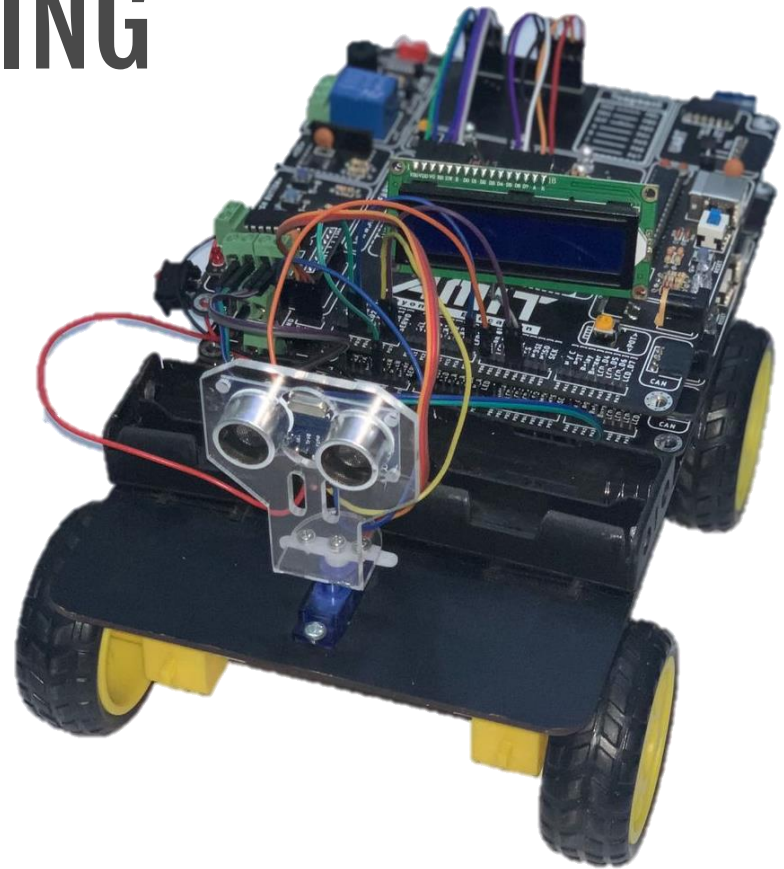
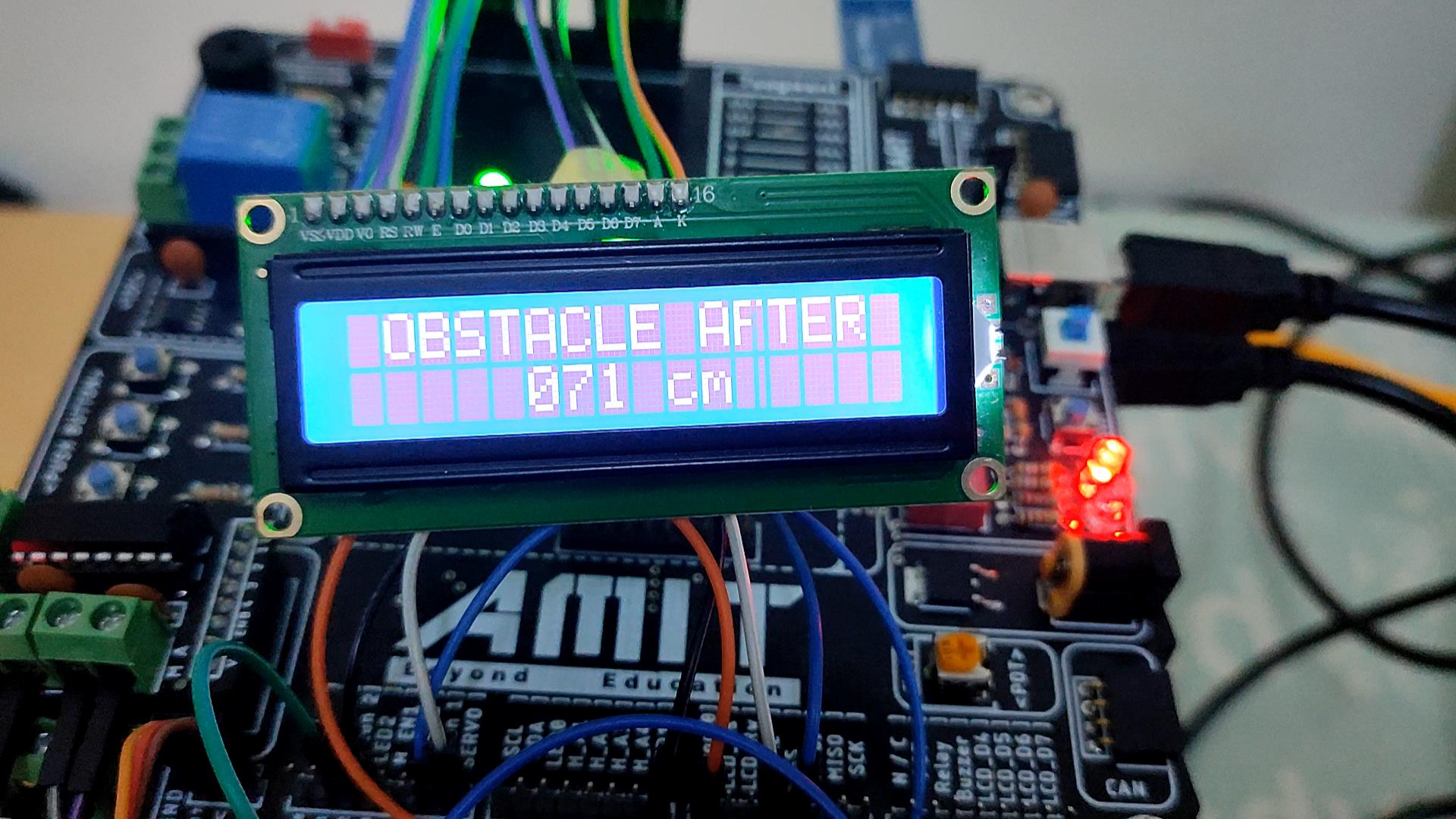
AOBSTACLE-AVOIDING MOBILE ROBOT



