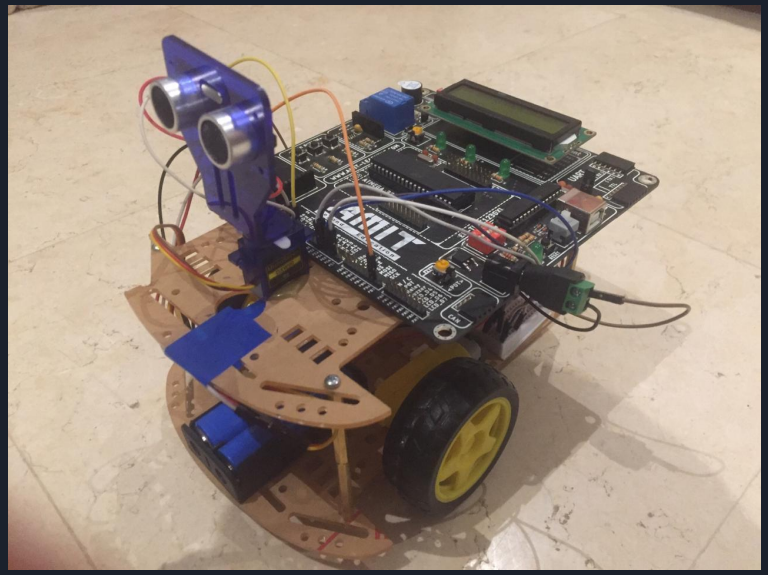
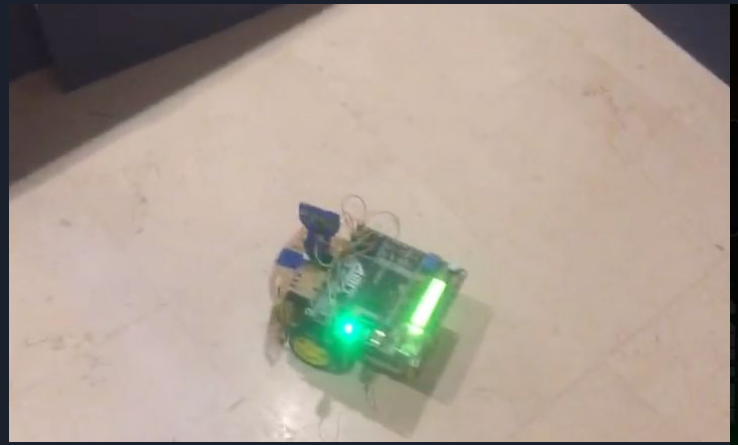
Autonomous Car



