AMIT

# *Amit Graduation Project*

## *: Autonomous Car*

***Name:*** Andrew Osama Yehia Rizk
***Group:*** D76
***Email:***andrewosamayehia@gmail.com

## *Introduction:*

Autonomous car is used various applications it aims to implement a self-driving Car that can avoid the obstacles with very fast response by using a ultrasonic sensor to define obstacles on the way of the car and servo motor to adjust the angle of view of ultrasonic sensor.

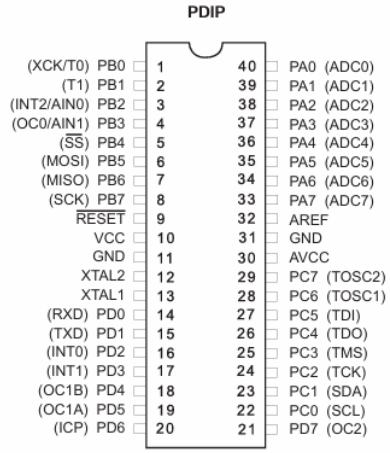
Parts:

1- Atmega 32
2- Ultrasonic sensor (**HC-SR04**)
3- Servo motor(**SG90 Micro servo**)
4- L293D(**H.Bridge Motor Driver**)
5- 2WD Robot car kit
6- LCD(16*2)

## *Parts description:*

### 1- Atmega32

Pinouts ATmega32

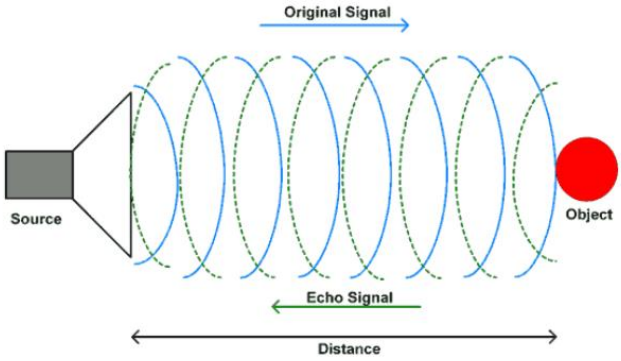



It is a High-performance, Low-power AVR® 8-bit Microcontroller which is going to be used as main brain of autonomous car.
Timer 1 and Timer 2 are needed for ultrasonic echo creation (input capture mode of timer 1) and servo motor PWM adjustment and global interrupts are required .

### 2- Ultrasonic Sensor (HC-SR04)

The ultrasonic sensor works on the principle of SONAR and RADAR system which is used to determine the distance to an object.


An ultrasonic sensor generates high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as an echo which is sensed by the receiver as shown in below figure.




Ultrasonic Sensor Working Principle: By measuring the time required for the echo to reach to the receiver, we can calculate the distance. This is the basic working principle of Ultrasonic module to measure distance.

We must give trigger pulse, so that it will generate ultrasound of frequency 40 kHz. After generating ultrasound, i.e. 8 pulses of 40 kHz, it makes echo pin high. The echo pin remains high until it does not get the echo sound back. So the width of echo pin will be the time for sound to travel to the object and return back. Once we get the time we can calculate distance, as we know the speed of sound.

**Ultrasonic Sensor HC-SR04 can measure up to range from 2 cm - 400 cm.**

Ultrasonic Sensor HC-SR04 Working Principle

1. We need to transmit a trigger pulse of at least 10 us to the HC-SR04 Trig Pin.
2. Then the HC-SR04 automatically sends Eight 40 kHz sound wave and wait for rising edge output at Echo pin.
3. When the rising edge capture occurs at Echo pin, start the Timer and wait for falling edge on Echo pin.
4. As soon as the falling edge is captured at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return back from an object.

For this operation we are going to use timer 1 16 bit register in input capture mode on PD6 on atmeag32 which is pin 40

The input capture function is used in many applications such as:

- Pulse width measurement
- Period measurement
- Capturing the time of an event

In AVR ATmega32, Timer1 can be used as an input capture to detect and measure events happening outside the microcontroller.

Upon detection of a defined event i.e. rising edge or falling edge on ICP pin (PORTD.6), the TCNT1(Timer / Counter register) value is loaded into the ICR1 (input capture) register and the ICF1 flag will get set.

To program, first, let us see for example TCCR1B (Timer Counter Control Register B)

**TCCR1B:** Timer Counter Control Register B

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ICNC1 | ICES1 | – | WGM13 | WGM12 | CS12 | CS11 | CS10 |

**Bit 7 - ICNC1:** Input Capture Noise canceller

Setting this bit activates the noise canceller. It causes a delay of 4 clock cycles as it considers a change only if it persists for at least 4 successive system clocks.