Submitted By:
Mahammad Heshmat Abd El Raheem


OBSTACLE AFTER
071 cm

MAIN COMPONENTS

01 **ATmega32**

2x Atmega 32
To control to start stop condition

02 **LCD**

To displays the current direction that car immediately move.

03 **Bluetooth Module**

To interrupt the main micro controller to start it or stop it

04 **UltraSonic Sensor**

To check if there is an object and how far it is

05 **4 Dc motor & Wheels**

To move the robot

06 **Servo Motor**

To control the movement of the ultrasonic sensor

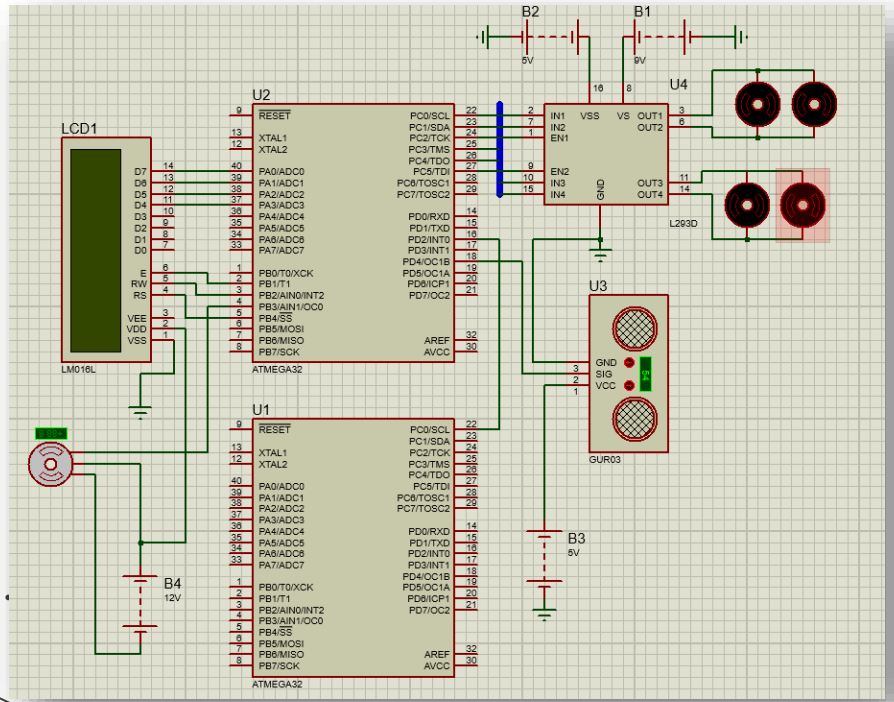
07 **Batteries**

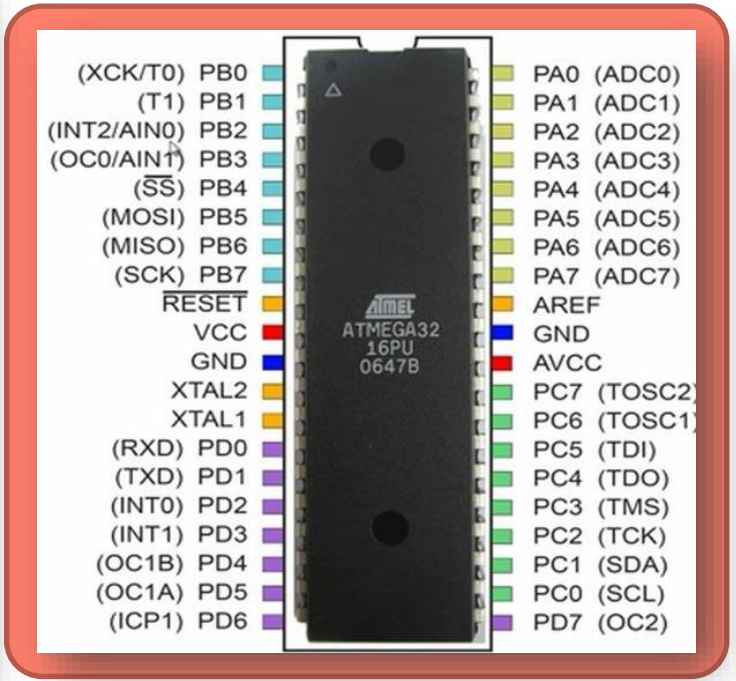
To provide power to DC motor & ATmega32

08 **Dc Driver**

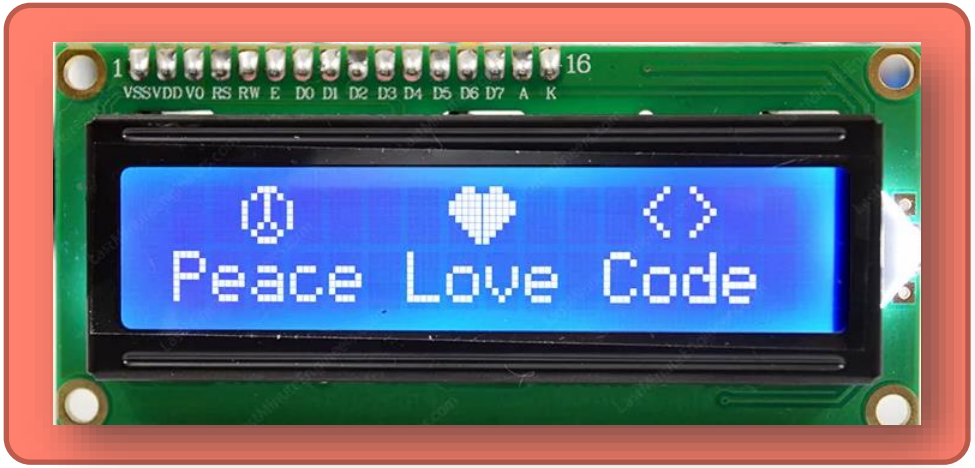
L293D H-bridge

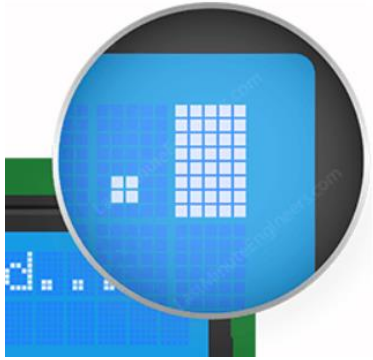
ATmega32


B2
5V
B1
9V
U2
U4
LCD1
U3
U1
B3
5V
B4
12V
L293D
GUR03
LM016L
ATMEGA32
ATMEGA32

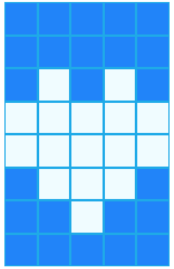

(XCK/T0) PB0
(T1) PB1
(INT2/AIN0) PB2
(OC0/AIN1) PB3
(SS) PB4
(MOSI) PB5
(MISO) PB6
(SCK) PB7
RESET
VCC
GND
XTAL2
XTAL1
(RXD) PD0
