OBSTACLE AVOIDING CAR--MOTOR CONTROL IMPLEMENTATION USING STM32F401CB MICROCONTROLLER

EMBEDED SYSTEMS LAB

Abstract

In This Project we set out to implement Motor Control system, speed and direction, along with the Ultrasonic sensor to achieve the objective of building an obstacle sensitive car prototype using STM32F401CB Microcontroller.

TALHA REHMAN

2018-UET-NML-ELECT-07

Group Members:

- **Arslan Mahmood**
- **Haroon Rasheed**
- **Muhammad Usama**

NAMAL INSTITUTE

Department of Electrical Engineering

EMBEDED SYSTEMS LABORATORY

LAB INSTRUCTOR: SIR AWAIS YAQOOB

COURSE INSTUCTOR: DR. HAMZA ZAD GUL

SEMESTER PROJECT 1: Motor Control / Obstacle Avoiding Car

Report by:

TALHA REHMAN

talha2018@namal.edu.pk

My LinkedIn Profile : TALHA REHMAN

CONTENTS

Objectives: 3
Knowledge-Course Modules used: 3
Requirements: 3
Summary: 4
Introduction: 5
Body 5
STM32CubeMx: 5
HCSR04 Ultrasonic Sensor: 7
SERVO Motor Control: 15
L298 Motor Driver – Motor Direction and Speed Control 19
Obstacle Avoiding Logic: 27
Conclusion 33
Simulation limitations: 33
References 34
Appendix 35

Objectives:

Our goal is to use the Microcontroller to implement following objectives

- To build a Motor Control System, Direction and Speed.
- To use Ultrasonic Sensor (HCSR-04) for Measuring the Distance.
- To implement the servo motor control for controlling the position of sensor.
- To build a combine logic of an obstacle sensitive car system and make its hardware working prototype using the above points.

Knowledge-Course Modules used:

Following Pre-Knowledge of Microcontroller and course modules were covered in this project:

- <u>PWM generation</u> of desired time period <u>using Timers</u>.
- Configuring and Using <u>GPIOs</u> (general purpose input and output) Ports of Microcontroller.
- Using <u>Timer</u> as an <u>Input Capture Device</u>.
- Interfacing the LEDs with Microcontroller.
- Proteus software and using its libraries for simulation.
- Using STM32cubeMx to generate initial code, and configure Microcontrollers PINs
- Using KEIL u vision 5 to build logic and write code for working of microcontroller.
- C programing language.

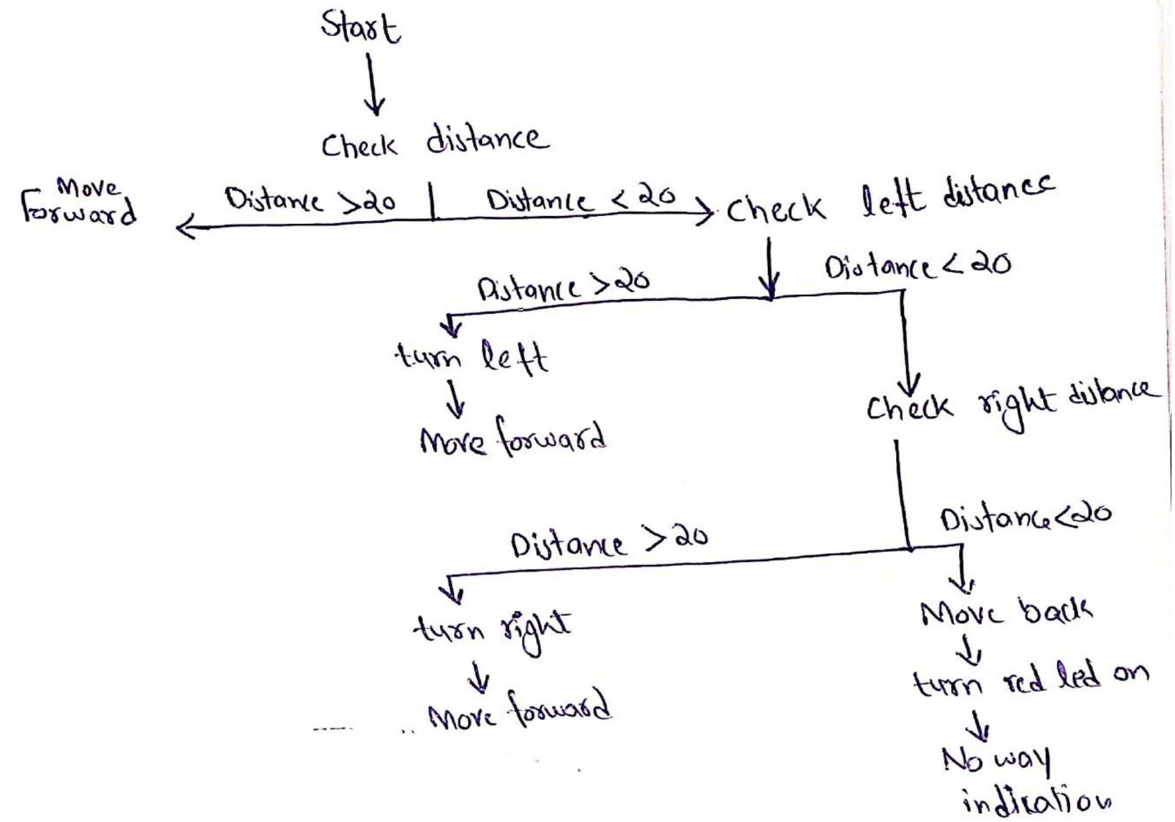
Requirements:

Following Hardware Instruments and Software support is required for this project:

1. Hardware Requirements:
    - STM32F401CB Microcontroller
    - L298 Motor Driver
    - HCSR04 Ultrasonic sensor
    - Servo 180 degree
    - DC Motors
2. Software requirements:
    - KEIL u vision 5
    - Proteus
    - Stm32CubeMx

Summary:

The following flow chart explains the basic working of our Project.



Motor Control

DC motor : i) direction control → left turn logic, right turn logic, forward logic, backward logic — using GPIO outputs

ii) speed control ⇓ Duty cycle control (PWM) using timer

Sensor

i) Timer as Input Capture device

ii) Microsecond delay function with timer

Servo

PWM signal control to turn servo to desired angle.

Introduction:

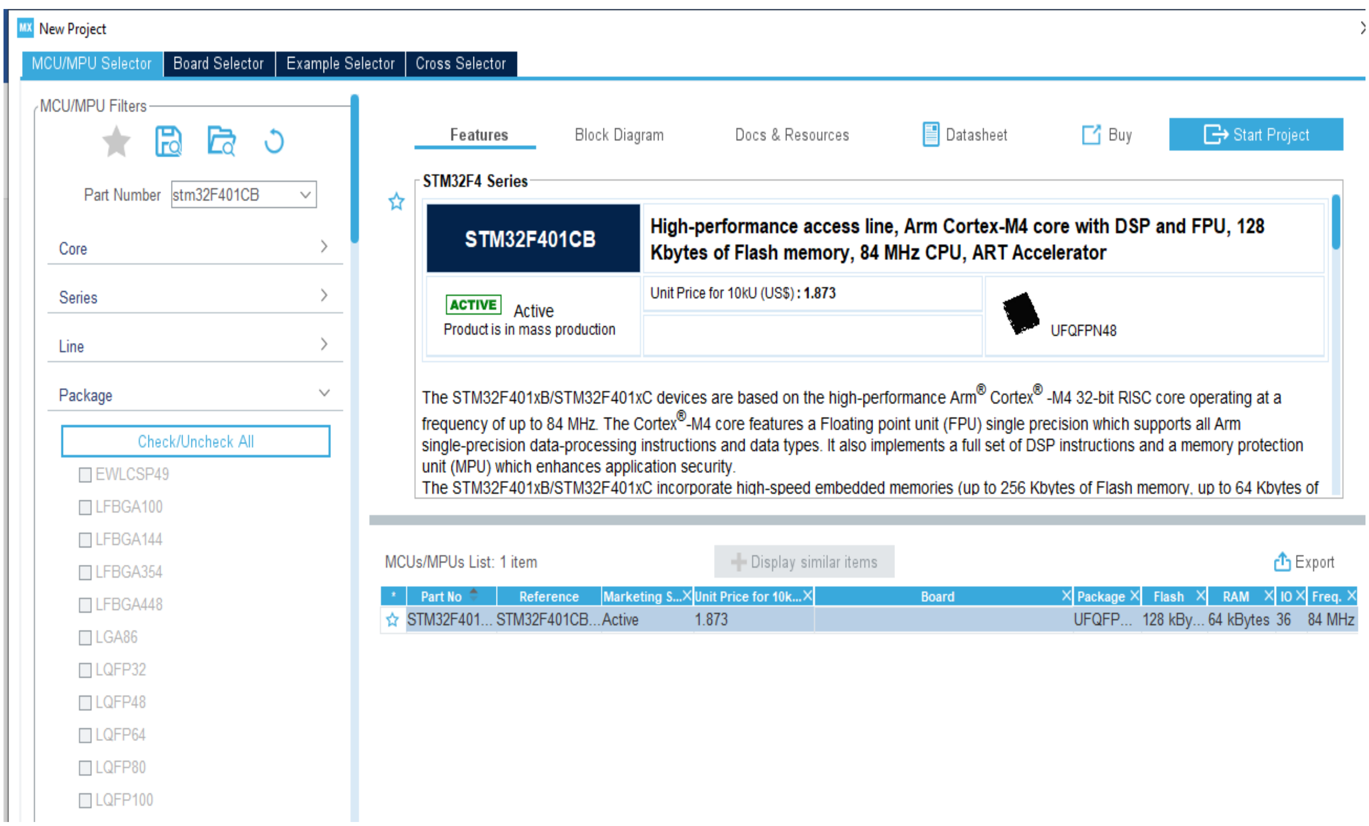
In this report we will explain all the building blocks of Obstacle sensitive car. We will look at the advantages of using the STM32cubeMx software. We will discuss briefly about each part of the project and how we use and interface them with the microcontroller. We will be using the KEIL to build the logic and test it using the Proteus software as a simulator. The Proteus help us to test our code working before deploying it to the original Hardware. It is easy and very useful for debugging the code but there are certain limitations that we will discuss later on. After our code is finished we will build a hardware step by step and then test our finalized code over the hardware.

Body

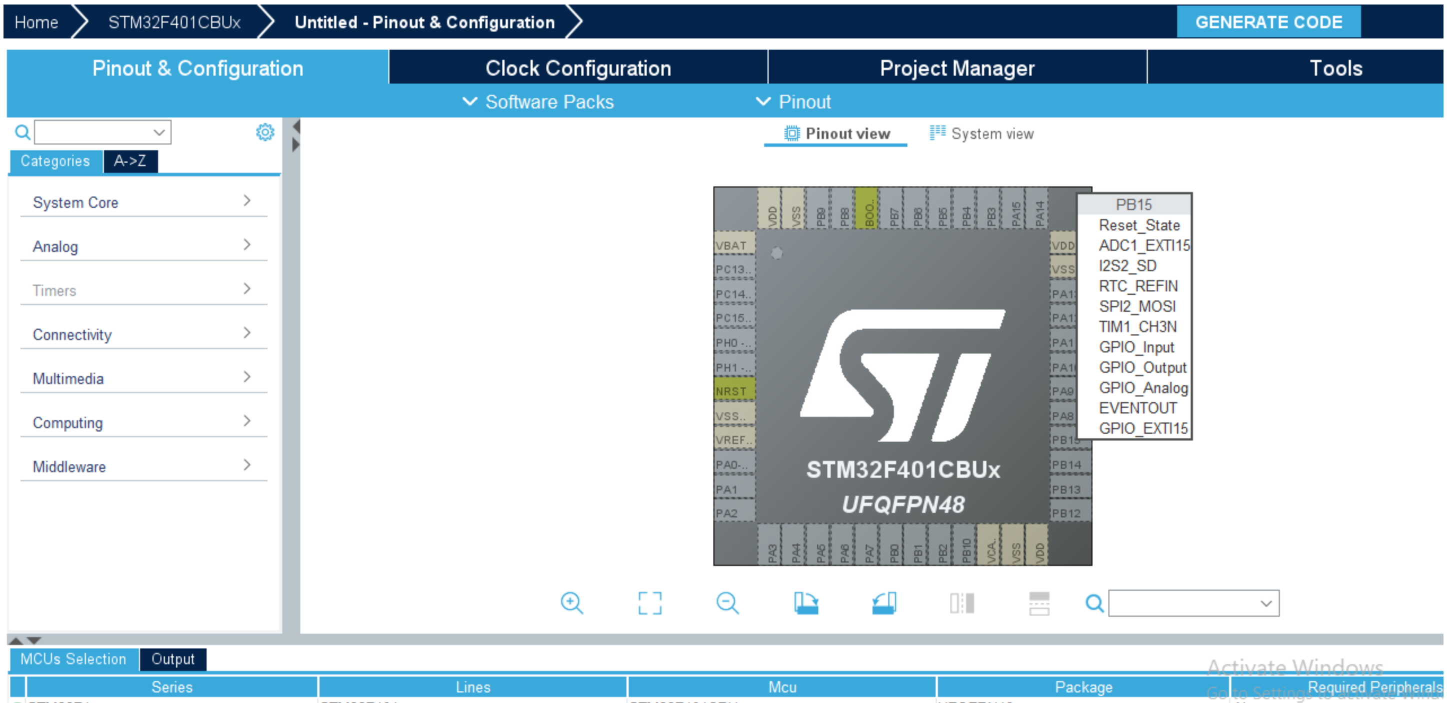
STM32CubeMx:

This software helps us to configure the Ports/Pins of Microcontroller easily without getting into the difficulty of looking at the datasheet and manually typing all the code for configuring the timers, GPIOs, Interrupts etc. It provides us with a user friendly environment in which we can easily set all the parameters that we find necessary for our project.