**Bit 6 - ICES1:** Input Capture Edge select

Select edge detection for input capture function.

**0 =** Capture on the falling edge

**1 =** Capture on rising edge

**Bit 4: 3 - WGM13 : WGM12:** Timer1 Mode select

These bits are used for mode selection like Normal mode, PWM mode, CTC mode, etc. here we will select normal mode, so set these bits to zero.

**Bit 2: 0 - CS12: CS10:**Timer1 Clock Select

| CS02 | CS01 | CS00 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer / Counter stopped) |
| 0 | 0 | 1 | clk (no pre-scaling) |
| 0 | 1 | 0 | clk / 8 |
| 0 | 1 | 1 | clk / 64 |
| 1 | 0 | 0 | clk / 256 |
| 1 | 0 | 1 | clk / 1024 |
| 1 | 1 | 0 | External clock source on T0 pin. Clock on falling edge |
| 1 | 1 | 1 | External clock source on T0 pin. Clock on rising edge. |

**Steps to Program**

1. Initialize the TCCR1A and TCCR1B for proper timer mode (any mode other than 8, 10, 12, 14), to select the edge (Positive or Negative).
2. Monitor the ICF1 flag in the TIFR register to see if the edge has arrived. Upon the arrival of the edge, the TCNT1 value is loaded into the ICR1 register automatically by the controller.

**Note:** Input capture pin, which is PORTD.6, has one more function i.e. output of the analog comparator. We can use ACIC bit from the ACSR register, to make this pin function as the ‘analog comparator output’ by setting it to logic HIGH. Otherwise, this pin remains as an ICP pin by default after power on or reset. So we don’t need to define this Register here.

## 3- Servo motor (SG90 Micro servo)


A servo motor is an electric device used for precise control of angular rotation. It is used where precise control is required, like in the case of control of a robotic arm.

It consists of a suitable motor with control circuitry for precise position control of the motor shaft.

It is a closed-loop system.

The rotation angle of the servo motor is controlled by applying a PWM signal to it.

By varying the width of the PWM signal, we can change the rotation angle and direction of the motor.

In this project it is going to be used for directional control of angle ultrasonic sensor view

SG90 servo motor practical duty cycle time for -90° to +90 rotation.

At ~0.52ms duty cycle we get shaft position at -90° of its rotation.

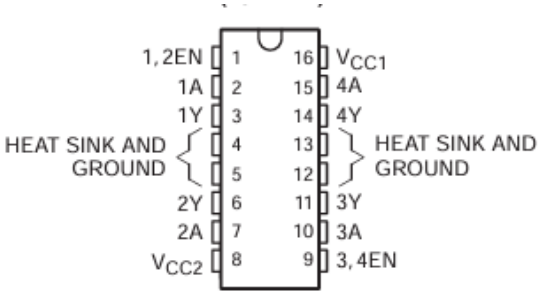
At ~1.4ms duty cycle we get shaft position at 0° (neutral) of its rotation.

At ~2.4ms duty cycle we get shaft position at +90° of its rotation.

In this project we will use timer 2 for PWM control so we don't interfere with operation of input capture mode of timer 1 to do that we are going to use PD7(OCR2) PIN on Atmega 32

And adjust it PWM in code .

4- **L293D (H. Bridge Motor Driver)**




It is used to drive dc motors for wheels.

DC motor converts electrical energy in the form of Direct Current into mechanical energy.

In the case of the motor, the mechanical energy produced is in the form of a rotational movement of the motor shaft.

The direction of rotation of the shaft of the motor can be reversed by reversing the direction of Direct Current through the motor.

The motor can be rotated at a certain speed by applying a fixed voltage to it. If the voltage varies, the speed of the motor varies.

Thus, the DC motor speed can be controlled by applying varying DC voltage, whereas the direction of rotation of the motor can be changed by reversing the direction of current through it.

For applying varying voltage, we can make use of the PWM technique.

For reversing the current, we can make use of an H-Bridge circuit or motor driver ICs that employ the H-Bridge technique or other mechanisms.

As we are using amit`s pcb board we are going to use the following sequence for h bridge connection

H-Bridge pins
H_EN1 → PORTD.4
H_EN2→ PORTD.5
H_A1 → PORTC.3
H_A2 → PORTC.4
H_A3 → PORTC.5
H_A4 → PORTC.6
C.0
SDA→PORTC

## 5- LCD(16*2)

LCDs (Liquid Crystal Displays) are used in embedded system applications for displaying various parameters and status of the system.


LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each.

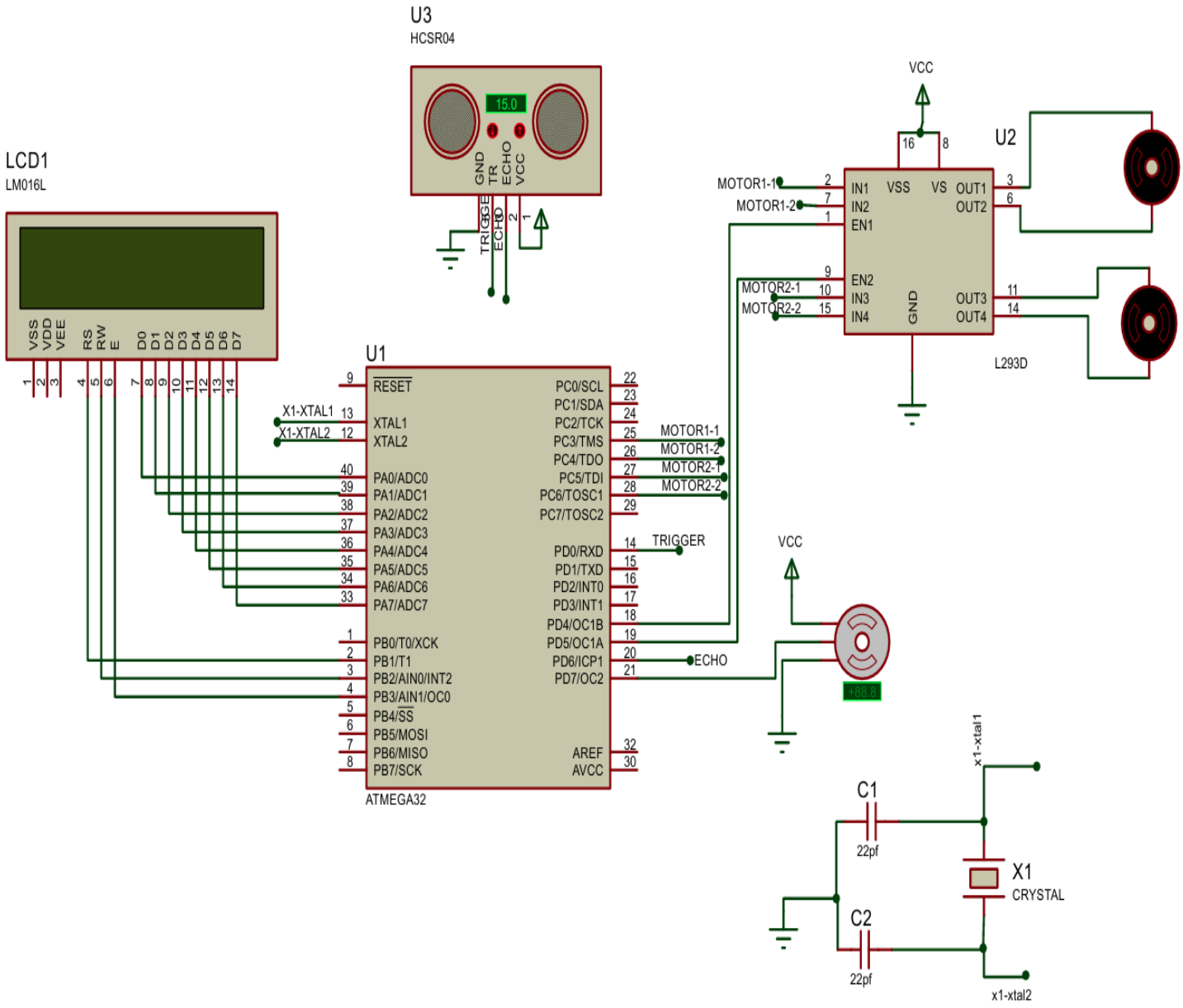
LCD 16x2 can be used in 4-bit mode or 8-bit mode.

It is also possible to create custom characters.