(TXD) PD1
(INT0) PD2
(INT1) PD3
(OC1B) PD4
(OC1A) PD5
(ICP1) PD6
PA0 (ADC0)
PA1 (ADC1)
PA2 (ADC2)
PA3 (ADC3)
PA4 (ADC4)
PA5 (ADC5)
PA6 (ADC6)
PA7 (ADC7)
AREF
GND
AVCC
PC7 (TOSC2)
PC6 (TOSC1)
PC5 (TDI)
PC4 (TDO)
PC3 (TMS)
PC2 (TCK)
PC1 (SDA)
PC0 (SCL)
PD7 (OC2)
ATMEGA32
16PU
0647B

16x2 LCD





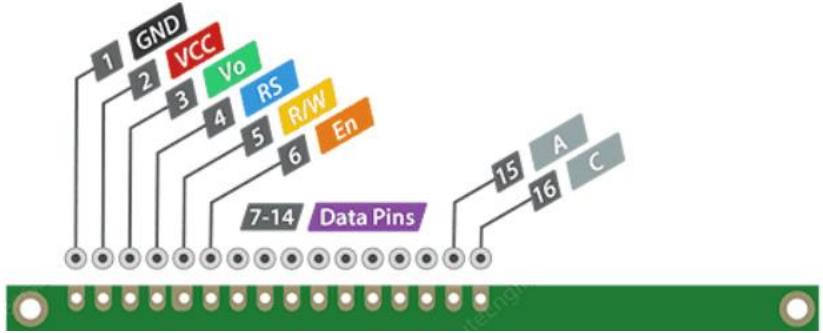
Start selecting pixels



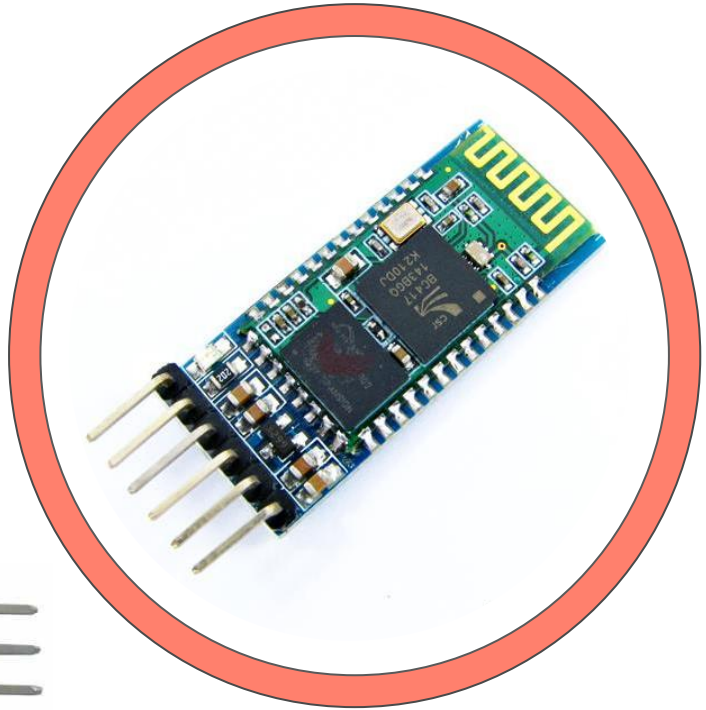
Clear all

```
byte
Character[8] =
{
  0b00000,
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000
};
```