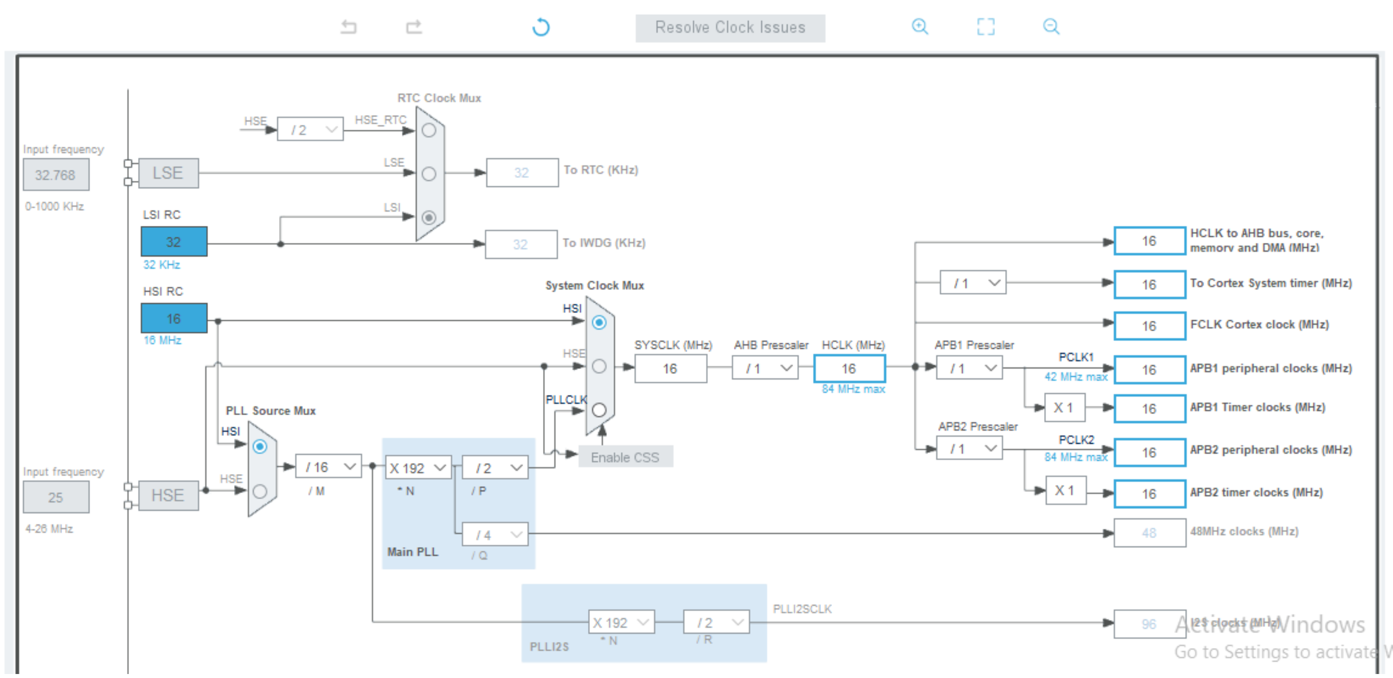
First we will have to select the microcontroller board that we are using:



Then we can start working out on it. We can set the clock in clock configuration menu, also we can set the PINOUT configuration also we can select the alternate functionalities like UART, I2C, Timers and others all with just few clicks and parameter settings. We will see in how to set all this in next sections of the report.

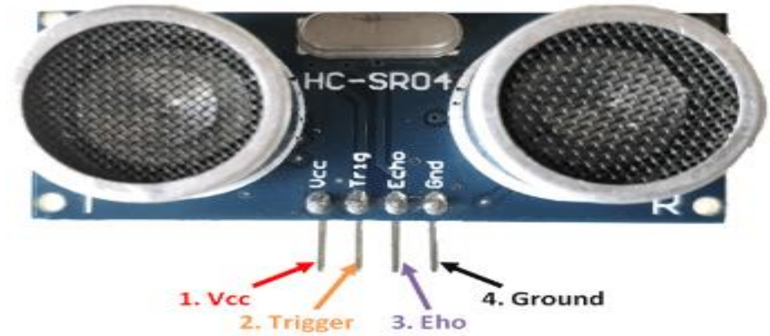


Clock Configuration can be done through this.



HCSR04 Ultrasonic Sensor:

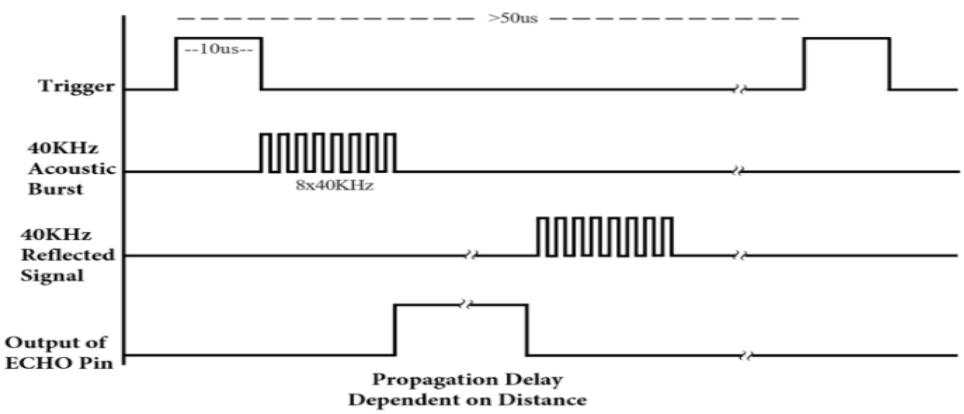
This Ultrasonic sensor uses the sound waves to measure the distance of object from the sensor.



Ultrasonic Sensor HC SR04 Pin Diagram

1. **Working:**

   We will have to give a 10us pulse to the trigger pin which will generate the acoustic signals from the transmitter then after reflecting from the object the receiver will receive the sound signals and corresponding to that the sensor will generate a pulse signal at the Echo Pin.



After that the Distance can be measured by the following formula

$$distance = \frac{time\ taken\ x\ speed\ of\ sound}{2}$$

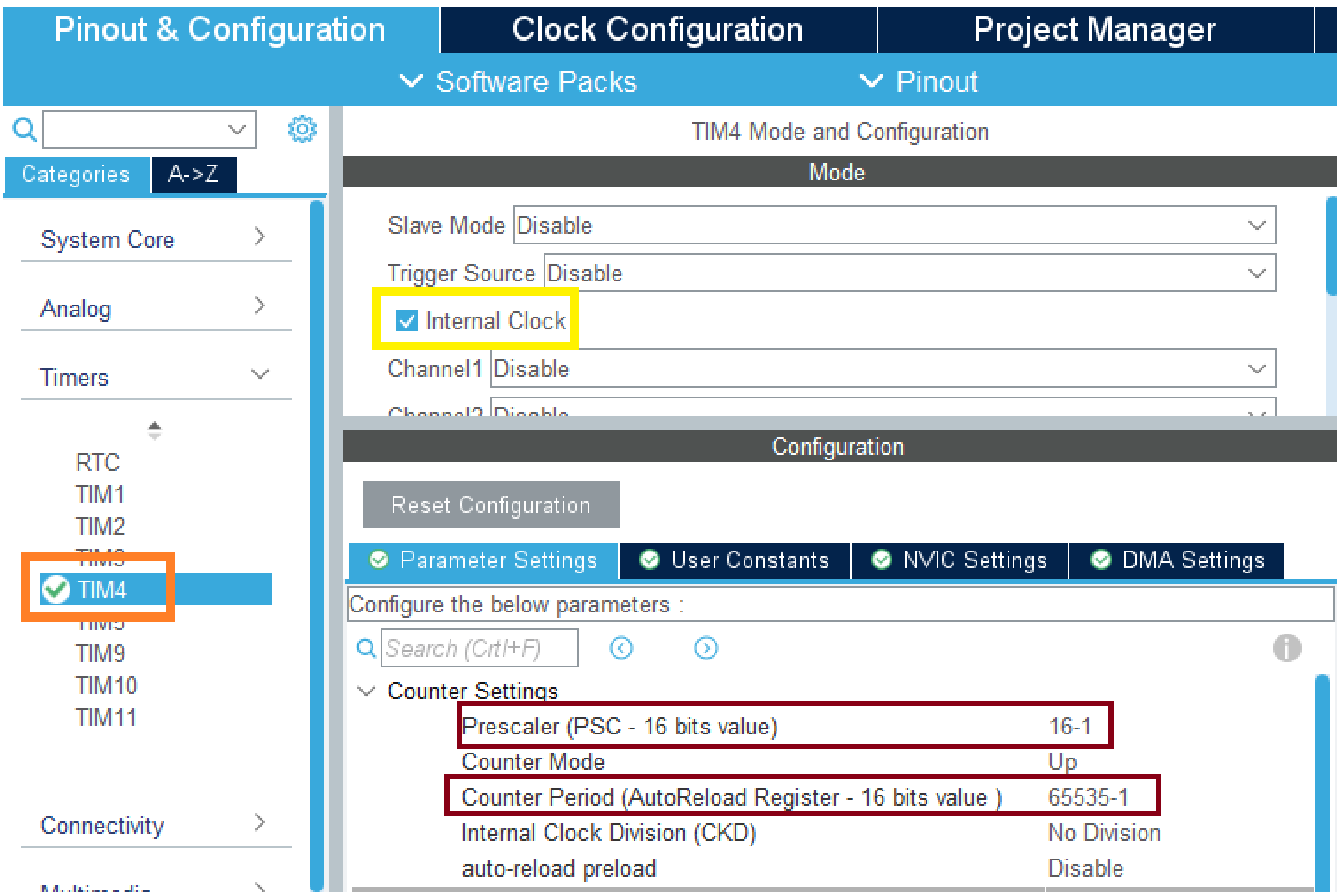
2. Interfacing the sensor with Microcontroller:

We have to provide the TRIG for 10us and then read the ECHO pin signal. Since we do not have the library to provide the 10us delay so we will use the Timer and built our own US delay function. We will configure one pin as an output which will be connected to the trigger pin of sensor to make it to logic High in the code for 10us. Furthermore, we will use a timer as an input capture device to measure the time stamp of the echo pulse signal to calculate the distance.

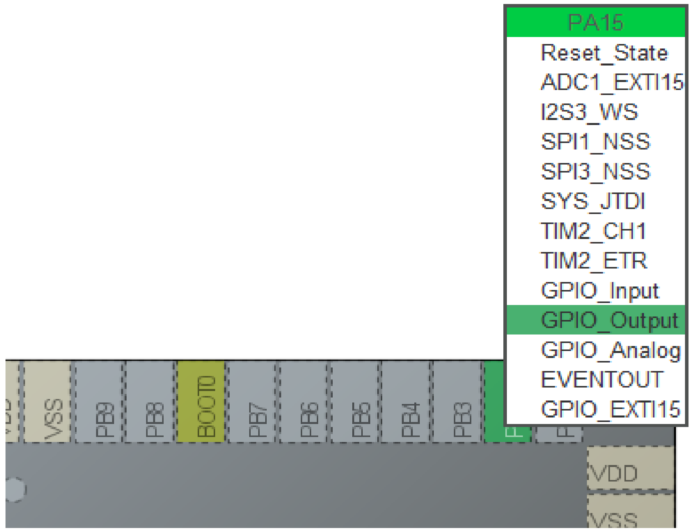
a. Microsecond Delay Configuration and Setting the Trigger pin

In this we will configure a timer for the frequency 1MHz for which the clock will be 1us so if we run the counter its each ticks will be equal to 1us and hence in code run the counter for 10 ticks which will be equal to the 10us delay. Following will be the setting that should be done in the Stm32CubeMx:

- Selecting the TIM4 (timer 4) and enabling its clock
- Set the Pre-scaler to 16, or depending on at which clock frequency our Microcontroller is working, in our case it is 16MHz so we select (16-1 i.e. equal 16 in original due to start from 0) Pre-scaler. Hence, 16MHz/16 = 1MHz.
- Setting the Auto-Reload-Register Counter period value to its maximum possible i.e. 65535-1.



- Next we set the PA15 pin of Microcontroller to the GPIO output and set its label as the "TRIG".



that is all now generate the code of KEIL for building logic and testing the code.

I. CODING PART:

The following function will provide the desired delay. We can pass any positive integer value to this function and it will give us that delay.