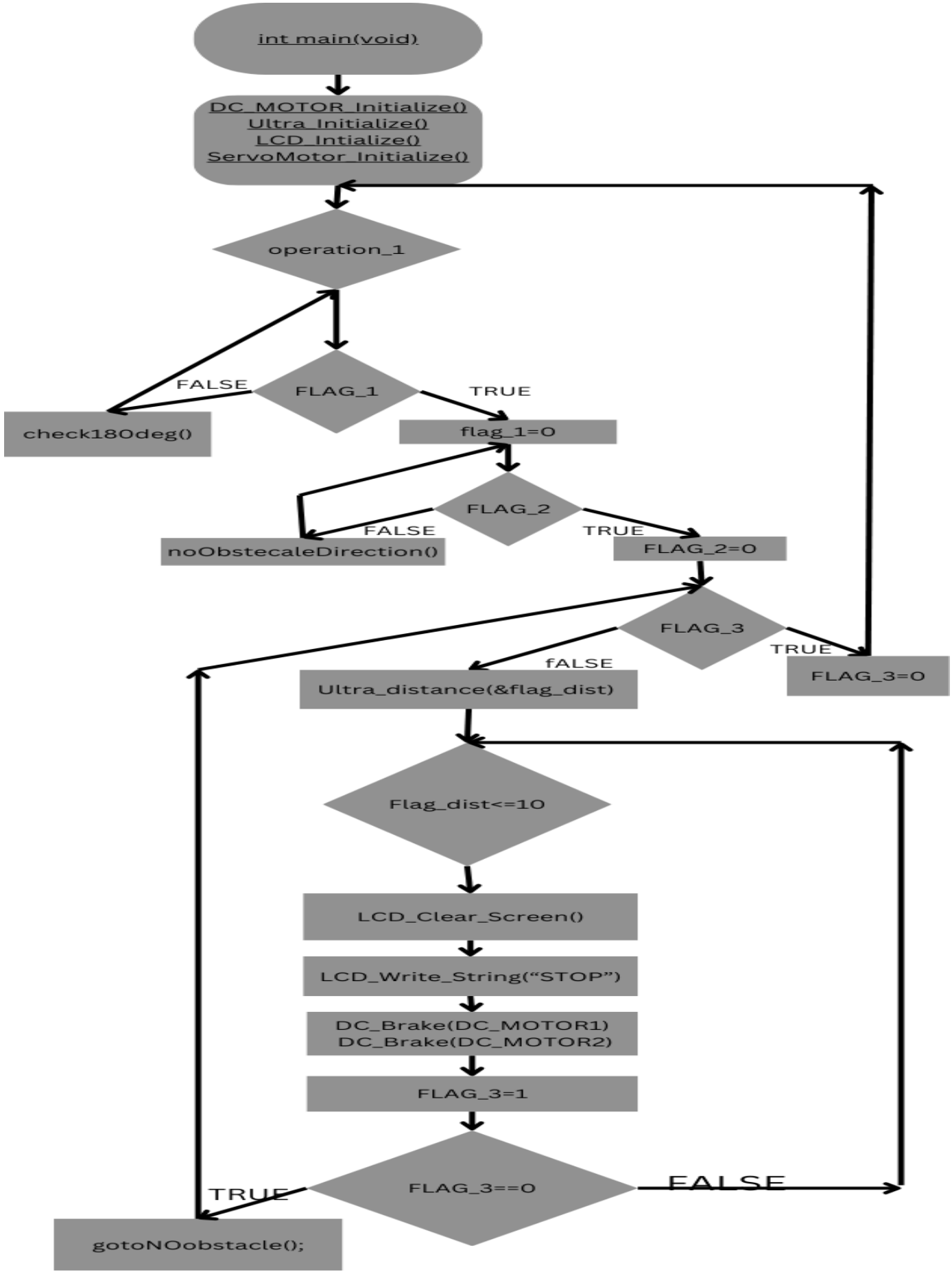
It has 8 data lines and 3 control lines that can be used for control purposes.

It is used in this project to display distance observed by ultrasonic sensor in the three different directions (left, straight, right)

# Proteus connection



Flow chart


int main(void)
DC_MOTOR_Initialize()
Ultra_Initialize()
LCD_Intialize()
ServoMotor_Initialize()
operation_1
FLAG_1
FALSE
TRUE
check180deg()
flag_1=0
FLAG_2
FALSE
TRUE
noObstecaleDirection()
FLAG_2=0
FLAG_3
fALSE
TRUE
Ultra_distance(&flag_dist)
FLAG_3=0
Flag_dist<=10
LCD_Clear_Screen()
LCD_Write_String("STOP")
DC_Brake(DC_MOTOR1)
DC_Brake(DC_MOTOR2)
FLAG_3=1
FLAG_3==0
TRUE
FALSE
gotoNOobstacle();