Copy this code to your sketch



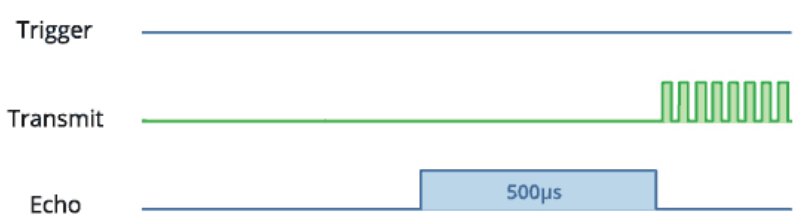
Bluetooth Module



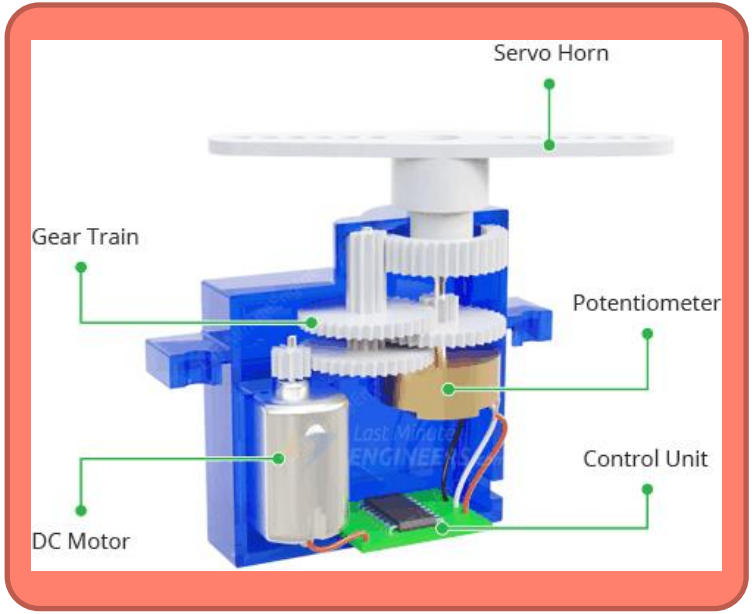
TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

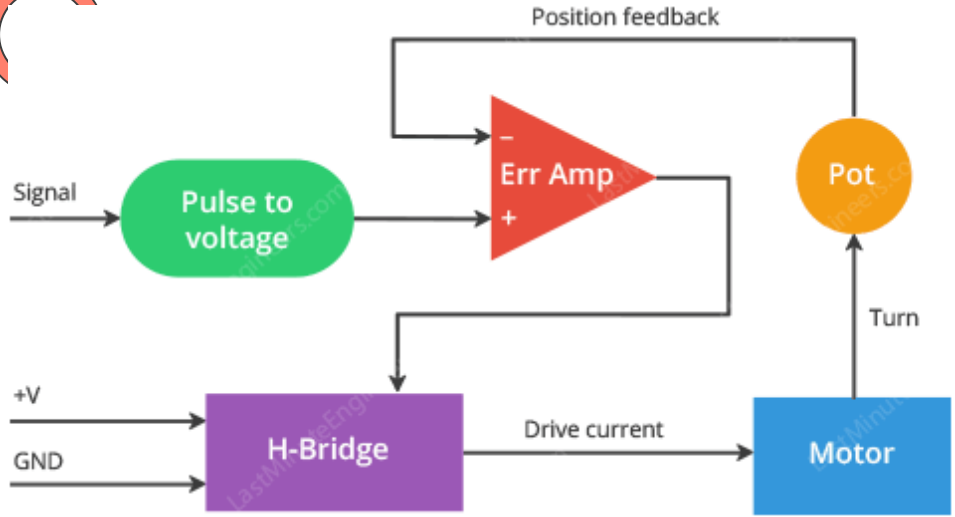
RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