By: Ahmed Yasser Amin Sallam
Group: Emb Nasr 43
Email: ahmedys2010@gmail.com

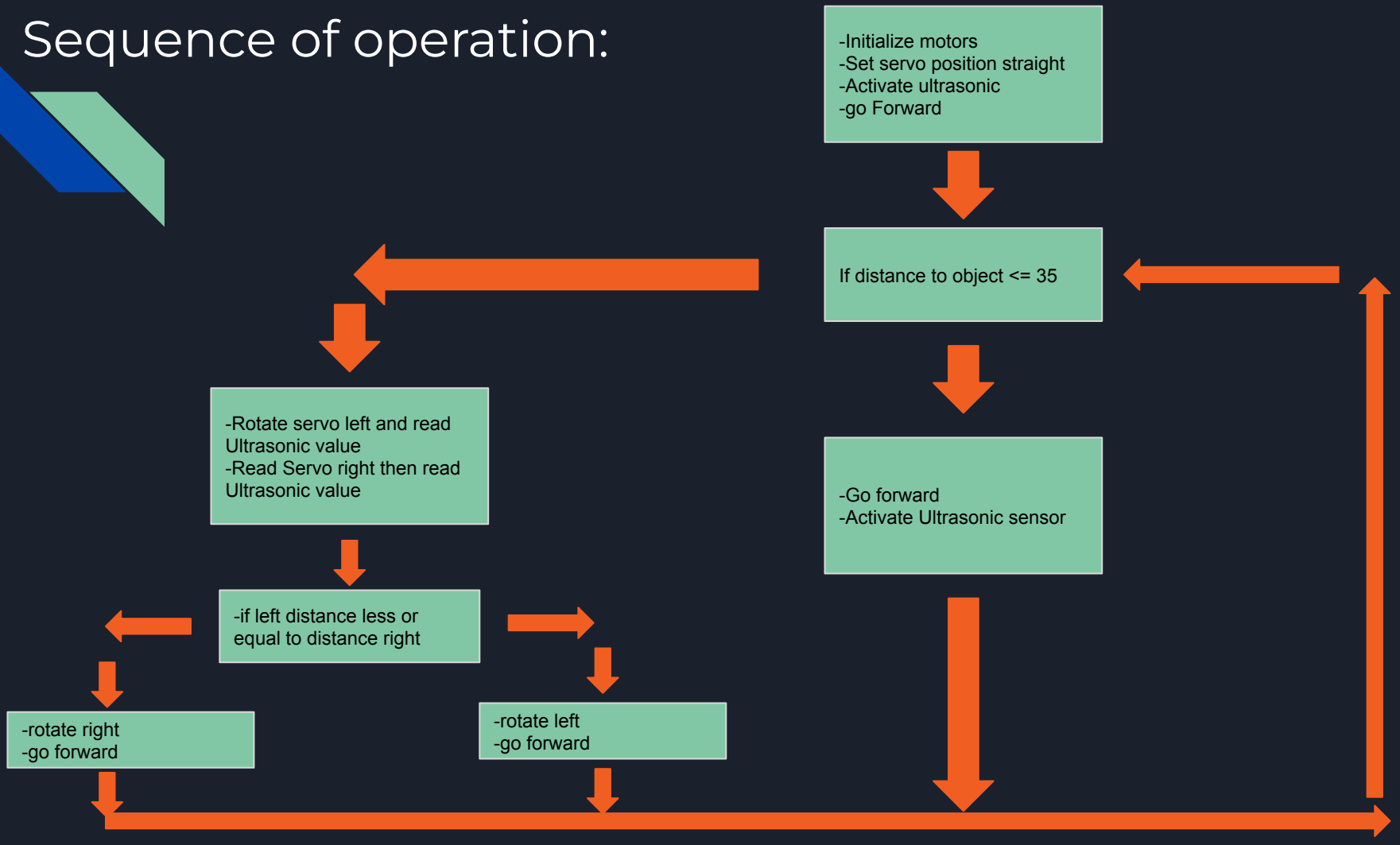
Description:

The autonomous car avoids collision with its surroundings by using a sensor to stop it at a certain distance, then a servo motor attached to the sensor will rotate it 180 degrees to check left and right sides for which position to move to. If the left side distance is greater than right side distance then the robot will turn left and move at the direction, and vice versa. These actions continues indefinitely.

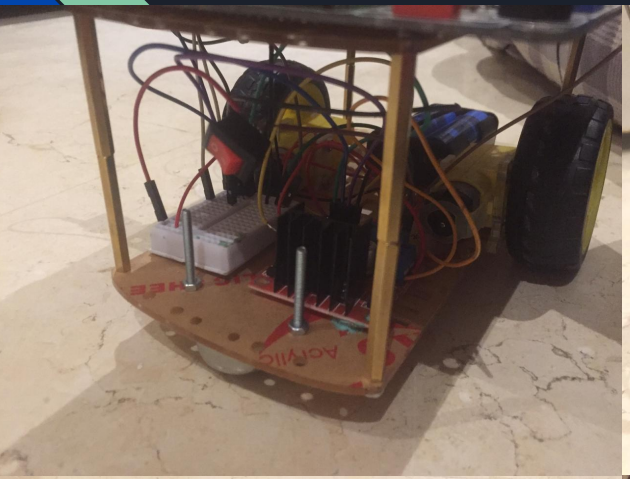


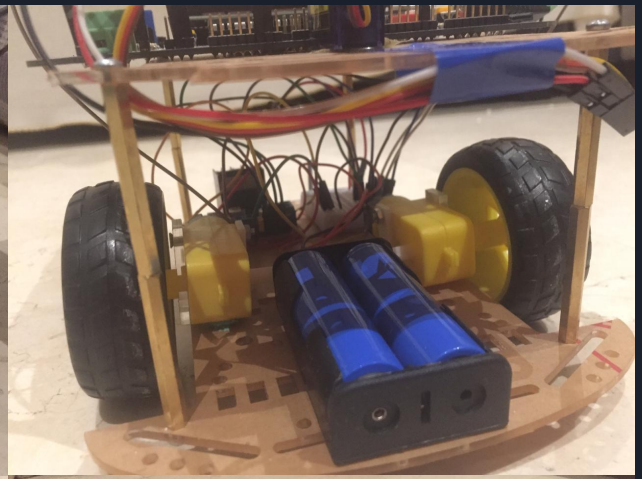
Attached to the robot is a 16x2 LCD that consistently tells the direction the robot is currently heading and the data currently obtained from the Ultrasonic sensor.

