design an automated car parking system with the help of 89S52 Microcontroller.
- The system controls steering, acceleration and braking automatically, based on the parking zone and location information gathered from the ultrasonic sensor, to achieve parallel parking and garage parking.

COMPONENTS REQUIRED:

S. No	Component Name	Specification	Quantity	Cost (in Rs.)
1.	AT89S52 Microcontroller IC	24 PU 1625A	1	80
2.	HCSR04 Ultrasonic Module	DC5V,15mA,40 HzMax Range: 4m,Min Range: 4cm	1	150
3.	LCD Display	16x2 inch	1	100
4.	Jumper Wires		2 set	120
5.	PCB Board	16cmx12cm	1	110
6.	Crystal Oscillator	11.059MHz	1	8
7.	L298N H Bridge Motor driver	V(mot)=46V,Vcc =5V,Imax=4A	1	250
8.	Capacitors	33uF	2	5
9.	8051 Burner	-	1	800
10.	Motor	3kgcm,5.9v	2	300
11.	Chasis	20cmx20cm	1	100

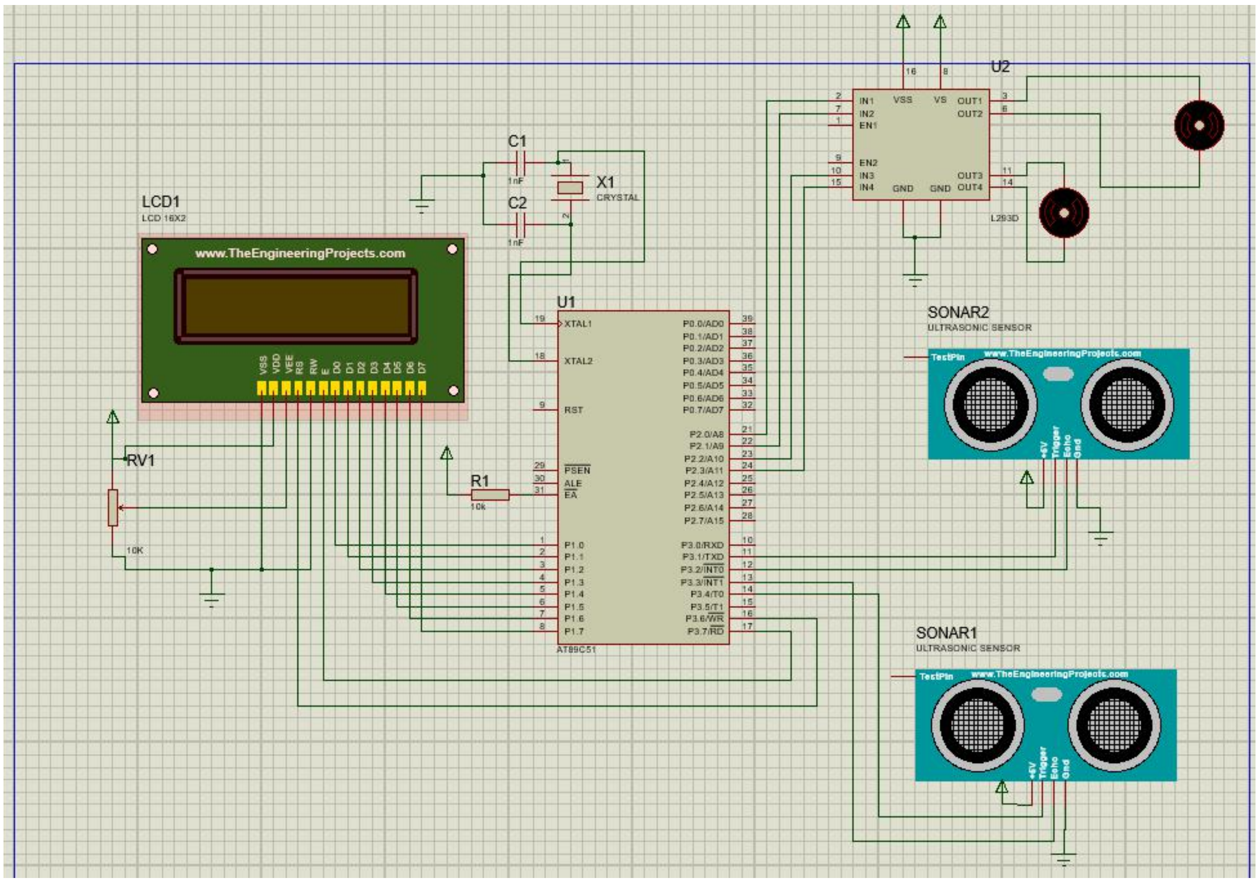
Overall cost of the Project: **2,023** **(in Rupees)**

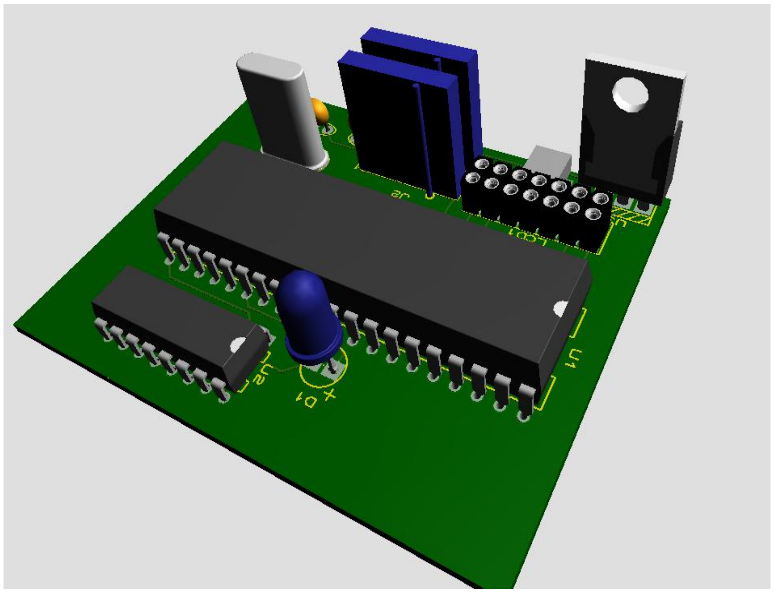
BLOCK DIAGRAM:



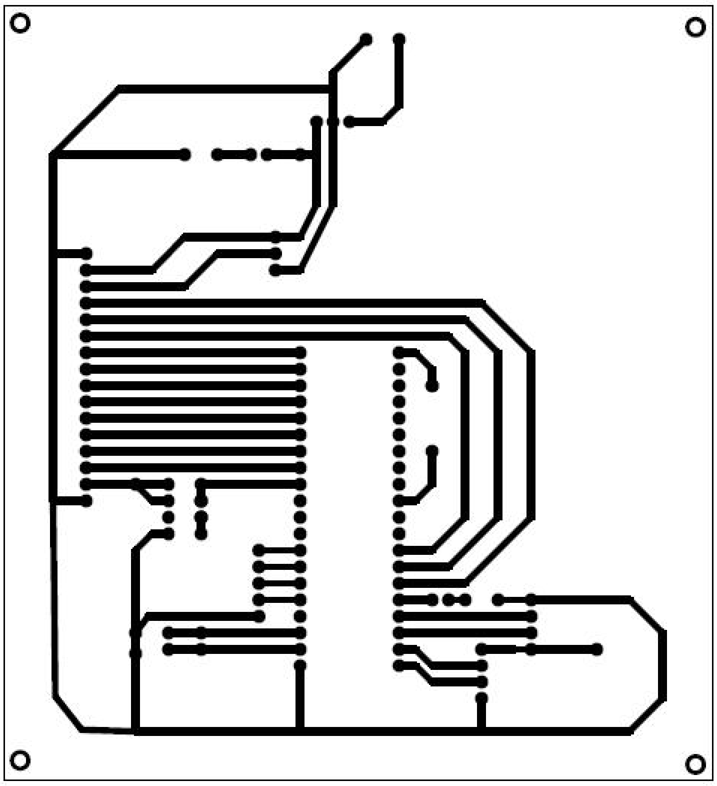
Two Ultrasonic sensors HCSR04 are connected together and are individually interfaced with the AT89S52 microcontroller and connected to it via a transmission and reception path. Two motors are interfaced namely the left and right motors. The 16x2 LCD display is interfaced with the microcontroller.

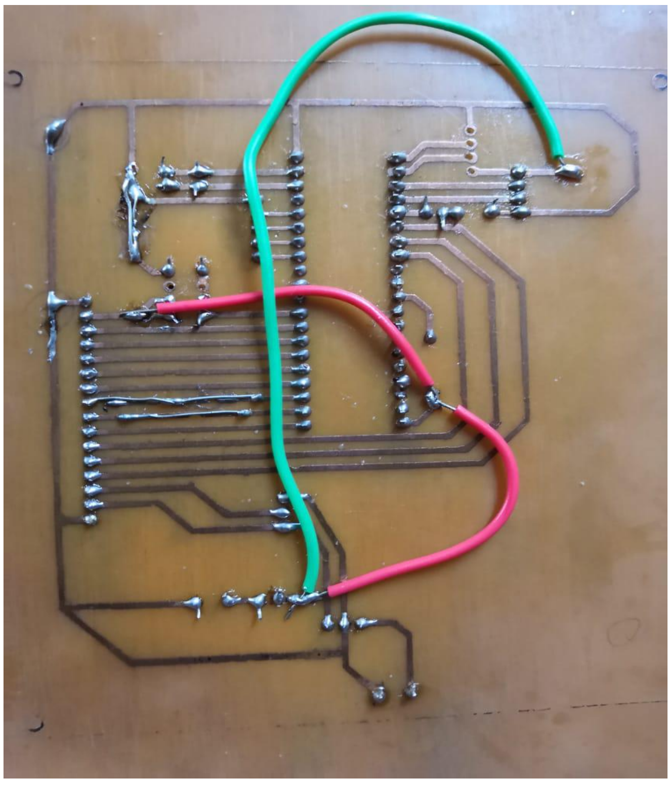
SCHEMATIC DIAGRAM





PCB LAYOUT





PROJECT DESCRIPTION

The objective of the project is to study, identify and apply the properties of ultrasonic sensor and use them to design an automated car parking system with the help of AT89S52 Microcontroller with the basic idea of 8051 as reference.

The system controls steering, acceleration and braking automatically, based on the parking zone and location information gathered from the ultrasonic sensor, to achieve parallel parking and garage parking.

DETAILED DESCRIPTION:

The concept basically relies upon Ultrasonic Waves using an Ultrasonic Sensor. The HCSR04 sensor has two capsules-transmitting and receiving. The transmitting capsule emits an ultrasonic wave that hits the obstacle, returns back and is received by the receiving capsule. A 10us pulse is sent to the trigger pin. Upon Excitation it releases an Ultrasonic sound wave from its transmitting capsule and the echo pin is simultaneously pulled to a high logic level, and remains high until the sound wave is received back at the receiving capsule, after which it is again pulled low. The State of the echo pin is constantly monitored using our 8051 microcontroller and the width of the pulse generated by echo pin is recorded using the built in timer register. The distance of the obstacle is measured by the formula:

$$D = (V*T)/2$$

Where,

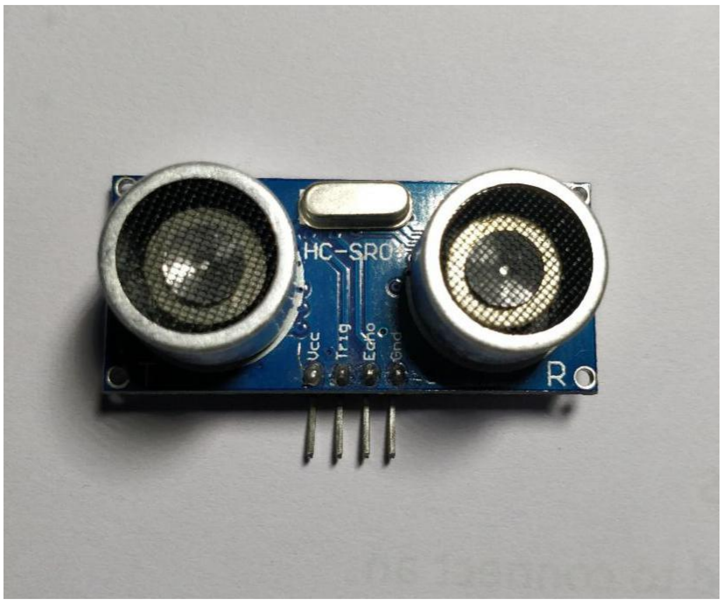
V=343m/s and T is the value taken from TH and TL register