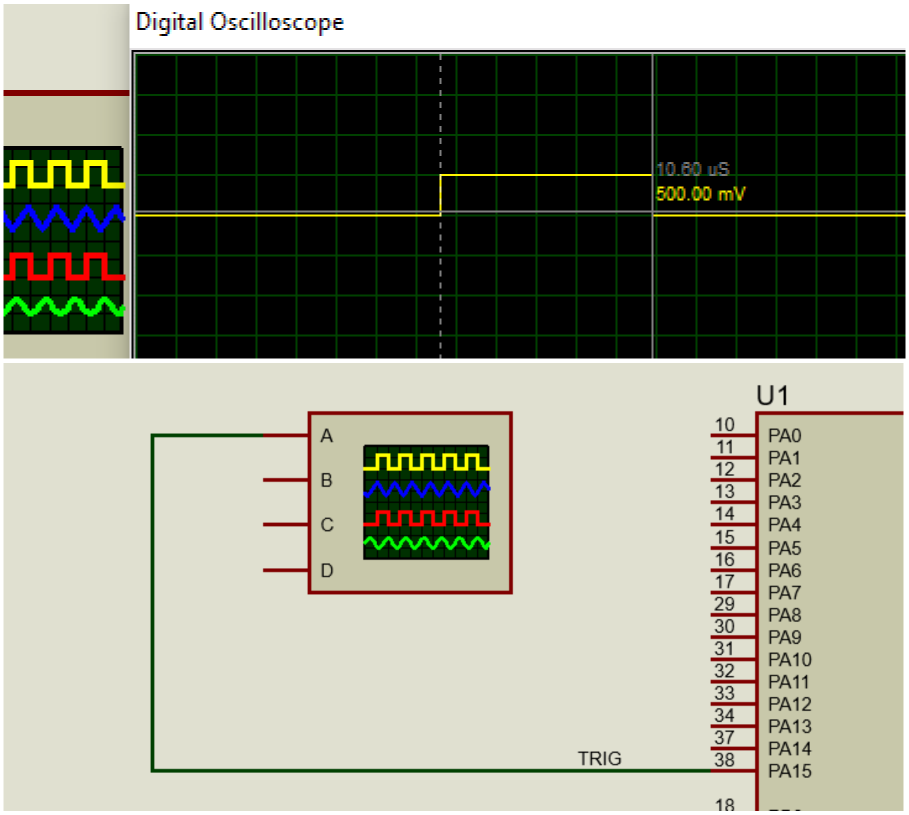
```
// Microsecond delay function
void usDelay(uint16_t time)
{
    __HAL_TIM_SET_COUNTER(&htim4,0);  // set the counter value a 0
  while (__HAL_TIM_GET_COUNTER(&htim4) < time);  // wait for the counter to reach the time input in the parameter
}
```

In the main we will test the delay function by setting the TRIG pin high for 10us.

```
HAL_TIM_Base_Start(&htim4); //Initialize the Timer 4
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* The following code will test the Microsecond Delay Function
  in this code the TRIG pin will be set High for 10us, we can check the code by simulating it on proteus
  PA15->GPIO output->TRIG
  TIM4-> At 1MHz

  code: we will get 10us pulse at TRIG pin(PA15) for every 500ms

  */
  HAL_GPIO_WritePin(TRIG_GPIO_Port,TRIG_Pin,GPIO_PIN_SET); //Set the TRIG Pin High
  usDelay(10);//Delay of 10us
  HAL_GPIO_WritePin(TRIG_GPIO_Port,TRIG_Pin,GPIO_PIN_RESET); //Set the TRIG pin low
  HAL_Delay(500); // Delay of 500ms
```

II. Simulation Part

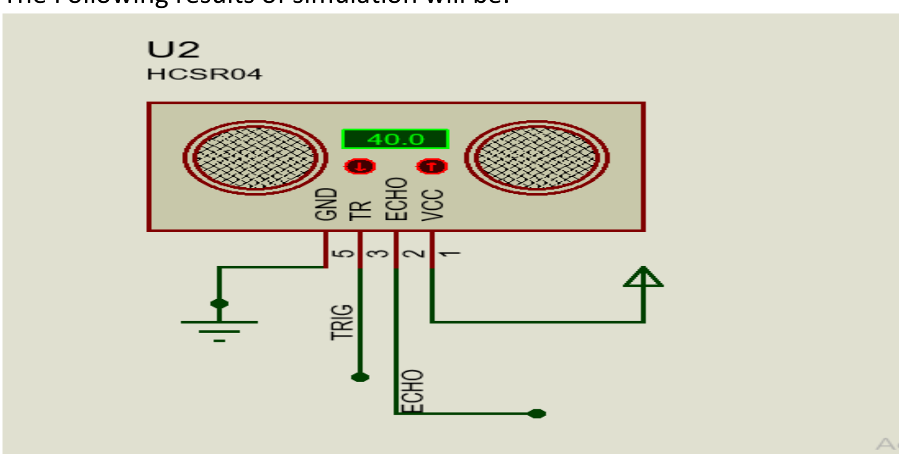
In simulation we are receiving the approximately 10us pulse output at the TRIG pin

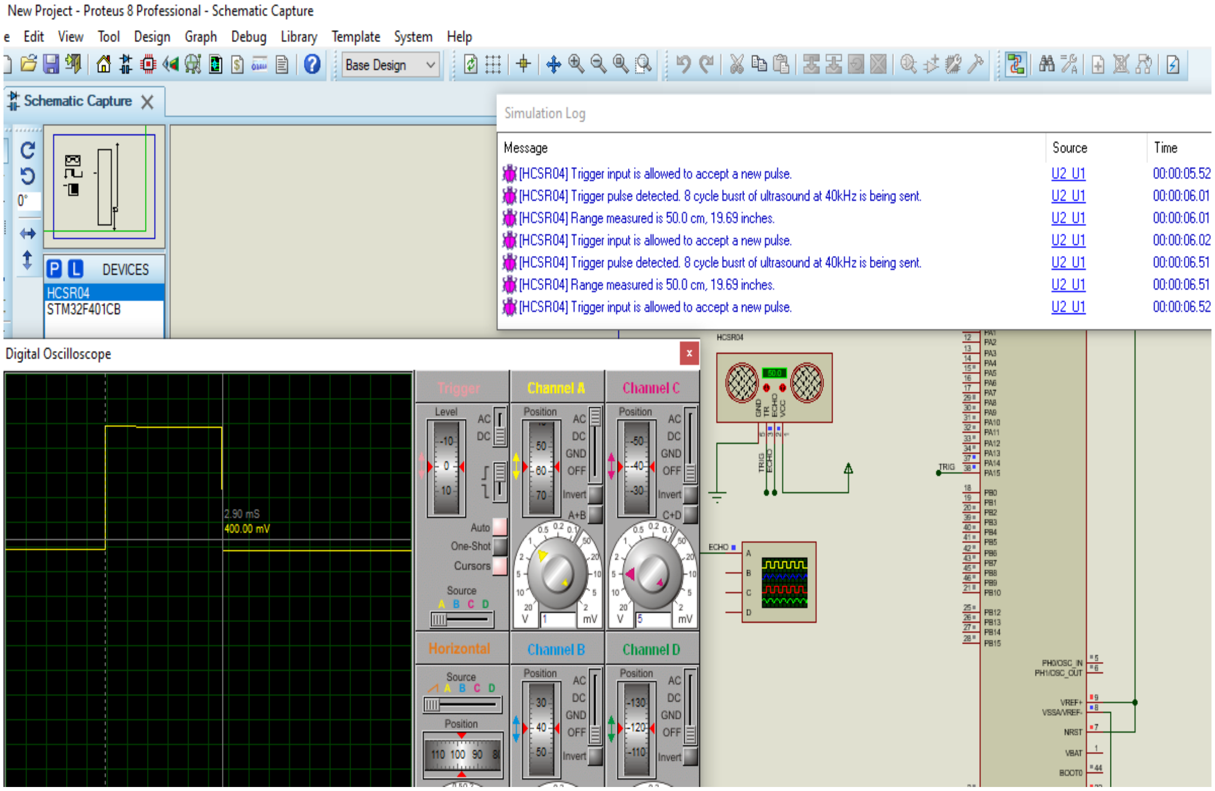


b. Timer as an Input Capture device

if we provide this trigger pulse to the HCSR04 sensor it will give us the ECHO pulse corresponding to the distance it measures.

The Following results of simulation will be:





Note that we are receiving the exact 2.90ms of pulse duration which corresponds to 50cm distance if we divide it with 58us we will get our desired result.

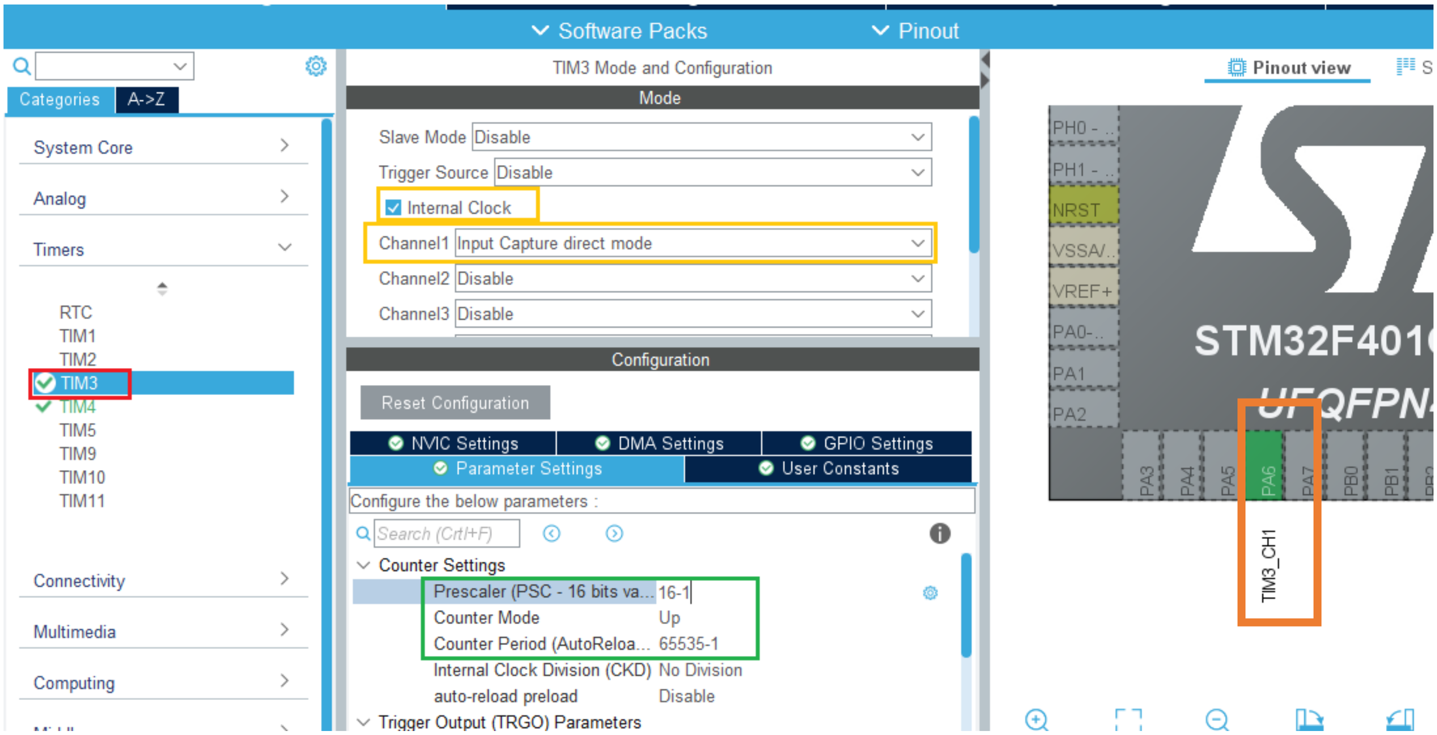
Now we will have to configure and used the Microcontroller timer for measuring this pulse time to calculate the distance. For this purpose, we use the timer as an input capture device.

PULSE DURATION MEASUREMENT:

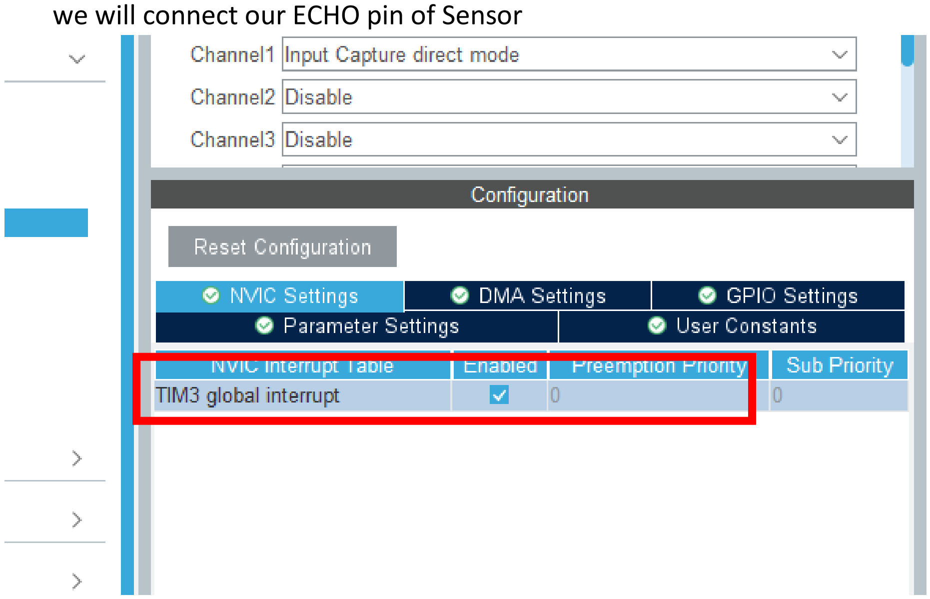
The timer input capture mode can be used for measuring pulse durations. The pulse duration can be defined as the time interval between a rising (low to high transition) edge followed by a falling edge. In Our method, timer is configured to capture event time on a rising edge and generate an interrupt. While processing the timer interrupt for rising edge event, the timer is reconfigured for falling edge event. In addition, the edge time is recorded for further processing. Now the event time is captured again on the occurrence of a falling edge. Subtracting the event time corresponding to first edge from that of second edge, we can determine the pulse duration

Following will be the setting of the STM32CubeMx:

- We use the Timer 3 and provide it clock and select the Channel 1 as an Input Capture Direct mode.
- Setting the Pre-scale to 16-1 according to clock and Counter period(ARR) to its Max value i.e. 65535-1. You can see that PA6 will be marked as TIM3_CH1.