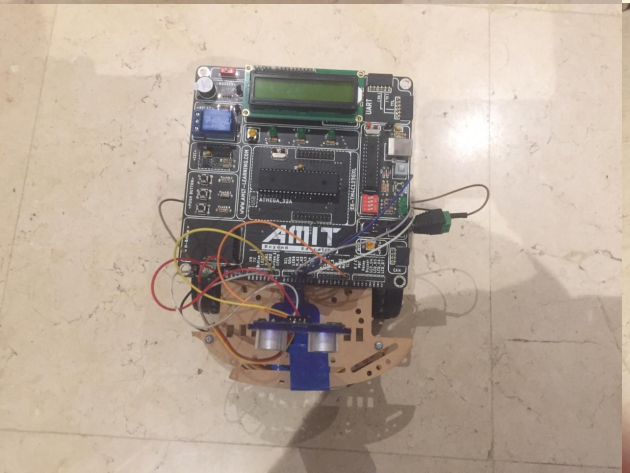
Sequence of operation:

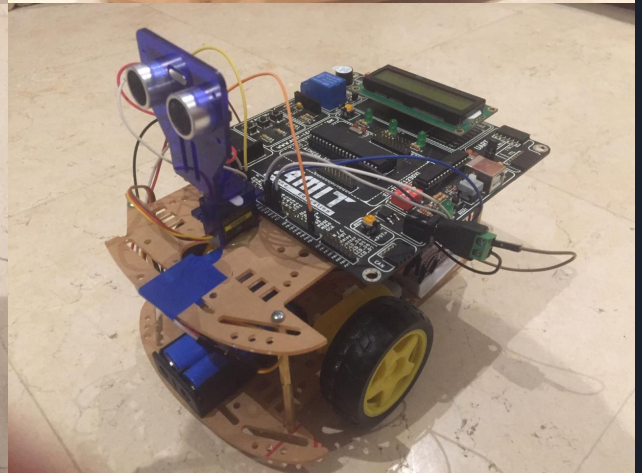


Components:





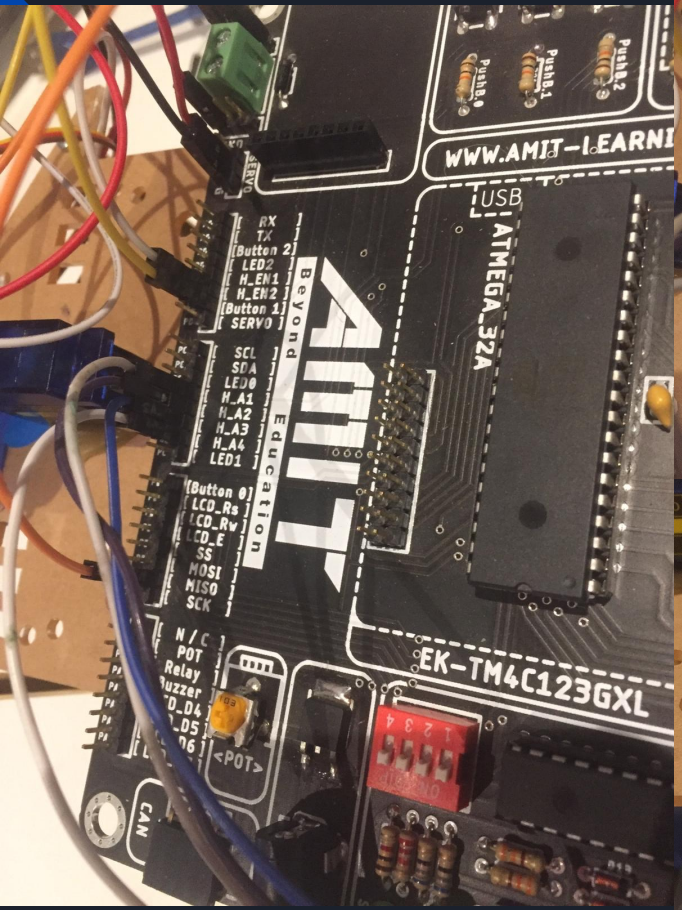


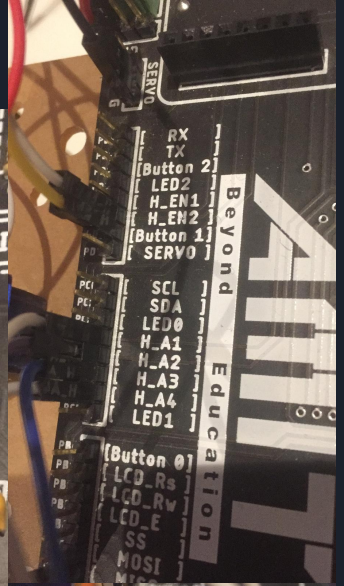


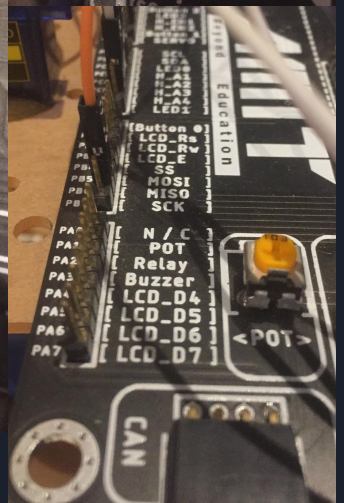
List:

1) 1x Amit Development Board(Atmega32 Microcontroller)
2) 2x 3.8V Lithium Batteries
3) 1x 2 Battery Holder
4) 2x Gear DC Motor
5) 1x L298N Motor Driver
6) 1x HC_SR04 Ultrasonic sensor
7) 1x Servo motor DxW90
8) 1x 16x2 LCD
9) 1x Mini Breadboard
10) 1x DC power Jack
11) 1x Robot car chassis
12) wires

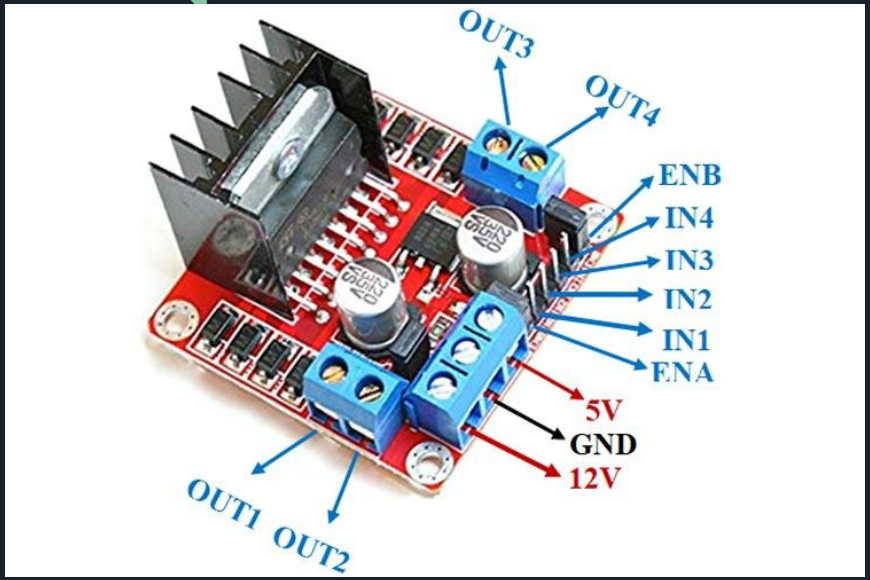
Pin distribution:







The Layout mostly follows the already laid out configuration on the AMIT development board. We didn't use the PD7 pin for the servo motor as it is connected to Timer2 but we need Timer1 to control the servo motor. Also we used an L298N motor driver instead of the custom built motor driver from the kit as we are only going to use 4 digital pins to control direction and rotate the motors at max speed. I addition, the PD5 pin used for the servo motor is connected to the Amit boards H bridge enable pin so this is another reason to why we couldn't use it as well



IN4 = H-A4 = PC6

IN3 = H-A3 = PC5

IN2 = H-A2 = PC4

IN1 = H-A1 = PC3

Ultrasonic sensor Calculations:

Calculation (distance in cm) (H1)

Sound velocity = 343.00 m/s = 34300 cm/s

$$\text{The distance of Object (in cm)} = \frac{\text{SoundVelocity} * \text{TIMER}}{2}$$

$$= \frac{34300 * \text{TIMER}}{2}$$

$$= 17150 * \text{TIMER}$$

Now, here we have selected an internal 16 MHz oscillator frequency for ATmega32, with No-prescaler for timer1 frequency. Period = 0.0625 us

So, the timer gets incremented after 0.0625 us.