The above formula has been encrypted inside the program. The vehicle designed exhibits reverse gear parking only. Since there are obstacles in the parking lot, the car senses it and stops at an appropriate distance. Here, we have assumed that distance to be 10cm.Whenever it comes within 10 cm of the obstacle, it stops and we consider it to be parked. Initially we had proposed the usage of two Ultrasonic sensors. Since we were working on microcontrollers for the first time, we worked with only one to reduce the complexity. The automation part of the project has been achieved successfully .We went through the logic for the working of Ultrasonic sensor and took reference of the codes of embedded C that uploads or burns to 89S52 for e.g. TMOD usage in embedded C and counter programming. Assimilation of the individual codes were done after interfacing was done separately. The algorithm worked as follows:

1. The system is switched on and the sensor constantly starts updating the distance once it is activated via emission and reception of pulse.
2. The pulses received are converted to distance and checked whether it is within 10cm or not. When the distance is more than 10cm, the LCD, Motor Driver and Ultrasonic work in parallel.
3. If less than 10cm the system is stopped or interrupt is used.
4. The motor driver is connected to the motor inside the Chassis and sends the pulses accordingly.
5. The pulses are only for backward motion.
6. The vehicle aligns itself on the information it gathers from its surroundings.

COMPONENT DESCRIPTION:

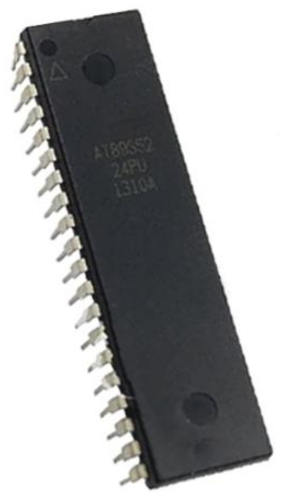
1. Ultrasonic Module HC-SR04

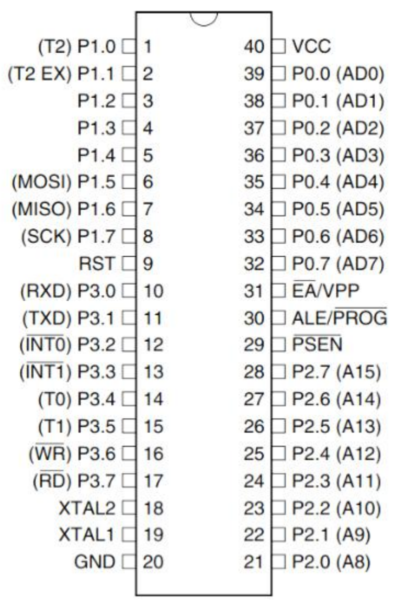


Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR system.

- The HC-SR04 module has ultrasonic transmitter, receiver and control circuit on a single board.
- The module has only 4 pins, Vcc, Gnd, Trig and Echo.
- When a pulse of 10µsec or more is given to the Trig pin, 8 pulses of 40 kHz are generated. After this, the Echo pin is made high by the control circuit in the module.
- Echo pin remains high till it gets echo signal of the transmitted pulses back.
- The time for which the echo pin remains high, i.e. the width of the Echo pin gives the time taken for generated ultrasonic sound to travel towards the object and return.
- Using this time and the speed of sound in air, we can find the distance of the object using a simple formula for distance using speed and time.

2. AT89S52 Microcontroller IC





Pin Number	**Pin Name**	**Description**
1	P1.0 (T2)	Timer/Counter or 0th GPIO pin of PORT 1
2	P1.1 (T2.EX)	Timer/Counter/External Counter or 1st GPIO pin of PORT 1
3	P1.2	2nd GPIO pin of PORT 1
4	P1.3	3rd GPIO pin of PORT 1
5	P1.4	4th GPIO pin of PORT 1
6	P1.5 (MOSI)	MOSI for in System Programming or 5th GPIO pin of PORT 1
7	P1.6 (MISO)	MISO for in System Programming or 6th GPIO pin of PORT 1

8	P1.7 (SCK)	SCK for in System Programming or 7th GPIO pin of PORT 1
9	RST	Making this pin high will reset the Microcontroller
10	P3.0 (RXD)	RXD Serial Input or 0th GPIO pin of PORT 3
11	P3.1 (TXD)	TXD Serial Output or 1st GPIO pin of PORT 3
12	P3.2 (INT0’)	External Interrupt 0 or 2nd GPIO pin of PORT 3
13	P3.3 (INT1’)	External Interrupt 1 or 3rd GPIO pin of PORT 3
14	P3.4 (T0)	Timer 0 or 4th GPIO pin of PORT 3
15	P3.5 (T1)	Timer 1 or 5th GPIO pin of PORT 3
16	P3.6 (WR’)	Memory Write or 6th GPIO pin of PORT 3
17	P3.7 (RD’)	Memory Read or 7th GPIO pin of PORT 3
18	XTAL2	External Oscillator Output
19	XTAL1	External Oscillator Input
20	GND	Ground pin of MCU
21	P2.0(A8)	0th GPIO pin of PORT 2
22	P2.1 (A9)	1st GPIO pin of PORT 2
23	P2.2 (A10)	2nd GPIO pin of PORT 2