UltraSonic Sensor



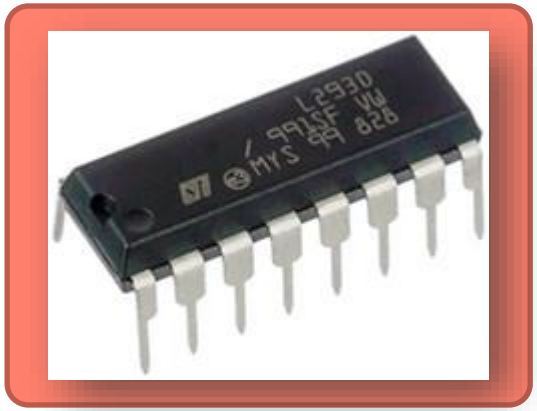
Servo Motor

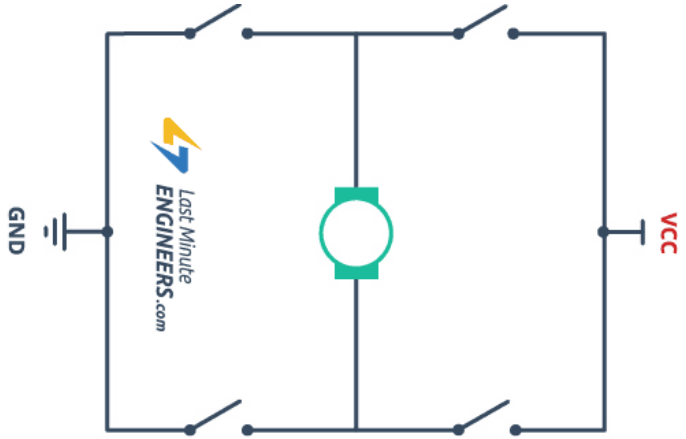




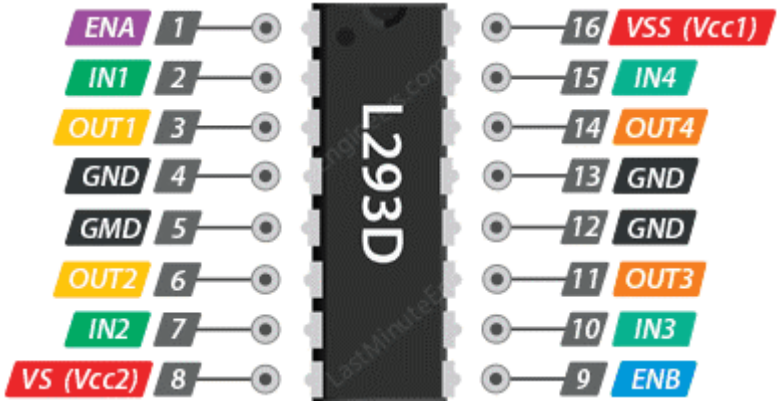
0 Degrees

L293D H-bridge





IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF



Peripheral Which Used :

01

DIO

Control To Pins

02

Lcd

To Displays The Closest Obstacle & The Direction To Move

03

Interrupt

Use EXTI 0&1
Exit 0 : Started Or Stopped Car
Exit 1 : To Avoid Obstacles

04

Timer 0

To Control The Servo Motor

05

Timer 1

Store Data From Ultrasonic

06

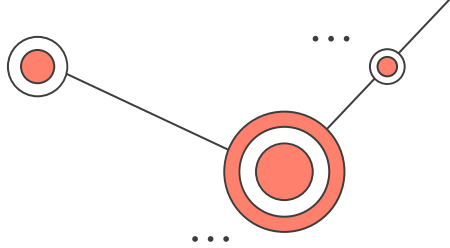
UART

To Communicate With The Bluetooth Module

01

FUNCTION.C

FUNCTION.C FILE



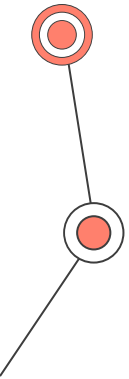
```
void GO_FORWARD (void)...
void BACKWORD (void)...
void STOP (void)...
void RETURN_180 (void)...
void TURN_RIGHT (void)...
void TURN_LEFT (void)...
void CHECK(void)...
void TIMER1_voidOvInt(void)...
f64 ULTRASONIC_f64Distance(void)...
```