- After that we will have to enable the global Interrupt because as the rising edge is detected at the Timer 3 input then an interrupt will be generated. We will have the Timer 3 Channel 1 at PA6 where we will connect our ECHO pin of Sensor



I. Coding Part

We have written the callback function which will be executed on the generation of interrupt and this function does the functionality as described above in the pulse duration measurement section.

```c
//This function will be executed when the input capture interuupt is generated i.e when there is a pulse at the echo pin
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)  // if the interrupt source is channel1
  {
    if (Is_First_Captured==0) // if the first value is not captured
    {
      First_Value = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1); // read the first value
      Is_First_Captured = 1;  // set the first captured as true
      // Now change the polarity to falling edge
      __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);
    }

    else if (Is_First_Captured==1)   // if the first is already captured
    {
      Second_Value = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);  // read second value
      __HAL_TIM_SET_COUNTER(htim, 0);  // reset the counter

      if (Second_Value > First_Value)
      {
        Difference = Second_Value-First_Value;
      }

      else if (First_Value > Second_Value)
      {
        Difference = (0xffff - First_Value) + Second_Value;
      }

      Distance = Difference * .034/2;
      Is_First_Captured = 0; // set it back to false

      // set polarity to rising edge
      __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);
      __HAL_TIM_DISABLE_IT(&htim3, TIM_IT_CC1);
    }
  }
}
```

Then for activating the Input capture mode and Trigger the Sensor for ECHO we have written another function so that we do not have to enable and trigger it manually all the time instead just calling the function

```c
void Dist_Measure()
{
  HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_SET);  // pull the TRIG pin HIGH
  usDelay(10);  // wait for 10 us
  HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_RESET);  // pull the TRIG pin low

  __HAL_TIM_ENABLE_IT(&htim3, TIM_IT_CC1);  //Enables the input capture
}
```

Before the while following things must be done

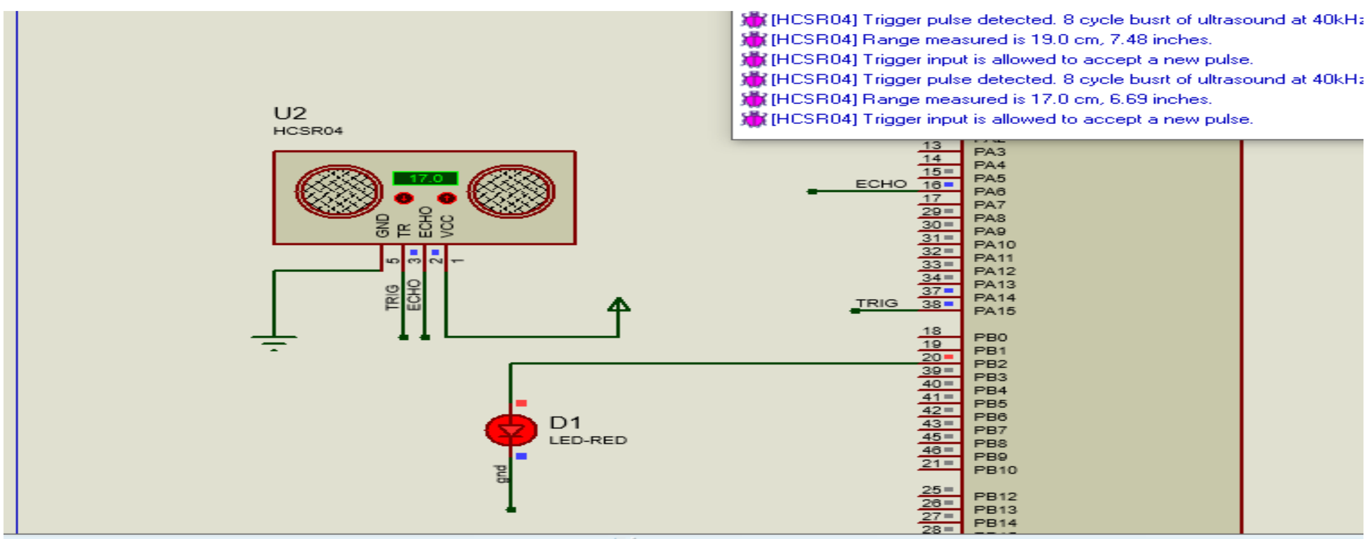
```
HAL_TIM_Base_Start(&htim4); //Initialize the Timer 4

HAL_TIM_IC_Start(&htim3,TIM_CHANNEL_1); //start the input capture mode at timer 3 channel 1 i.e PA6 pin of Microcontroller

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
```

In order to test the working of our sensor and Pulse duration measurement code we have written the following test code to check. In addition, we have declared the LED at PB2.

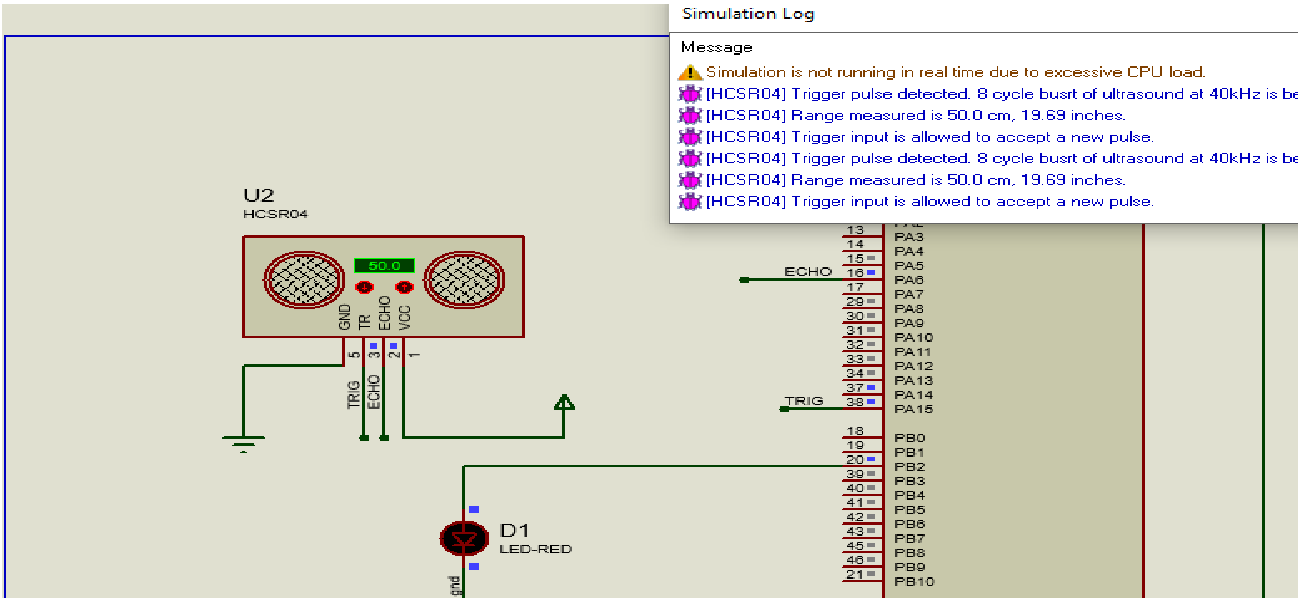
```
/*
Logic to Test the measured value of distance
If the Distance is less then 20cm the Red LED at PB2 will be ON
Marking that there is an obstacle close
*/
Dist_Measure();
if(Distance <20 )
{
  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);
  HAL_Delay(100);
}
else
{
  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET);
  HAL_Delay(100);
}
HAL_Delay(500);
```

II. Simulation Part

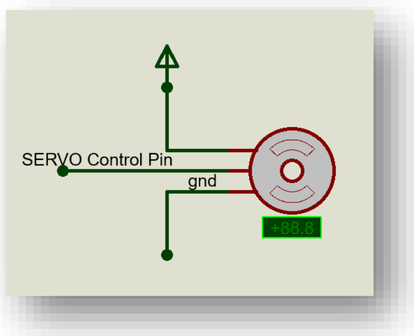
As the distance is less than 20cm the led gets On.



As the distance is greater than 20cm the LED turns off



SERVO Motor Control:



i) Working

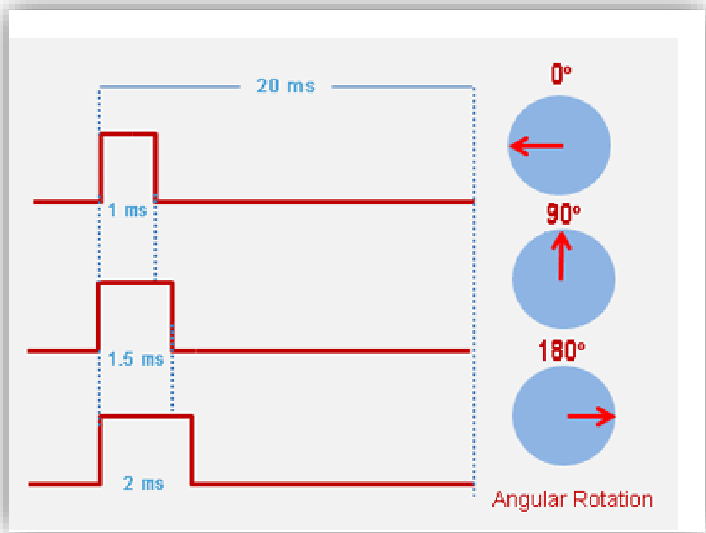
Servo motor is most commonly used in motion and position control applications, in our project we used a 180-degree servo motor to control the direction of the sensor which means that:

Straight-> 90 degree

Right -> 180 degree

Left ->0 degree

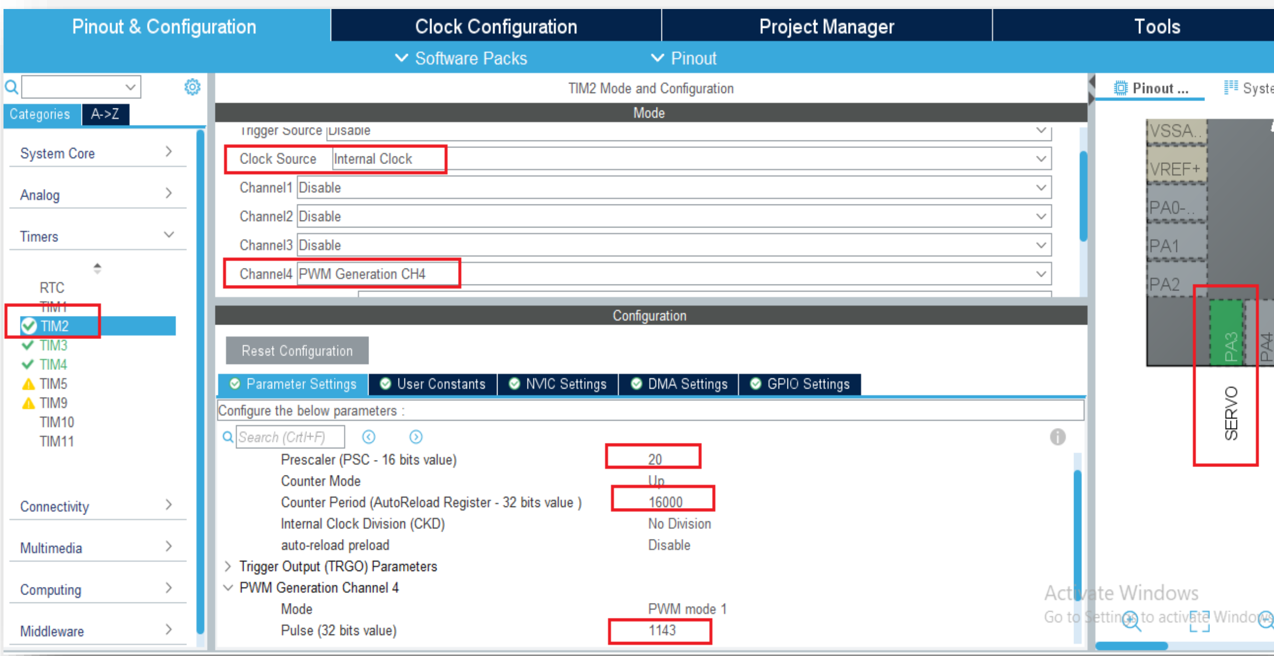
Servo motor is controlled by PWM (Pulse with Modulation) which would be provided by the Microcontroller to the servo Control Pin. There is a minimum pulse 0.5ms, a maximum pulse 2ms and a repetition rate. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.



ii) Interfacing with Microcontroller (PWM generation)