24	P2.3 (A11)	3rd GPIO pin of PORT 2
25	P2.4 (A12)	4th GPIO pin of PORT 2
26	P2.5 (A13)	5th GPIO pin of PORT 2
27	P2.6 (A14)	6th GPIO pin of PORT 2
28	P2.7 (A15)	7th GPIO pin of PORT 2
29	PSEN’	Program store Enable used to read external program memory
30	ALE / PROG’	Address Latch Enable / Program Pulse Input
31	EA’ / VPP	External Access Enable / Programming enable Voltage
32	P0.7 (AD7)	Address / Data pin 7 or 7th GPIO pin of PORT 0
33	P0.6 (AD6)	Address / Data pin 6 or 6th GPIO pin of PORT 0
34	P0.5 (AD5)	Address / Data pin 5 or 5th GPIO pin of PORT 0
35	P0.4 (AD4)	Address / Data pin 4 or 4th GPIO pin of PORT 0
36	P0.3 (AD3)	Address / Data pin 3 or 3rd GPIO pin of PORT 0
37	P0.2 (AD2)	Address / Data pin 2 or 2nd GPIO pin of PORT 0

38	P0.1 (AD1)	Address / Data pin 1 or 1st GPIO pin of PORT 0
39	P0.0 (AD0)	Address / Data pin 0 or 0th GPIO pin of PORT 0
40	VCC	Positive pin of MCU (+5V)

Brief explanation of AT89S52 Microcontroller

- The **AT89S52** comes from the popular 8051 family of Atmel Microcontrollers. It is an 8-bit CMOS microcontroller with 8K as Flash memory and 256 bytes of RAM. Since it is similar to the trust worthy 8051 architecture these microcontrollers are as per industry standard. It has 32 I/O pins comprising of three 16-bit timers, external interrupts, full-duplex serial port, on-chip oscillator and clock circuitry.
- The Microcontroller also has Operating mode, Idle Mode and Power down mode which makes it suitable for battery operated applications. Few considerable drawback of the microcontroller is that it does not have in-built ADC and does not support SPI or I2C protocols. However you can utilise external modules for the same.

Programming AT89S52 Microcontroller

Atmel microcontroller can be programmed with different software's that is available in the market. Keil uVision is one of the most used platforms for the same.

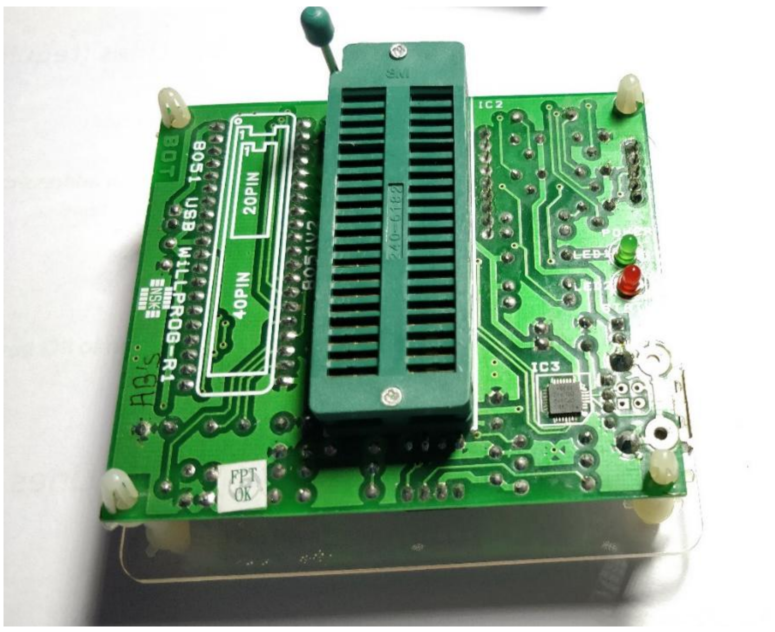
In order to program the Atmel microcontroller we will need an IDE (Integrated Development Environment), where the programming takes place. A compiler, where our program gets converted into MCU readable form called HEX files. An IPE (Integrated Programming Environment), which is used to dump our hex file into our MCUs.

IDE: Keil uVision IDE

Programming Hardware:

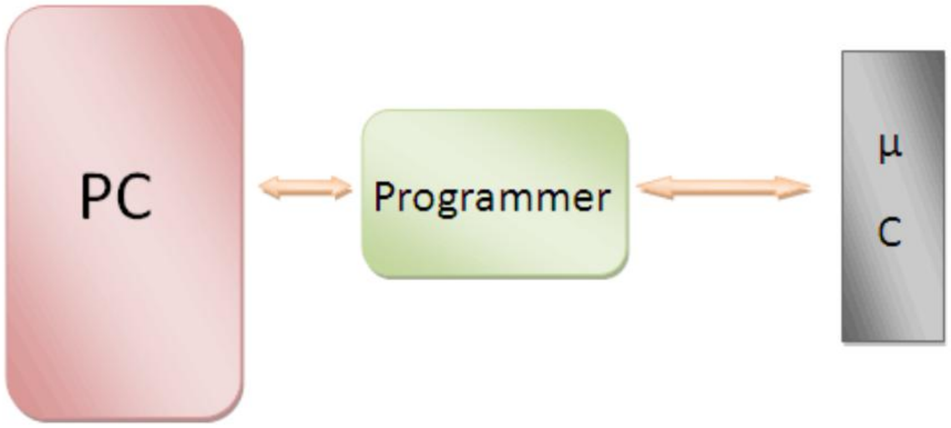
Programmer: 8051 USB WillPROG-R1

By simulating program on software like Proteus Design Suite 8.3 before trying it on hardware will save a lot of time.