To Control To Dc Motor

To TAKE A DECISION FROM THE ULTRASONIC

To Control OV INTERRUPT

To GET THE DISTANCE



To Control To Dc Motor

```
void GO_FORWARD (void)
{
    DIO_voidSetPinValue( H_A1 , HIGH );
    DIO_voidSetPinValue( H_A2 , LOW );
    DIO_voidSetPinValue( H_A3 , HIGH );
    DIO_voidSetPinValue( H_A4 , LOW );

    DIO_voidSetPinValue( H_EN1 , HIGH );
    DIO_voidSetPinValue( H_EN2 ,HIGH );
}
```

```
void BACKWORD (void)
{
    DIO_voidSetPinValue( H_A1 , LOW );
    DIO_voidSetPinValue( H_A2 , HIGH );
    DIO_voidSetPinValue( H_A3 , LOW );
    DIO_voidSetPinValue( H_A4 , HIGH );

    DIO_voidSetPinValue( H_EN1 , HIGH );
    DIO_voidSetPinValue( H_EN2 ,HIGH );

    _delay_ms(500);

    STOP();
}
```

IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

To Control To Dc Motor

```
void TURN_LEFT (void)
{
    DIO_voidSetPinValue( H_A1 , HIGH);
    DIO_voidSetPinValue( H_A2 , LOW );
    DIO_voidSetPinValue( H_A3 , LOW );
    DIO_voidSetPinValue( H_A4 , LOW );

    DIO_voidSetPinValue( H_EN1 , HIGH);
    DIO_voidSetPinValue( H_EN2 , HIGH);
    _delay_ms(1000);
    STOP ();
}
```

```
void TURN_RIGHT (void)
{
    DIO_voidSetPinValue( H_A1 , LOW );
    DIO_voidSetPinValue( H_A2 , LOW );
    DIO_voidSetPinValue( H_A3 , HIGH );
    DIO_voidSetPinValue( H_A4 , LOW );

    DIO_voidSetPinValue( H_EN1 , HIGH);
    DIO_voidSetPinValue( H_EN2 , HIGH);
    _delay_ms(1000);
    STOP ();
}
```

```
void STOP (void)
{
    DIO_voidSetPinValue( H_EN1 , LOW );
    DIO_voidSetPinValue( H_EN2 , LOW );

    DIO_voidSetPinValue( H_A1 , LOW );
    DIO_voidSetPinValue( H_A2 , LOW );
    DIO_voidSetPinValue( H_A3 , LOW );
    DIO_voidSetPinValue( H_A4 , LOW);

}
```

To TAKE A DECISION FROM THE ULTRASONIC

```
void CHECK(void)
{
    LCD_voidClear();
    LCD_voidSetCursor( LCD_U8_LINE1 , 2 );
    LCD_voidSendString("CHICKING...");

    TIMER0_void_SetCompareVal(24);
    _delay_ms(100);
    TIMER0_void_SetCompareVal(13);
    L_DISTANCE = ULTRASONIC_f64Distance();
    _delay_ms(300);
    TIMER0_void_SetCompareVal(34);
    R_DISTANCE = ULTRASONIC_f64Distance();
    _delay_ms(300);
    TIMER0_void_SetCompareVal(24);

    if ((L_DISTANCE < 10.00 ) && (R_DISTANCE < 10.00))
    {
        LCD_voidSetCursor( LCD_U8_LINE2 , 2 );
        LCD_voidSendString("RETURN BACK");
        BACKWORD(); RETURN_180();
    }
    else if (L_DISTANCE > R_DISTANCE)
    {
        LCD_voidSetCursor( LCD_U8_LINE2 , 4 );
        LCD_voidSendString("TURN LEFT");
        BACKWORD(); TURN_LEFT();
    }
    else if (R_DISTANCE > L_DISTANCE)
    {
        LCD_voidSetCursor( LCD_U8_LINE2 , 3 );
        LCD_voidSendString("TURN RIGHT");
        BACKWORD(); TURN_RIGHT();
    }
    LCD_voidClear();
    LCD_voidSetCursor( LCD_U8_LINE1 , 1 );
    LCD_voidSendString("OBSTACLE AFTER");
    LCD_voidSetCursor( LCD_U8_LINE2 , 9 );
    LCD_voidSendString("cm");
}
```