In order to generate a PWM signal from the microcontroller we use the Timer

Following will be the setting of Stm32CubeMx



The total PWM frequency will be given by

$$f_{pwm} = \frac{f_{clk}}{prescale\ X\ Counter\ Period}$$

In our case 16/20x16000 = 50Hz = 20ms

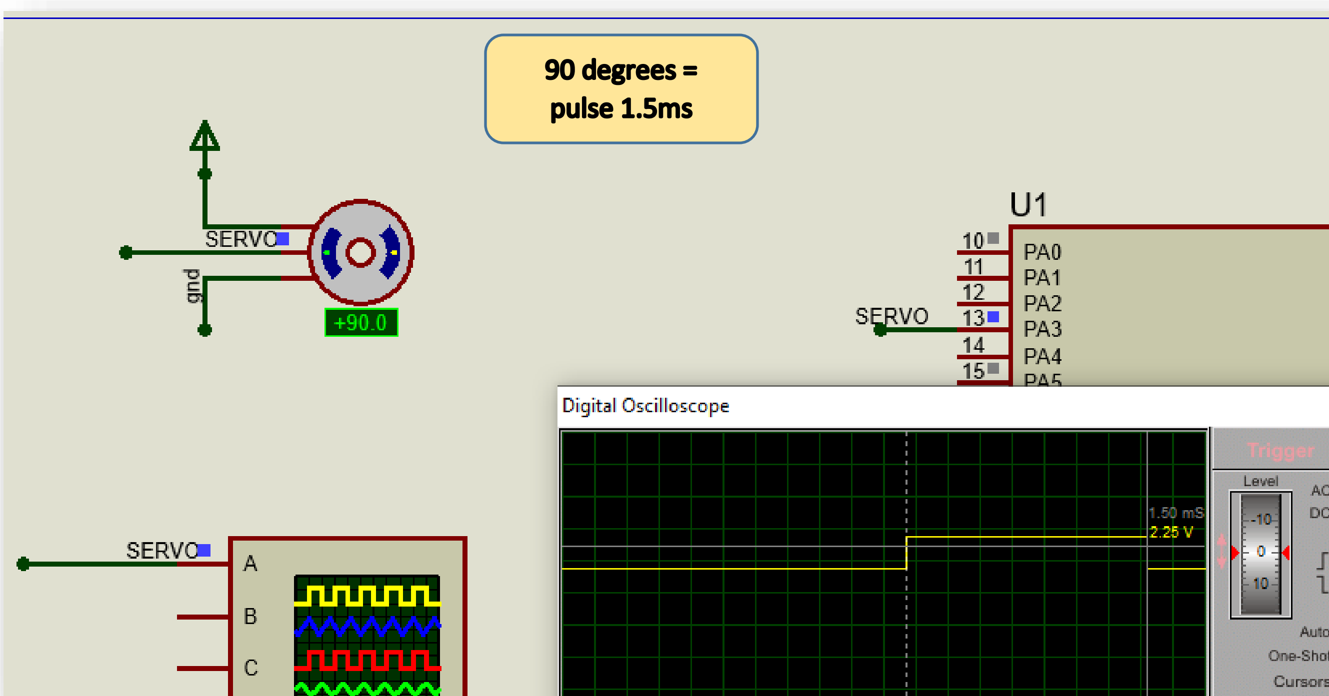
a. Coding Part

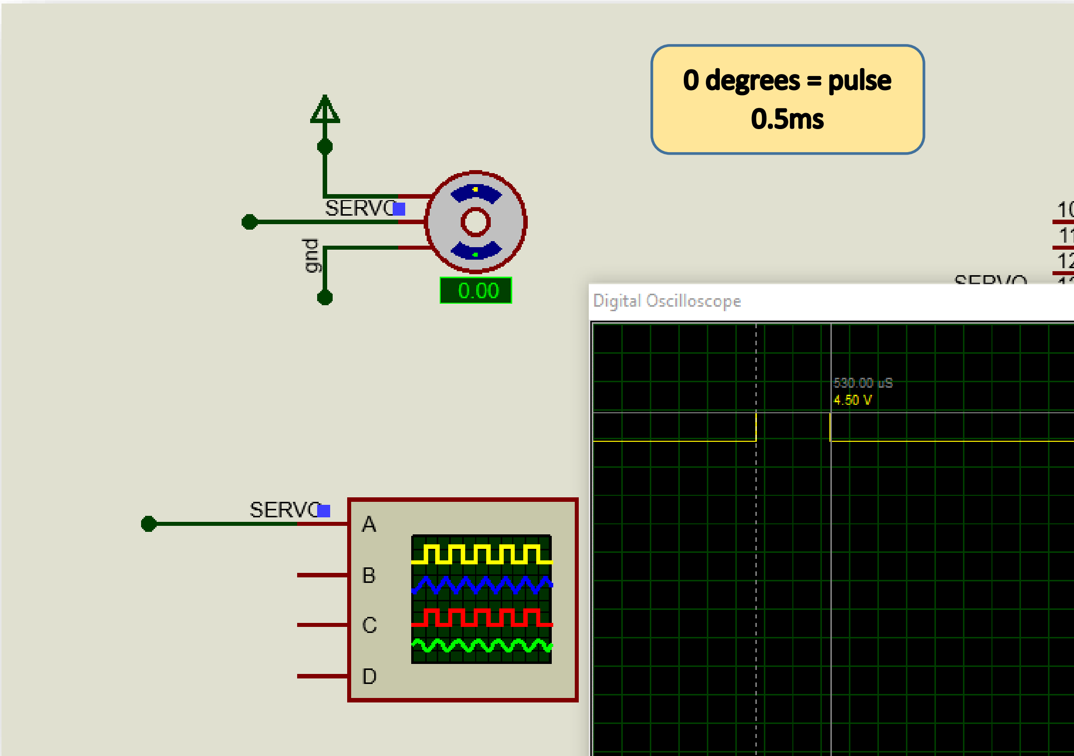
Before while loop in order to start PWM generation we have to write the command

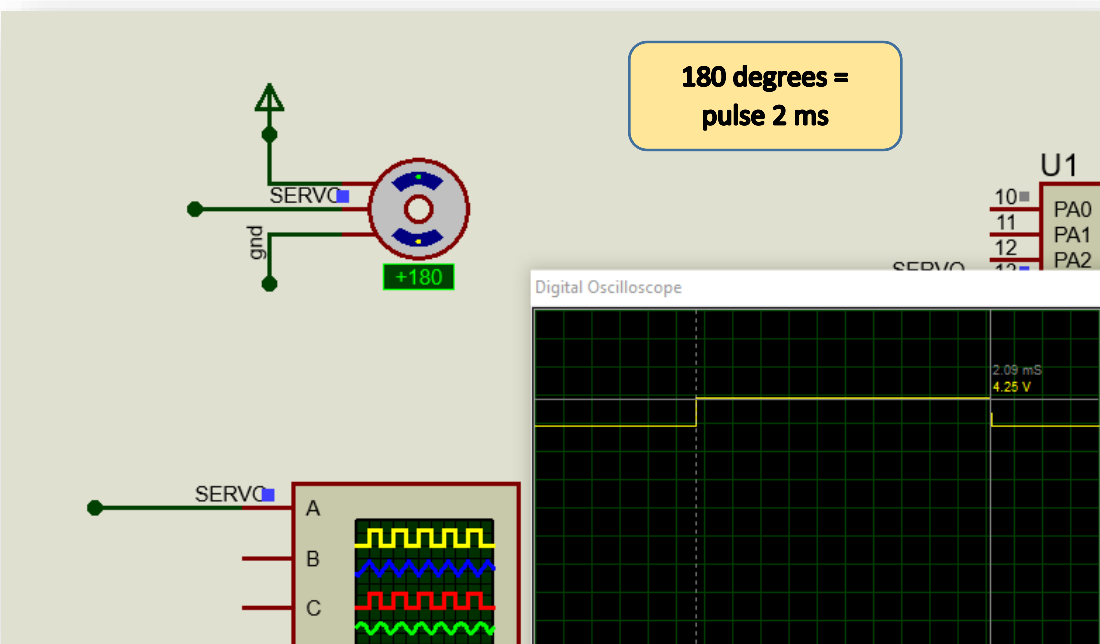
HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_4) //we are using Timer 2 Channel 4 for PWM

```
The Following code is to test the servo motor working
the value of 1143 = 1.5ms pulse  90 degree angle
the value of 400 = 0.5ms pulse    0 degree angle
the value of 1600 = 2ms pulse    180 degree angle

First terminal is VCC connected to power label in simulation or source 5v in hardware
Second terminal is Servo Control pin which will be connected to PA3 of Microcontroller
Third terminal is GND  connected to the ground.

the following code will turn the servo to different angles after every 2 seconds

*/

htim2.Instance->CCR4 = 1143;  // Moves the servo to 90 degree
HAL_Delay(2000);
htim2.Instance->CCR4 = 400;  // Moves the servo to 0 degree
HAL_Delay(2000);
htim2.Instance->CCR4 = 1600; // Moves the servo to 180 degree
HAL_Delay(2000);
```

b. Simulation Part

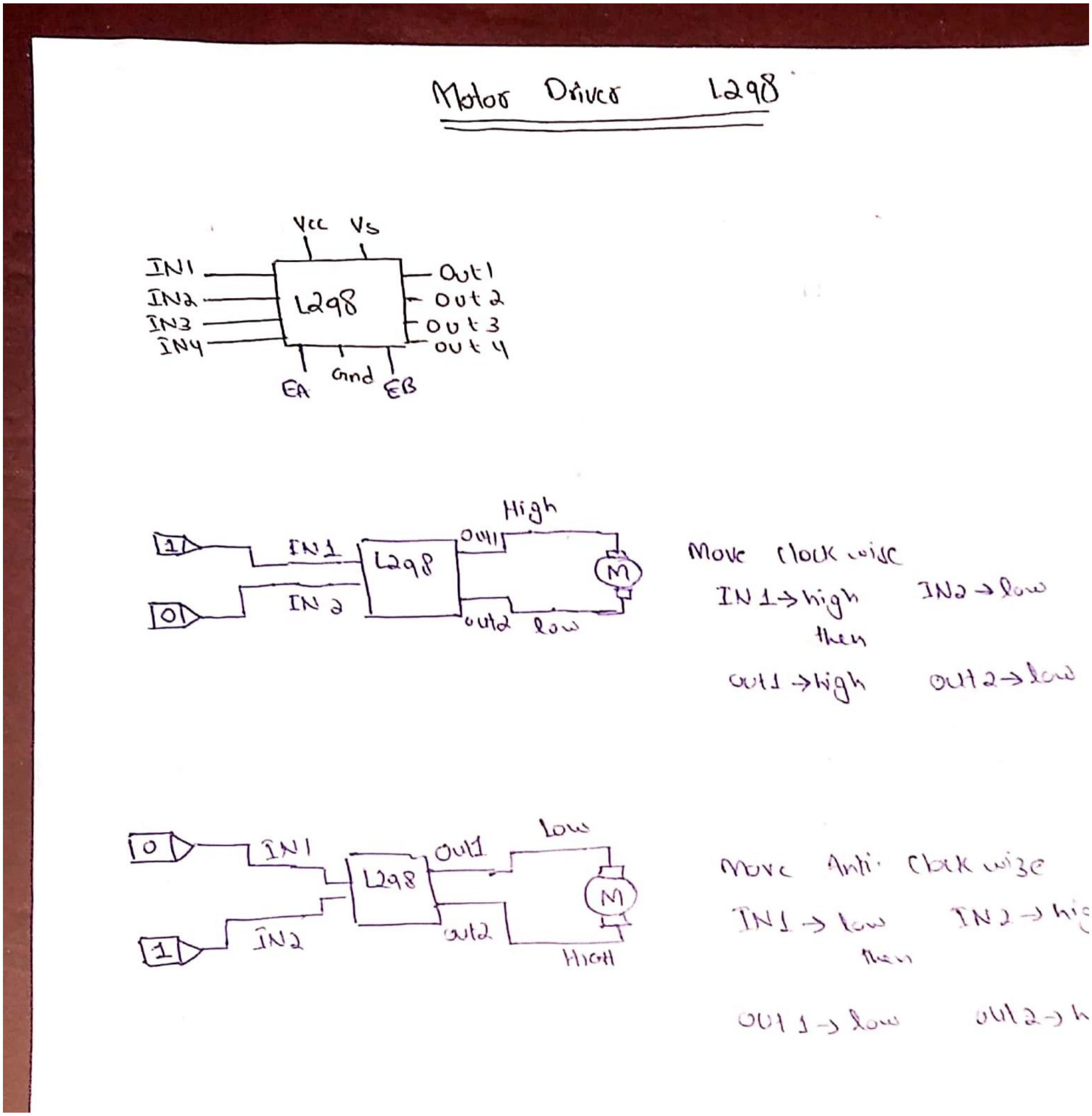



0 degrees = pulse 0.5ms
SERVO
gnd
0.00
SERVO
A
B
C
D
Digital Oscilloscope
530.00 uS
4.50 V

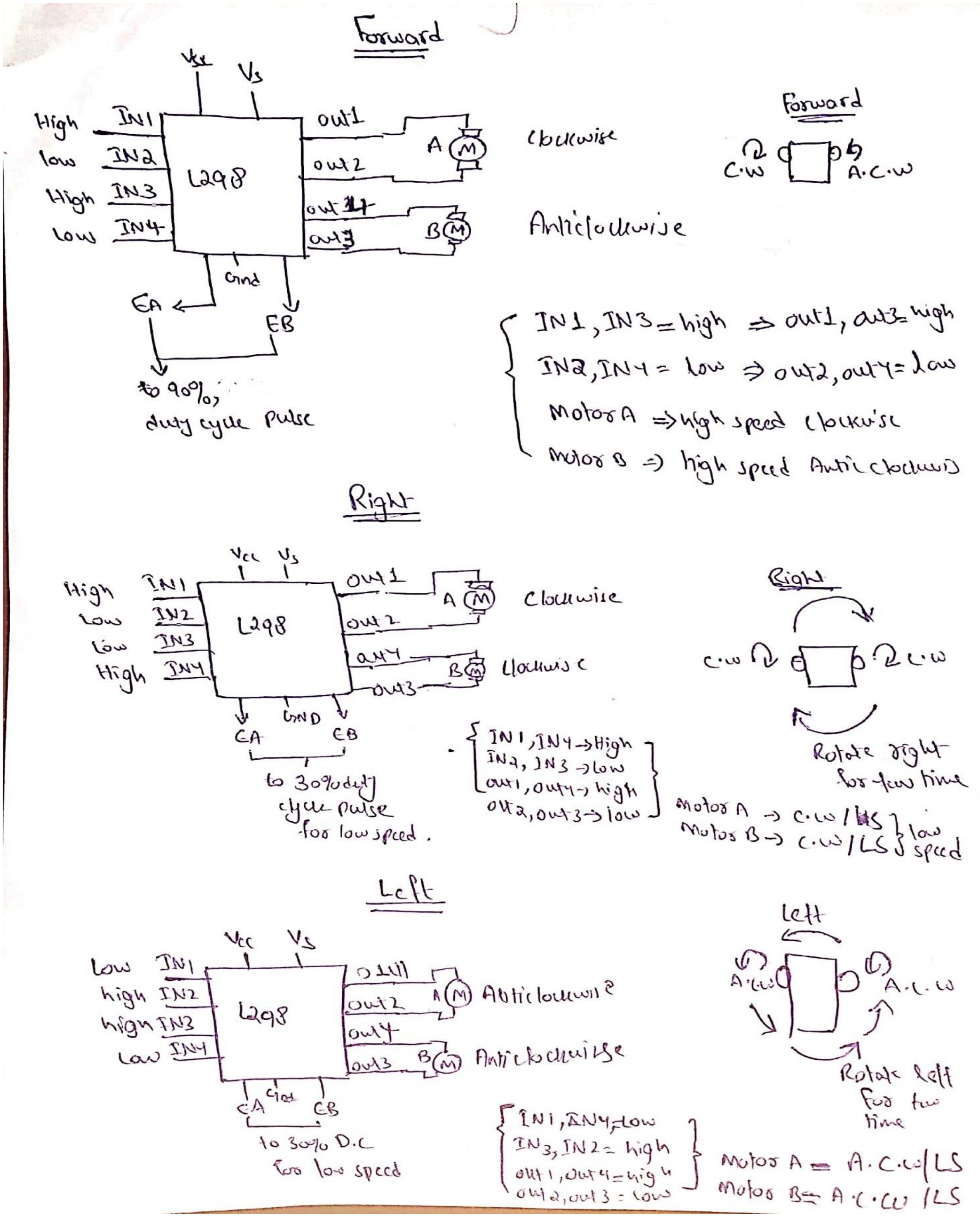

180 degrees = pulse 2 ms
SERVO
gnd
+180
U1
PA0
PA1
PA2
SERVO
A
B
C
Digital Oscilloscope
2.09 mS
4.25 V