3. 8051 USB WillPROG-R1

A microcontroller programmer or microcontroller burner is a hardware device accompanied with software which is used to transfer the machine language code to the microcontroller/EEPROM from the PC. The compiler converts the code written in languages like assembly, C, java etc to machine language code (which is understandable by the machines/microcontrollers) and stores it in a hex file. A microcontroller programmer acts as an interface between the PC and the target controller. The API/software of the programmer reads data from the hex file stored on the PC and feeds it into the controller's memory. The target controller on which the program needs to be burned is placed on the programmer using a ZIP socket. The software transfers the data from the PC to the hardware using serial, parallel or USB port.



Block Diagram of a Microcontroller Programmer

Depending on the way it interacts with PC, there are three types of microcontroller programmers:

Parallel Programmer uses the parallel port of the PC. They are low cost programmer but not widely used.

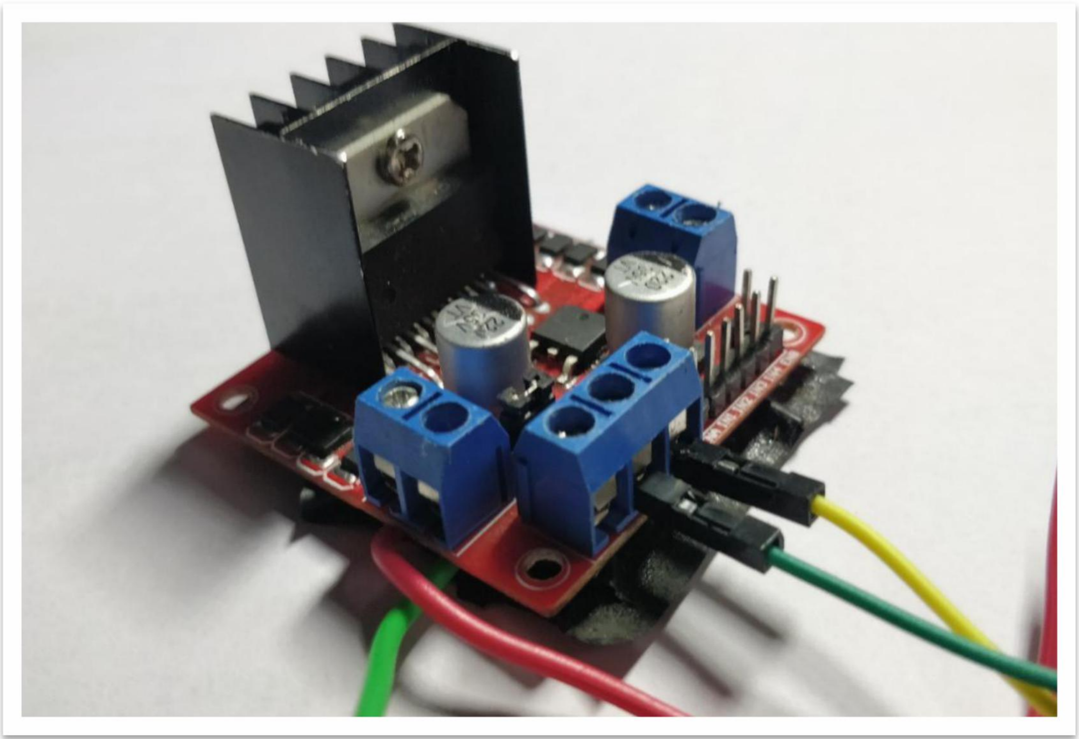
Serial Programmers uses the serial port to interact with PC via RS232 protocols. They are more popular among hobbyist working on PC. However both the serial and parallel programmers will become obsolete in near future. The major reason being unavailability of parallel and serial ports in the PCs & Laptops in the coming years.

USB Programmer uses the USB interface to transfer the data from PC. The main advantage of the USB burner is that they are powered from the PC itself and there is no need of any additional supply. The USB programmers have already become popular and will soon replace the serial and parallel programmer.

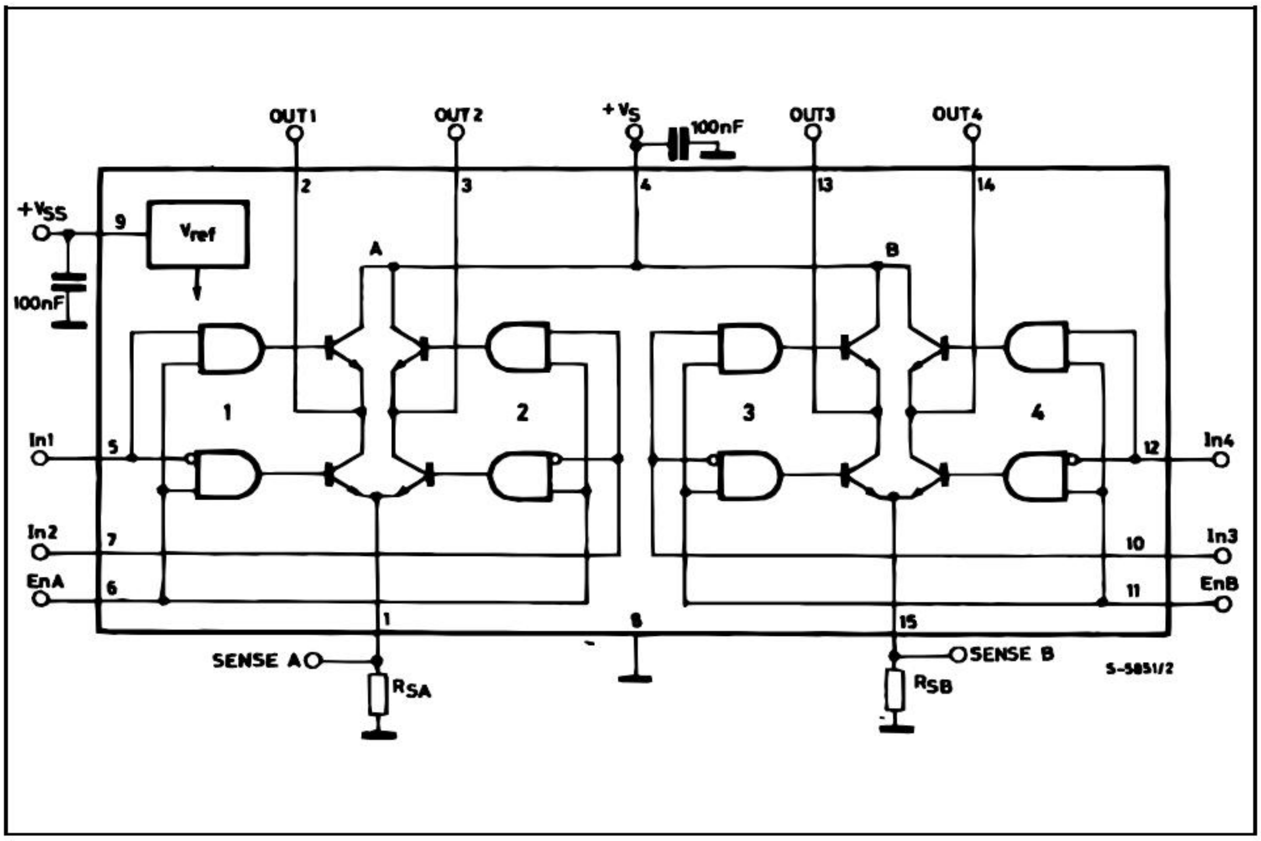
The programmer generally contains a microcontroller which is preprogrammed to take data from the PC and program the target controller. The programmer burns the target controller using any of the protocols like SPI, parallel interfacing, I2C/TWI or CAN. The speed of burning depends on the way of programmer is interfaced with PC and the protocols used to burn the target controller.