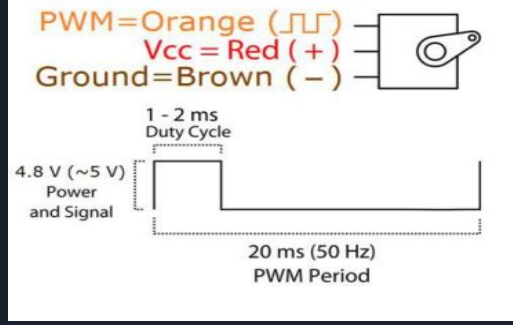
Distance = 17150 x (TIMER value) x 0.0625 x 10^-6 cm

Distance = (TIMER value) / 932.95 cm

```c
/*
 * Ultrasonic.c
 *
 * Created: 2/4/2023 5:37:55 PM
 *  Author: Ahmed Yasser
 */
#include "Ultrasonic.h"

void Ultrasonic_Init(void){
        TimerOvf = 0;
        SetBit(DDRB_Reg, 6);
        ClearBit(DDRD_Reg, 6);
        sei();
        SetBit(TMSK_REG, 2);
        TCCR1A_REG = 0;
        TCCR1B_REG = 0;
}
void Ultrasonic_Ping(void){
        SetBit(PORTB_Reg, Trigger);
        _delay_us(10);
        ClearBit(PORTB_Reg, Trigger);
        TCNT1L_REG = 0;
        SetBit(TCCR1B_REG, 6);
        SetBit(TCCR1B_REG, 0);
        SetBit(TIFR_REG, 5);
        SetBit(TIFR_REG, 2);
        while ((TIFR_REG & (1 << 5)) == 0);
        TCNT1L_REG = 0;
        ClearBit(TCCR1B_REG, 6);
        SetBit(TCCR1B_REG, 0);
        SetBit(TIFR_REG, 5);
        SetBit(TIFR_REG, 2);
        TimerOvf = 0;
        while ((TIFR_REG & (1 << 5)) == 0);
        count = ICR1L_REG + (65535 * TimerOvf);
        distance = (double)count / 932.95;
}

ISR(TIMER1_OVF_vect)
{
        TimerOvf++;
}
```

```c
/*
 * Ultrasonic.h
 *
 * Created: 2/4/2023 5:37:28 PM
 *  Author: Ahmed Yasser
 */
#ifndef ULTRASONIC_H_
#define ULTRASONIC_H_

#include <avr/interrupt.h>
#include <string.h>
#include "Clock.h"
#include "STD_Types.h"
#include "TIMER1.h"
#include "DIO.h"
#include "LCD.h"

#define Trigger 6

volatile int TimerOvf;
long count;
double distance;

void Ultrasonic_Init(void);
void Ultrasonic_Ping(void);

#endif
```

Servo Motor Calculations:



Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

Because a period with frequency 50 Hz is required to operate the servo motor we cannot use Timer2 as it can only go as low as 61 Hz. We can achieve 50 Hz by using timer1 and setting it to fast pwm with TOP = ICR1 register. Then the OCR1A register value can be chosen based on this frequency.

If we try to put the biggest prescaler we have we can only reach 312.5, this cannot be reached by an 8 bit timer.

Although it states the required pulse for middle right or left, it may not be exact on physical application so we have to choose the right value through trial and error. For example, if we calculate for 0 position, we get a value of 2000 but in actuality it is 3080. So OCR1A = 3080 for 0 position.

Our specs:

Prescaler = 8.

Fclk = 16 MHZ

Desired frequency = 50Hz

-If we put them in the equation then TOP which is ICR1 = 3999

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

Table 47. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Recommendations:

It is needless to say that the code is not perfect and can be optimized and enhanced further for better response and operation. There are some additions to the robot that will make it more efficient in its job.

1) Adding 2 more ultrasonic sensors to the front side will allow it to sense corners as an ultrasonic sensor can only sense what is exactly straight in front of it and not beside it. Because of this the robot may hit corners.
2) The Amit Development board offers easy use of devices however this comes at a cost of freedom so it may be better to make your own development board from scratch if you really need that added freedom.
3) The robot does not always go on a straight line and the reason for this is due to motor slack. This can be solved by getting an encoder and checking the ratio of speed between the motors and multiplying the ratio on the speed of the motor.

References:

- https://www.electronicwings.com/avr-atmega/servo-motor-interfacing-with-atmega16
- https://www.electronicwings.com/avr-atmega/ultrasonic-module-hc-sr04-interfacing-with-atmega1632
- https://exploreembedded.com/wiki/Interfacing_LCD_in_4-bit_mode_with_8051
- https://components101.com/modules/l293n-motor-driver-module
-

THANK YOU