L298 Motor Driver – Motor Direction and Speed Control

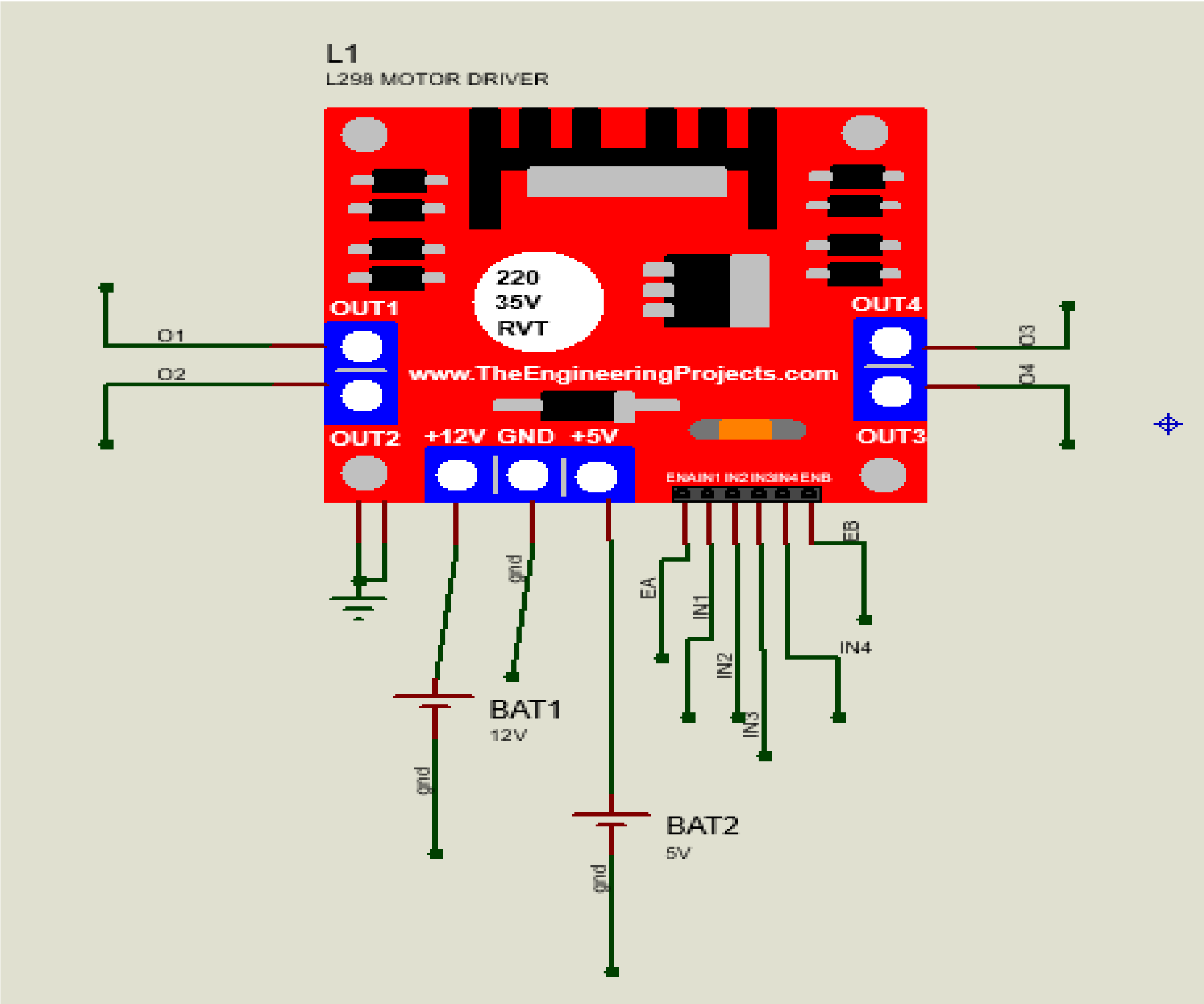
Following Image explain the working of Motor Driver



The Function for motion control will be made according to the following logic



The speed control will be explained in the next section in short we will be changing the PWM signal by changing the value of CCR register on runtime.

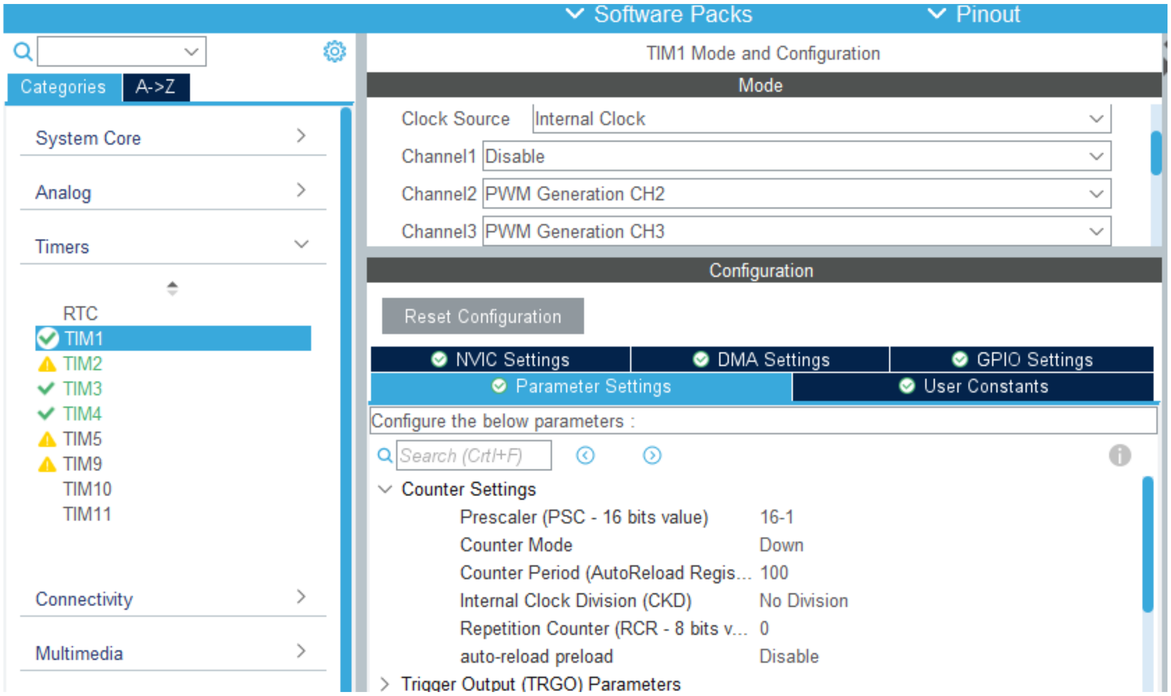


Interfacing with Microcontroller

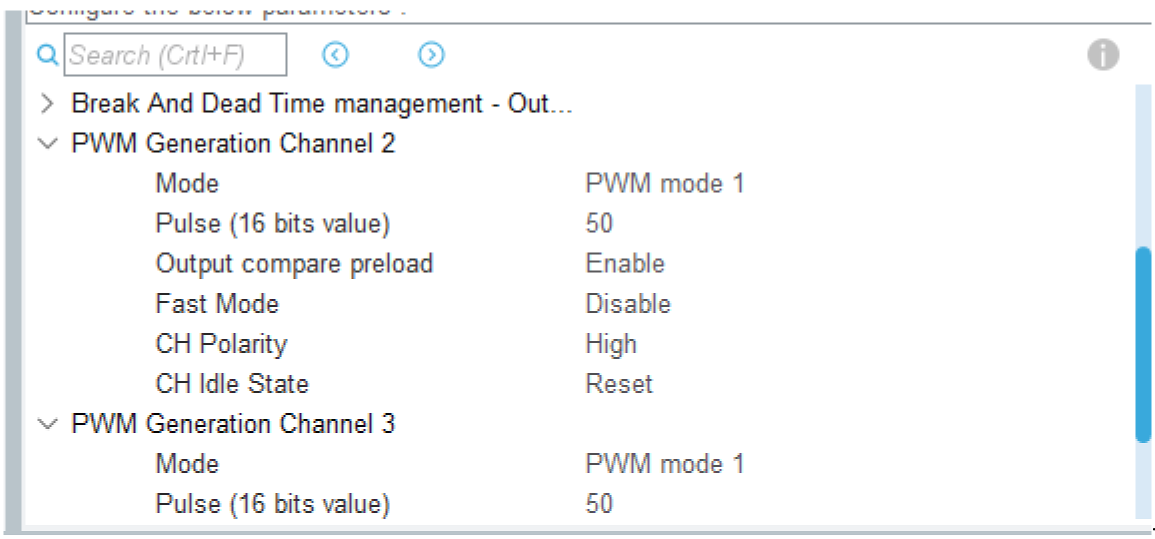
For interfacing the L298 we will have to configure the GPIOs as Output for Input pins IN1 to IN4 and We will configure the Timers for PWM signal for the Enables to control the speed of the motors.

Following setting of the Stm32CubeMx is to be done:

- Selecting the Timer 1 channel 2 and 3 for PWM generation for EA and EB
- The frequency will be 10KHz since Pre-scaler is 16 and Counter period is 100 then the frequency of the timer will be 16M/16x100 = 10KHz.

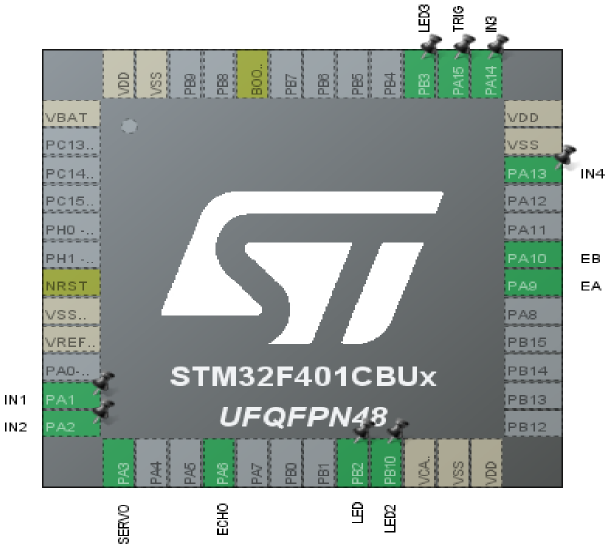


Since the Counter Period is 100 which means that we will have the 50% pulse at Pulse value 50.



7

By changing the value in Capture Compare Register (CCR1 and CCR2) we can vary the PWM signal duty cycle.



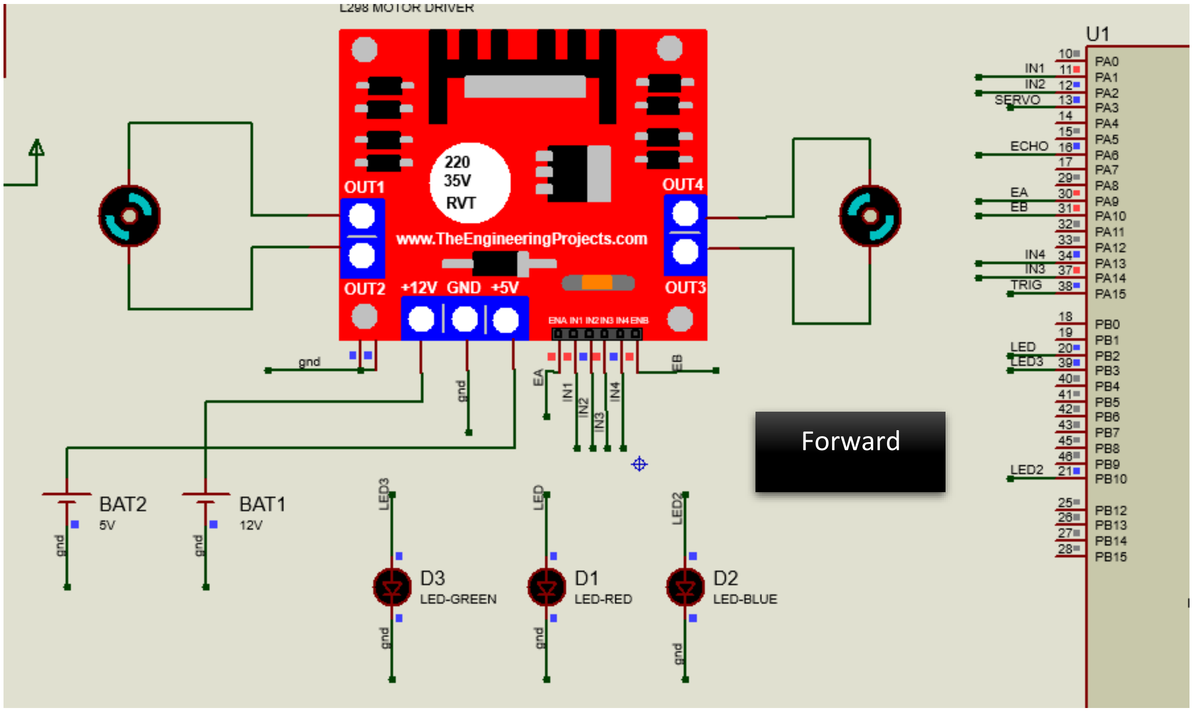
This will be the Pin out configuration of Microcontroller so far. We used the PA13, PA14 and PA1, PA2 for the L298 Motor Driver Inputs. Furthermore, the PIN outputs are also used for the LEDs for checking the status of the motors. The LEDs and motor relation is given in the coding part.