The conventional method to burn a controller is to take it out the circuit, place it on burner and then dump the hex file into the controller using the API. In order to remove this problem of removing the controller from the circuit every time it needs to be programmed, the controllers have now been upgraded with In System Programmer (ISP) feature. This allows burning/programming a controller without removing the controller from the circuit it is used in. The latest controllers are coming with the feature like boot loader memory which allows self-burning capabilities, i.e. such microcontroller controller does not need any additional programmer hardware. They need only an API to transfer the program to the target controller. This API can also be incorporated in the compiler and hence the compiler can directly burn the target controllers.

4. L298 N H-Bridge

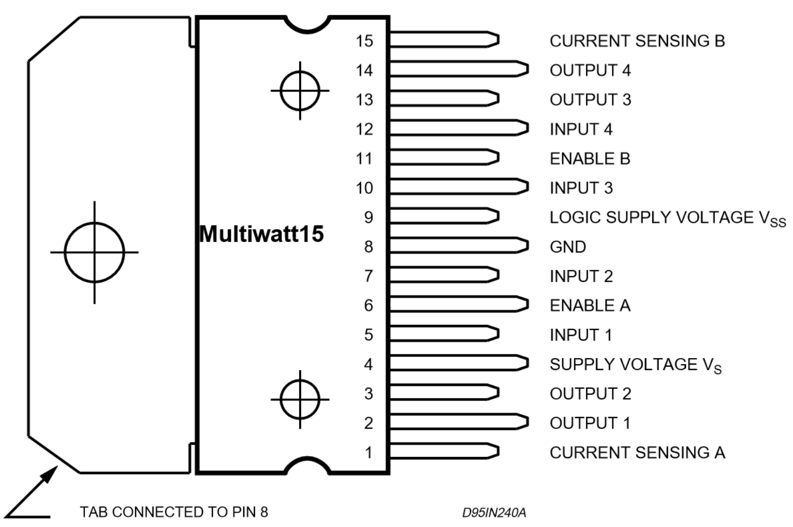


BLOCK DIAGRAM:



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C



Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (Please refer to the block diagram)

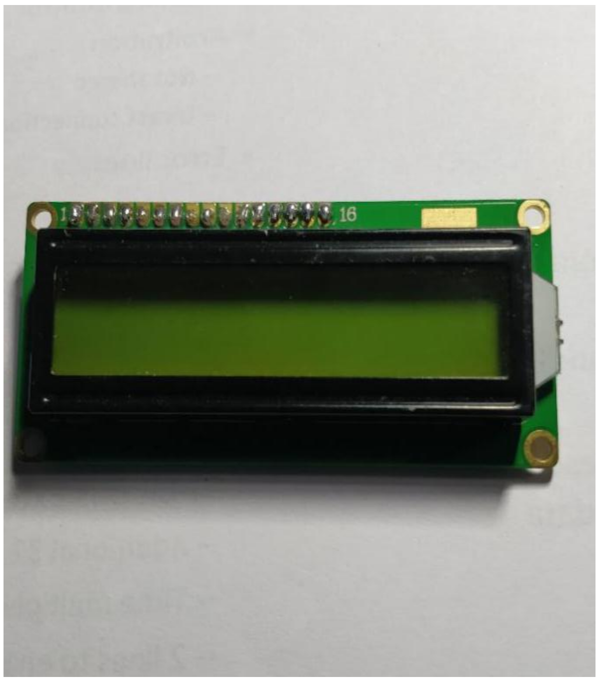
MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	μA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	μA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	μA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	μA
$V_{CEsat\,(H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\,(L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

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 de-signed 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.

5. LCD Display



An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16×2 LCD display is a very basic module commonly used in DIYs and circuits. The 16×2 translates a display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×7 pixel matrix

Pin No.	**Function**	**Name**
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; the best way is to use a variable resistor such as a potentiometer. The output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and	Vo / VEE

	backwards to adjust the LCD contrast.	
4	Selects command register when low, and data register when high	RS (Register Select)
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given; Extra voltage push is required to execute the instruction and EN(enable) signal is used for this purpose. Usually, we make it en=0 and when we want to execute the instruction we make it high en=1 for some milliseconds. After this we again make it ground that is, en=0.	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight VCC (5V)	Led+
16	Backlight Ground (0V)	Led-

RS (Register select)

A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

Command Register: The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. Processing for commands happens in the command register.