To GET THE DISTANCE

```c
f64 ULTRASONIC_f64Distance(void)
{
    DIO_voidSetPinValue(DIO_U8_PIN8,DIO_U8_HIGH);
    _delay_us(10);
    DIO_voidSetPinValue(DIO_U8_PIN8,DIO_U8_LOW);

    TCNT1 = 0;          // Clear Timer counter
    TCCR1B = 0b01000010 ;       // Setting for capture rising edge,  using clk/8 pre-scaler
    TIFR = 1<<ICF1;     // Clear ICP flag (Input Capture flag)
    TIFR = 1<<TOV1;     // Clear Timer Overflow flag

    /*Calculate width of Echo by Input Capture (ICP) on PortD PD6*/
    while ((TIFR & (1 << ICF1)) == 0);  // Wait for rising edge

    TCNT1 = 0;          // Clear Timer counter
    TCCR1B = 0b00000010 ;       // Setting for capture falling edge, using clk/8 pre-scaler
    TIFR = 1<<ICF1;     // Clear ICP flag (Input Capture flag)
    TIFR = 1<<TOV1;     // Clear Timer Overflow flag

    TimerOverflow = 0;  // Clear Timer overflow count

    while ((TIFR & (1 << ICF1)) == 0); // Wait for falling edge

    count = ICR1 + (65535 * TimerOverflow); // Take value of capture register
    distance = ((double)count * 0.008575 ) ;   // 8 MHz Timer freq, sound speed =343 m/s
    _delay_ms(100);

    return distance;
}
```

02

MAIN.C

Main Microcontroller

ISR FOR INTERROPTS

```
ISR(INT0_vect)
{
    if ( LOCK==0 )
    {
        LOCK=1;
    }
    else
    {
        LOCK=0;
        STOP();
    }
}
```

MAKING TOGGLE
TO START & STOP MAGNAM

```
ISR(INT1_vect)
{
    STOP();
    CHECK();
    DIO_voidSetPinValue(INT1_PIN,HIGH);
}
```

To MAKE A DESCISION

Int Main Code

```c
int main(void)
{
    DIO_voidInit();
    LCD_voidInit();
    TIMER0_void_Init();   // Enable Timer0 overflow interrupts
    TIMER1_voidOvInt();  // Enable Timer1 overflow interrupts
    DIO_voidSetPinValue(INT1_PIN,HIGH);  //PULLUP RESISTOR TO INT1
    EXTI_voidInitINT0();         //set INT0 on falling edge on PD2
    EXTI_voidEnableINT0();
    EXTI_voidInitINT1();        //set INT0 on falling edge on PD3 from PC7 that mean that our controller will interrupt its self
    EXTI_voidEnableINT1();
    GIE_voidEnable();   // Enable global interrupt

    TIMER0_void_SetCompareVal(24);  //SET THE SERVO ON 90 DEGREE

    LCD_voidClear();                                // clear our LCD
    LCD_voidSetCursor( LCD_U8_LINE1 , 1 );          // set our cursor on the upper line  in digit 1
    LCD_voidSendString("OBSTACLE AFTER");
    LCD_voidSetCursor( LCD_U8_LINE2 , 9 );
    LCD_voidSendString("cm");
```

Main Loop

```
while(1)
{
    while ( LOCK == 1 )
    {
        GO_FORWARD();

        Obstacle_Dis = ULTRASONIC_f64Distance();
        LCD_voidSetCursor( LCD_U8_LINE2 , 5 );
        LCD_voidSendNumberIII ((u16)Obstacle_Dis);

        if( Obstacle_Dis < 45.0000 )
        {
            DIO_voidSetPinValue(INT1_PIN,LOW);
        }
    }
}
```

03

MAIN.C

Sub-Microcontroller

MAIN .C (UART)

```c
int main(void)
{
    DIO_voidInit();
    UART_voidInit();
    u8 SWITCH ;

    while (1)
    {
        SWITCH =UART_u8ReceiveByte();

        if ( SWITCH == 'a' )
        {
            DIO_voidSetPinValue(LED2,HIGH);
            DIO_voidSetPinValue(DIO_U8_PIN28,HIGH);
            _delay_us(10);
            DIO_voidSetPinValue(LED2,LOW);
            DIO_voidSetPinValue(DIO_U8_PIN28,LOW);
            _delay_us(10);
            DIO_voidSetPinValue(LED2,HIGH);
            DIO_voidSetPinValue(DIO_U8_PIN28,HIGH);
        }
    }
}
```

Using Polling To Receive Data From Bluetooth Module

THANKS !