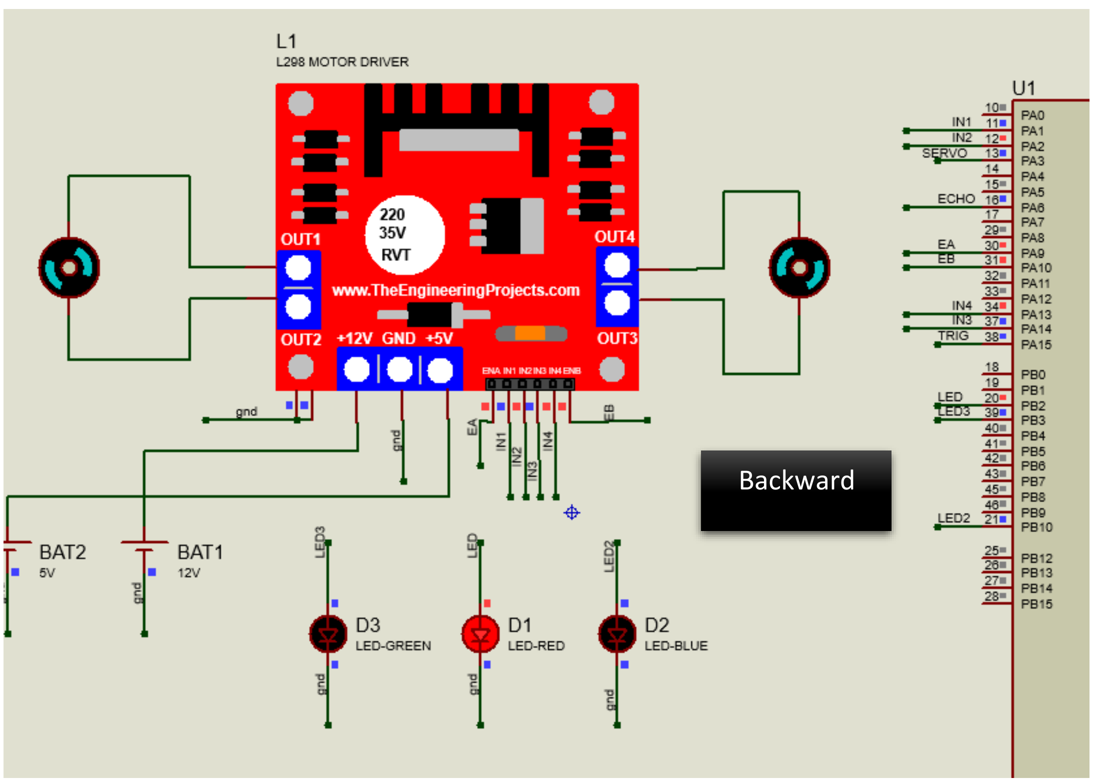
a. **Coding Part**

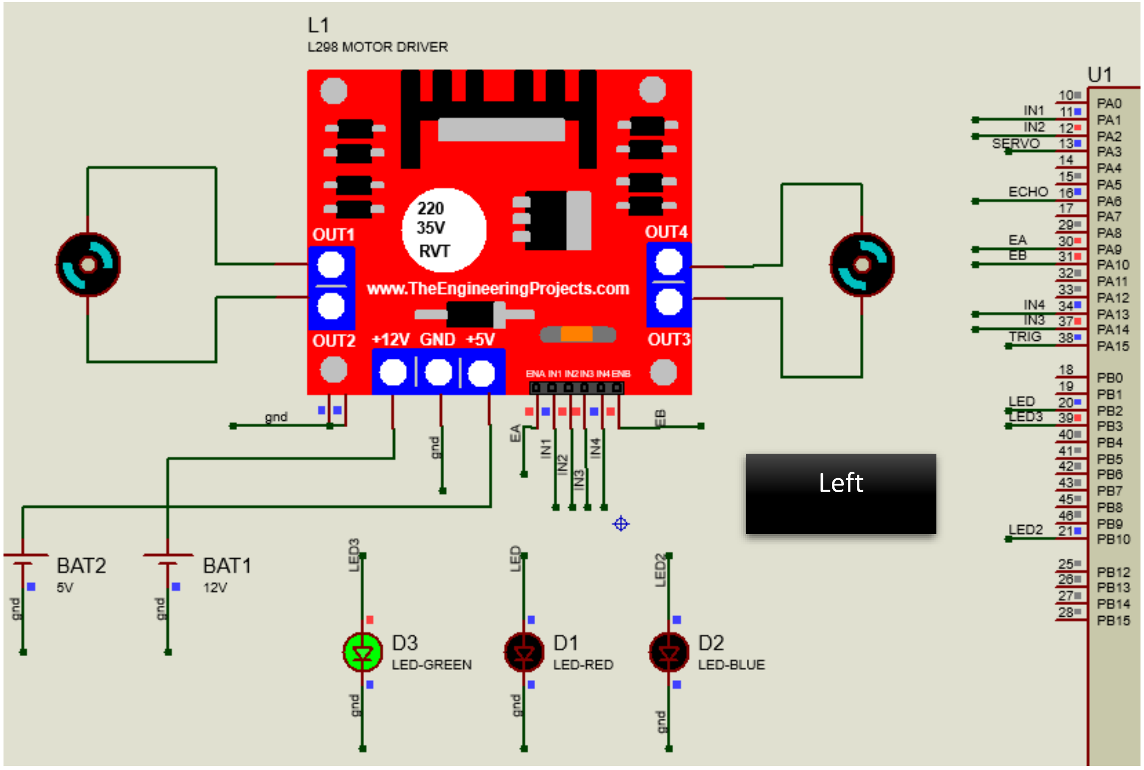
The following code was run as a test to check the direction control of the motor. The functions are given in the appendix section.

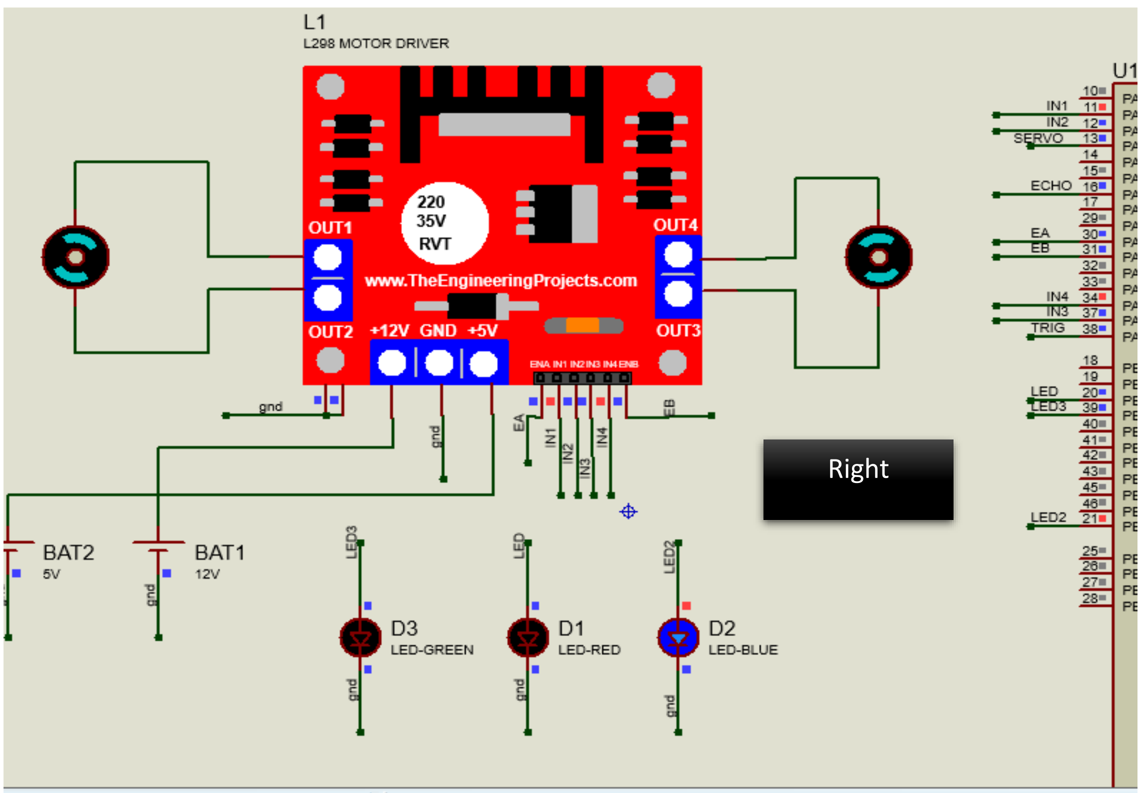
```
/*
The following code is to test the L298 motor driver
in this code the first the motor will move in forward direction
then turn left and then right and then backward all of this will be indicated by the LEDs
The fucntion will be given in the appendix section of the report.
LED-Red => BACKWARD => at PB2
LED-Blue => Right => at PB10
LED-Green=> Left => at PB3
all off for forward
EA at PA9   EB at PA10
IN1=> PA1
IN2=> PA2
IN3=> PA13
IN4=> PA14
*/
HAL_Delay(500);
Forward();
HAL_Delay(500);
HAL_GPIO_WritePin(LED3_GPIO_Port,LED3_Pin,GPIO_PIN_SET);
Left();
HAL_Delay(500);
HAL_GPIO_WritePin(LED3_GPIO_Port,LED3_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_SET);
Right();
HAL_Delay(500);
HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);
Backward();
HAL_Delay(500);
HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET);
```

b. **Simulation part**





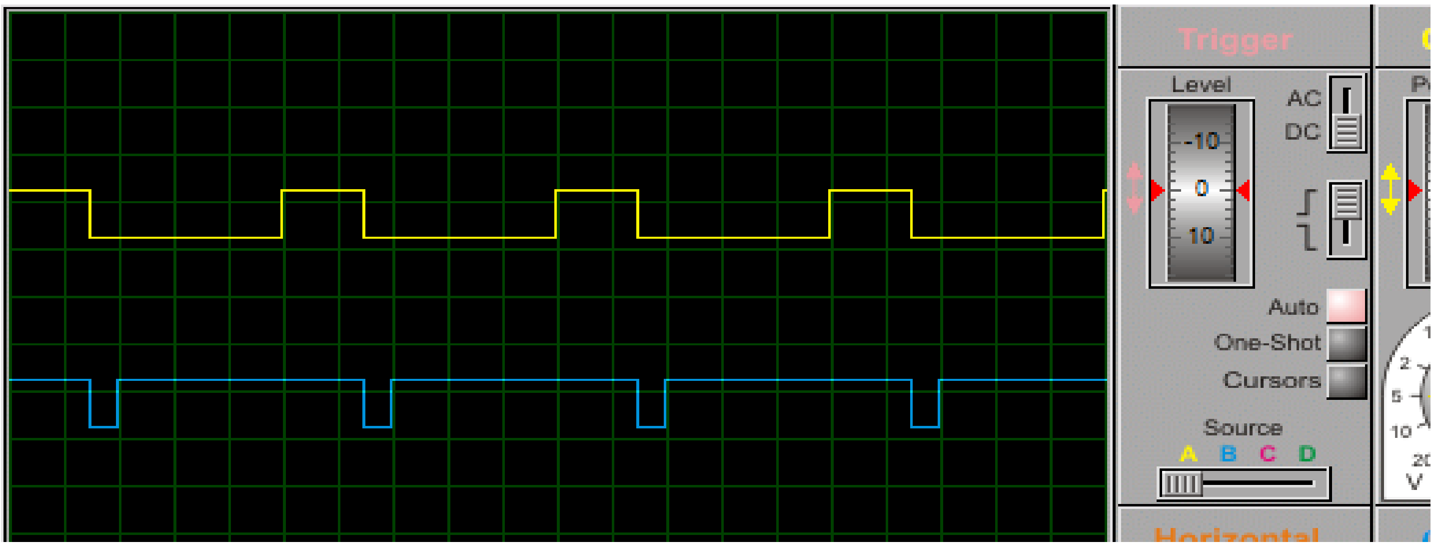

L1
L298 MOTOR DRIVER
220
35V
RVT
OUT1
OUT2
OUT3
OUT4
+12V GND +5V
www.TheEngineeringProjects.com
ENA IN1 IN2 IN3 IN4 ENB
U1
IN1
IN2
SERVO
ECHO
EA
EB
IN4
IN3
TRIG
LED
LED3
LED2
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB12
PB13
PB14
PB15
BAT2
5V
BAT1
12V
gnd
D3
LED-GREEN
D1
LED-RED
D2
LED-BLUE
Left


L1
L298 MOTOR DRIVER
220
35V
RVT
OUT1
OUT2
OUT3
OUT4
+12V GND +5V
www.TheEngineeringProjects.com
ENA IN1 IN2 IN3 IN4 ENB
U1
IN1
IN2
SERVO
ECHO
EA
EB
IN4
IN3
TRIG
LED
LED3
LED2
BAT2
5V
BAT1
12V
gnd
D3
LED-GREEN
D1
LED-RED
D2
LED-BLUE
Right

PWM Test for motor speed

```
  /* USER CODE END WHILE */

  /*
  Following code is to test the PWM signal duty cycle for Enabel pins of L298 motor driver
  EA at PA9
  EB at PA10

  in thsi code we will run the Motor A at low speed and Motor B with High speed by increasing the Dutycycle
  which in corespondence increase the average voltage and power.
  */

  htim1.Instance->CCR2 = 30;  // Less speed at EA
  htim1.Instance->CCR3 = 90; // High Speed at EB

  /* USER CODE BEGIN 3 */
```