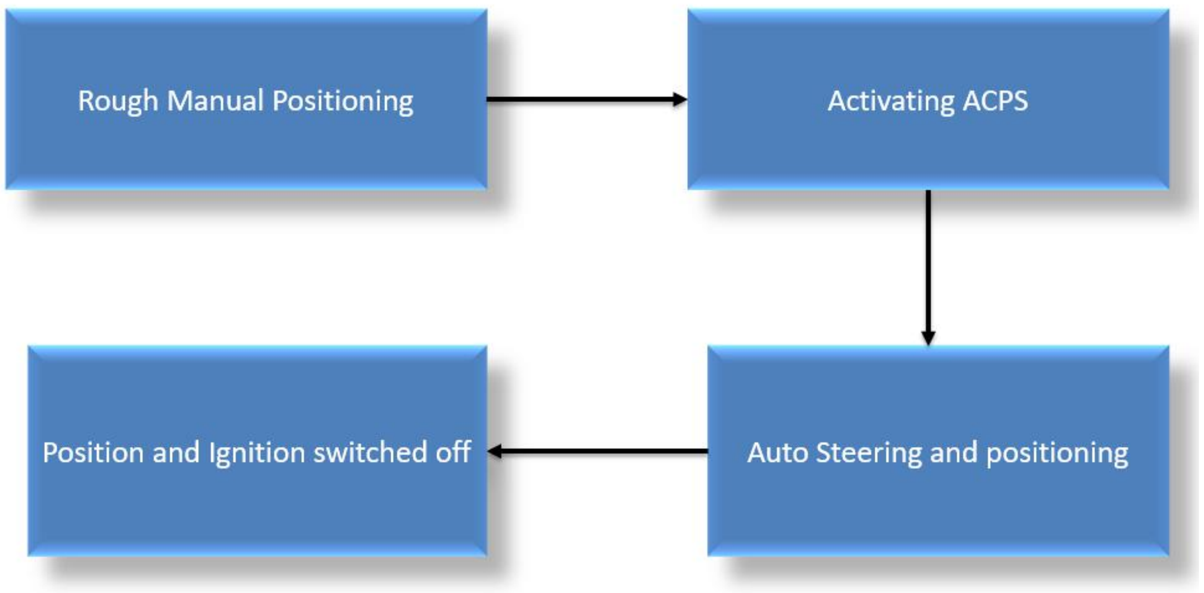
Data Register: The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected.

Important command codes for LCD

Sr.No.	**Hex Code**	**Command to LCD instruction Register**
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off

10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix

Flow chart:


Rough Manual Positioning
Activating ACPS
Auto Steering and positioning
Position and Ignition switched off

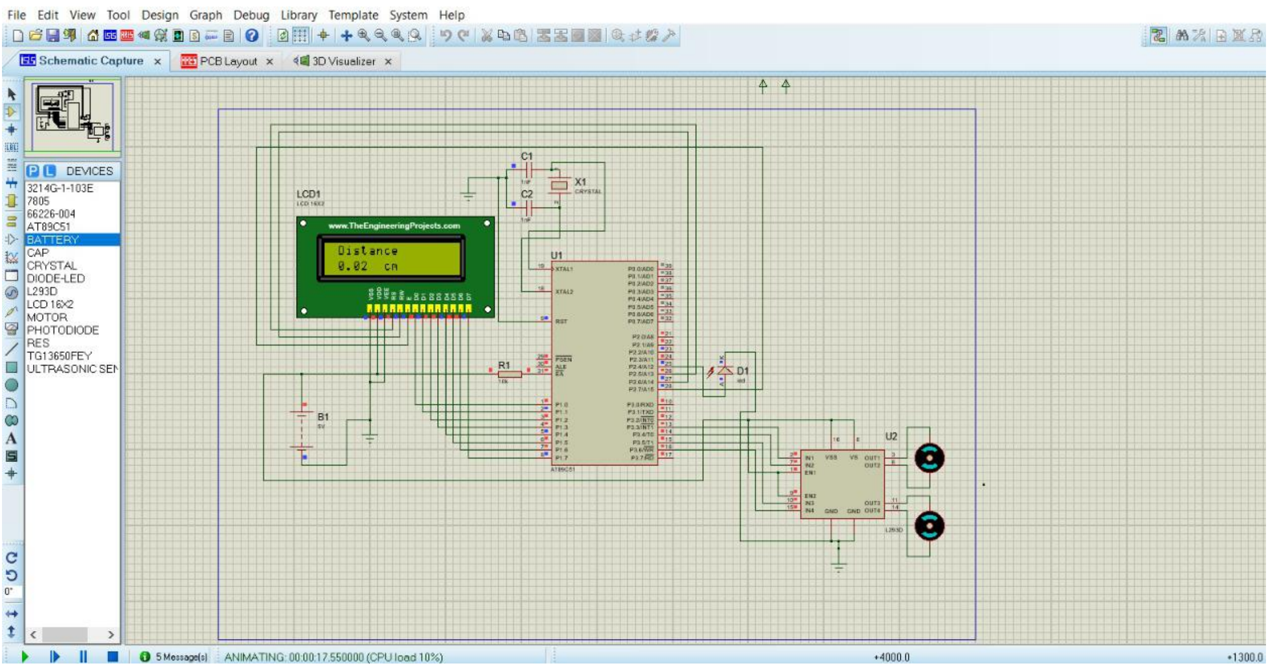
PROGRAM:

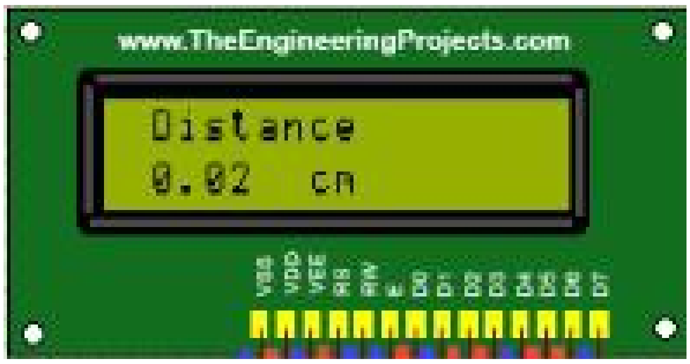
```
#include<reg52.h>

#include <stdio.h>

#include <LCD_8_bit.h>

#include <math.h>

#include <intrins.h>


#define sound_velocity 34300      /* sound velocity in cm per second */

#define period_in_us pow(10,-6)

#define Clock_period 1.085*period_in_us                  /* period for clock cycle of 8051*/


sbit Trigger_pin=P2^2;               /* Trigger pin */

sbit Echo_pin=P2^3;

sbit LED=P2^4;/* Echo pin */

sbit in1=P3^3;

sbit in2=P3^4;

sbit in3=P3^5;

sbit in4=P3^6;


void Delay_us()

{

                TL0=0xF5;

                TH0=0xFF;

                TR0=1;

                while (TF0==0);

                TR0=0;

                TF0=0;

}
```

```
void init_timer(){
        TMOD=0x01;                                                                          /*initialize
Timer*/
        TF0=0;
        TR0 = 0;
}

void send_trigger_pulse(){
         Trigger_pin= 1;             /* pull trigger pin HIGH */
   Delay_us();                /* provide 10uS Delay*/
   Trigger_pin = 0;           /* pull trigger pin LOW*/
}

void forward()
{
        in1=1;
        in2=0;
        in3=1;
        in4=0;
}
void stop()
{
        in1=in2=in3=in4=0;
}

void main()
{
        float distance_measurement, value;
        unsigned char distance_in_cm[10];

        LCD_Init();                                                                          /*
Initialize 16x2 LCD */

        LCD_String_xy(1,1,"Distance");

        init_timer();                                                              /* Initialize
Timer*/

        forward();


while(1)

{

                send_trigger_pulse();                              /* send trigger pulse of 10us
*/


                while(!Echo_pin);          /* Waiting for Echo */

   TR0 = 1;                  /* Timer Starts */

   while(Echo_pin && !TF0);    /* Waiting for Echo goes LOW */

   TR0 = 0;                  /* Stop the timer */


                /* calculate distance using timer */

                value = Clock_period * sound_velocity;

                distance_measurement = (TL0|(TH0<<8));
                                        /* read timer register for time count */

                distance_measurement = (distance_measurement*value)/2.0;  /* find
distance(in cm) */


                sprintf(distance_in_cm, "%.2f", distance_measurement);

                LCD_String_xy(2,1,distance_in_cm);
                                                /* show distance on 16x2 LCD */

                LCD_String("  cm  ");

        if(distance_measurement>10)
```

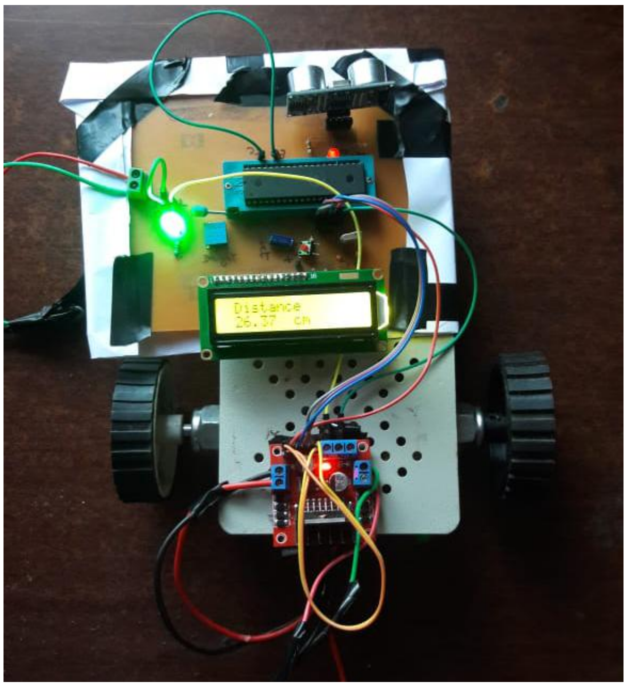
```
                forward();
        else
        {
                stop();
                LED=1;
                delay(10000);
        }

                delay(100);
}
}
```

SIMULATION OUTPUT:





IMPLEMENTATION OUTPUT:



INFERENCE:

- In first test ultrasonic sensor was used to measure the distance of object and display it on LCD.
- In second test motor driver was switching motors whenever there is an object in vicinity with sensor in range of 10cm.
- Three attempts were made to make the custom PCB for the schematic, the discontinuities were solved using soldering.
- Final assembled product when switched ON moves backward, displays distance between obstacle and the prototype and stops whenever distance is less than 10 cm.

CONCEPTS LEARNED:

1. The process of programming a standalone chip.
2. Interfacing of Ultrasonic sensor using 8051 microcontroller.
3. C programming for 8051 microcontroller.
4. Motor driver (H-bridge logic) interface using 8051 microcontroller.
5. LCD interface using 8051 microcontroller.

APPLICATIONS:

Automated car parking system allows the driver to get an easier view and analysis of the position of the car and the obstacles during its parking process and during this the car automatically positions itself in the particular parking slot.

Various advantages include:

- Helps to prevent scratches while parking
- Allows safe parking
- Enables fast parking
- Embedded sensors in car for parking
- Low cost

Difficulties Faced:

1. Without being inculcated the prior knowledge about the process of burning embedded C program into a standalone chip it becomes really difficult for the students to execute the process while using embedded C.
2. Breadboard testing and jumper wire testing is not a feasible option in such complicated projects due to numerous loose connections.

Future Propositions:

Till now we were using one ultasonic sensor to park the car. In this case the car should be parralel to parking slot. But we thought what if the car is not alligned to parking slot. For this problem the current prototype would be insufficient. To solve this problem one can add one more ultrasonic sensor to the current prototype. Two ultrasonic sensor could be placed to either corner of the car which can correct the allignment to parking slot.

References:

- **8051 Microcontoller and Embedded System - Mazidi**
- **Datasheet of HCSR-04**
- **www.keil.com/home/error_messages**
- **www.youtube.com/H.C.D.P.Education**