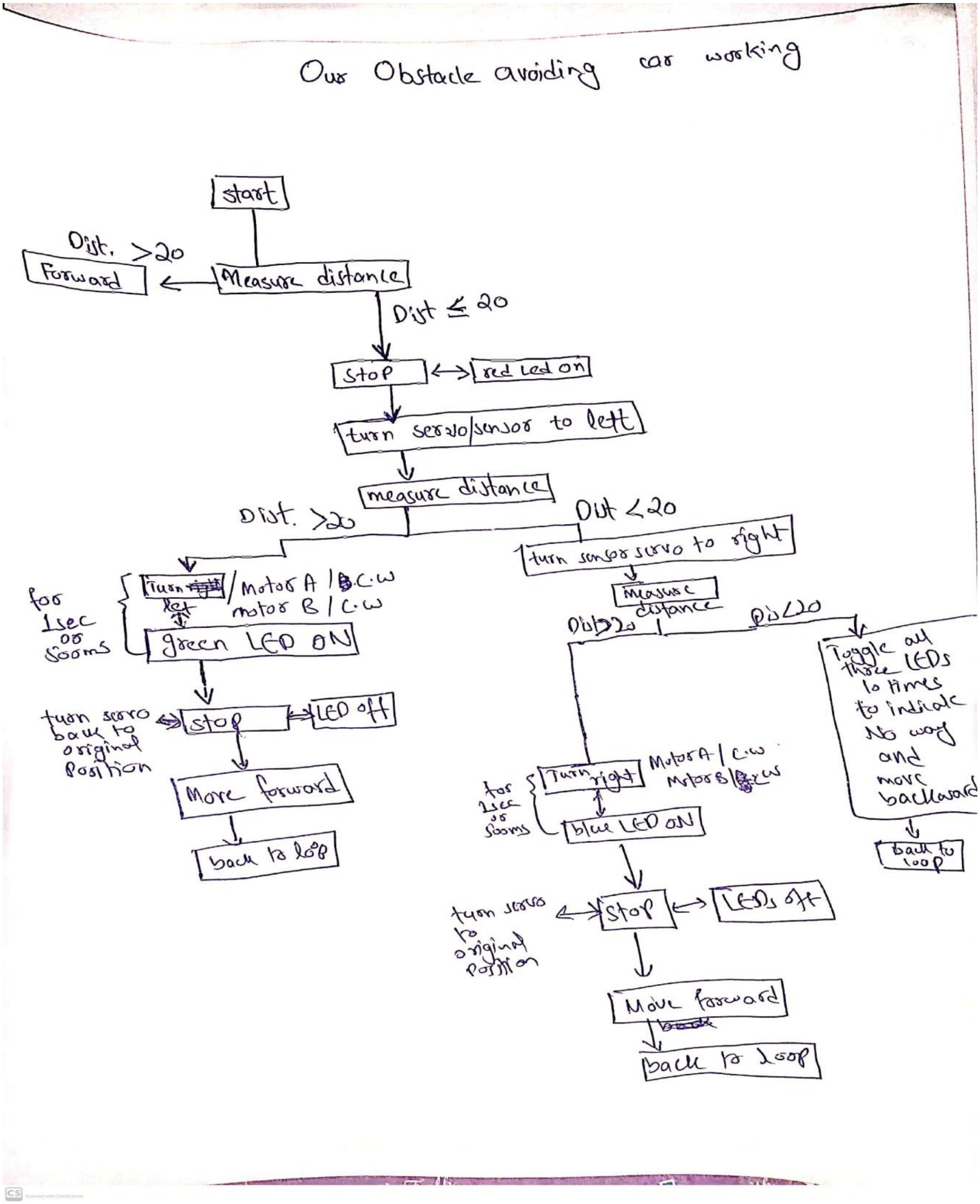
AS we can see in simulation the pulse duty cycle and hence the speed.



This will control the motor speed by controlling the ON time of the Motor and hence by increasing and decreasing the average voltage and power supplied to the motor.

Obstacle Avoiding Logic:

Following Block diagram explains the complete logic, and working principle of our project:



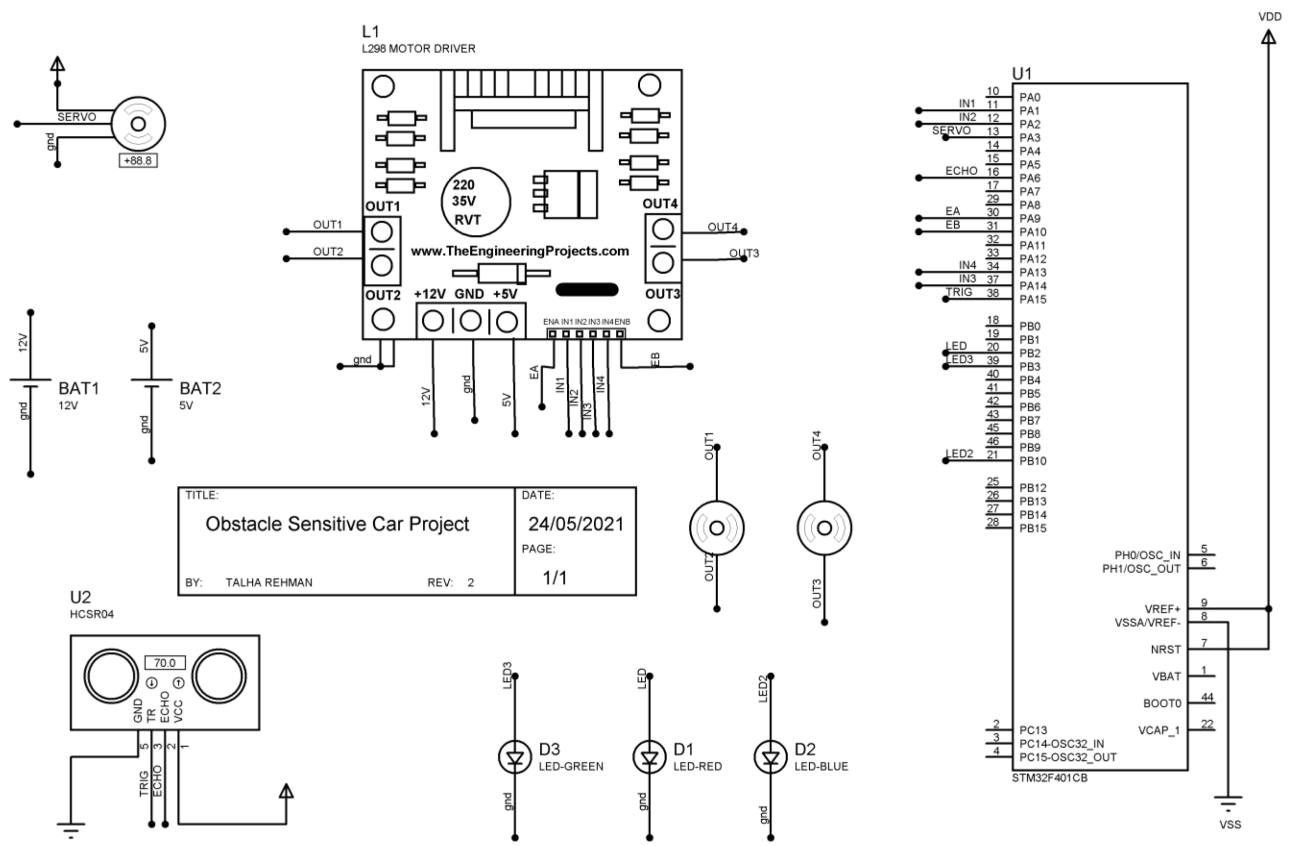
1. **Implementation with Microcontroller**

so far we have already discussed above all the sensor, servo, motor driver working and configuration now we will be building nested IF conditions logic to control our car based on the block diagram explained above.

Following is the summary of what we used in complete implementation:

TIMER 1 CHANNEL 2 -> PWM generation -> PA9 ->EA for motor A
TIMER 1 CHANNEL 3 -> PWM generation -> PA10->EB for motor B
TIMER 2 CHANNEL 4 -> PWM generation -> PA3
TIMER 3 CHANNEL 1 -> Input Capture Direct Mode -> PA6 for measuring ECHO pulse
TIMER 4 -> Microsecond delay function

GPIO outputs:
PB2 = RED LED => Stop indication/ obstacle indication
PB3 = GREEN LED => LEFT Clear/LEFT turn
PB12 = BLUE LED => Right Clear/Right turn
PA15 = TRIG => Trigger for Sensor HCSR04
PA1 = IN1 of motor driver
PA2 = IN2 of motor driver
PA13 = IN4 of motor driver
PA14 = IN3 of motor driver



a. CODING PART

```
/*
Complete Obstacle sensitive car code.
*/

HAL_Delay(70);// Hardware Delay necessary for the sensor perfect working

Dist_Measure();// Measures the Distance in cm

if(Distance == 0)continue;

if(Distance>30)
{
  Forward();     //If the distance is greater than 30cm the Car will move forward
}
else
{
  /*
  This part will be executed when there is an obstacle forward i.e. the forward obstacle is less than 30 cm
  */
  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_SET);  //Red led on indicating the obstacle near as the distance us less than 30

  stop(); // Stop the motor

  htim2.Instance->CCR4 = 400;  //Moving Sensor Servo to left  "0" degree to check any obstacle at left

  HAL_Delay(1000); //Simulation Checking Delay

  HAL_Delay(60); //Hardware Delay

  Dist_Measure();  //checking the left distance


if(Distance>30)   // if left distance is greater in this 'If' part the Car will turn left
{
  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET); //turn off the red LED

  htim2.Instance->CCR4 = 1143;// Moving sensor to original position --> 90 degree

  HAL_Delay(1000);// Simulation Checking Delay

  HAL_GPIO_WritePin(LED3_GPIO_Port,LED3_Pin,GPIO_PIN_SET); //Turning the green LED ON Indicating the Turning left

  Left(); //Turning Left

  HAL_Delay(1000);//Simulation delay -- In hardware must be 500ms

  HAL_GPIO_WritePin(LED3_GPIO_Port,LED3_Pin,GPIO_PIN_RESET); //Turn off the Green LED indicating that the car has turned left

  stop();//stoping the motors so it can now move forward
}
else
{
  /*
  This part is executed if  there is obstacle forward and also at the left
  */

  htim2.Instance->CCR4 = 1600;  //Moving Sensor Servo to Right  "180" degree to check any obstacle at Right

  HAL_Delay(1000); //Simulation Checking Delay

  HAL_Delay(70); //Hardware Delay

  Dist_Measure(); //checking the right distance
```

```c
if(Distance>30)
{
  /*
  This part will be executed when there is no obstacle at right and car will have to turn right.
  */
  HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET);//turn off the red led since motor is ready to move right as the distance is >20

  htim2.Instance->CCR4 = 1143;// Moving sensor to original position --> 90 degree

  HAL_Delay(1000);// Simulation Checking Delay

  HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_SET); // LED blue on indicating car is turning right

  Right(); //Turning Right

  HAL_Delay(1000);//Simulation delay -- In hardware must be 500ms

  HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET); //Turning off the Blue LED indicating the Car has turned right

  stop();

}
```

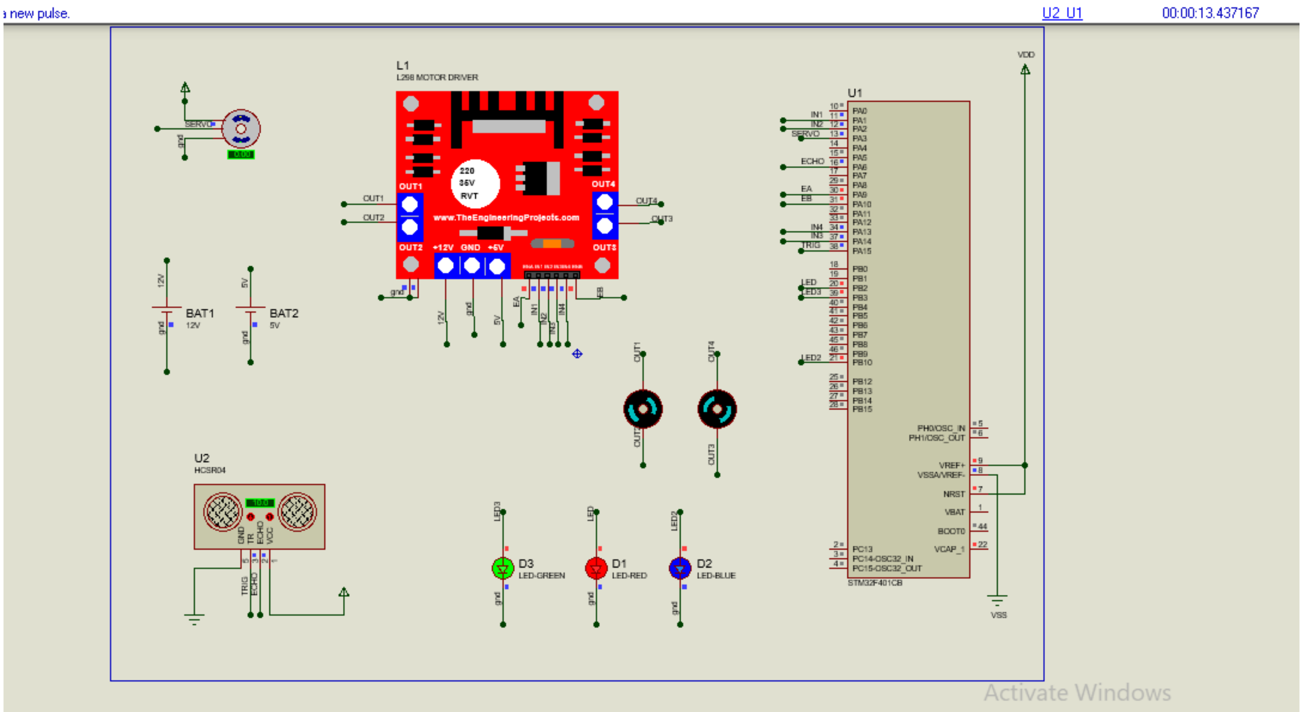
```c
    else
    {
      /*
        This part will be executed when there is obstacle on left right and forward
      */
      htim2.Instance->CCR4 = 1143;// Moving sensor to original position --> 90 degree

      HAL_Delay(1000);// Simulation Checking Delay

      htim1.Instance->CCR2=30;//setting motorA speed low 30%dutycycle

      htim1.Instance->CCR3=30;//seting motorB speed High 30%dutycycle

      Backward(); //moving backward

      HAL_Delay(1000);
      /*
      The following for loop is just to make indication that there is no way
      in this loop all the leds will start blinking for 2seconds
      */
      for(int i=10 ; i<=10 ;i++)
      {
        HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
        HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
        HAL_GPIO_TogglePin(LED3_GPIO_Port,LED3_Pin);
        HAL_Delay(300);
      }

      //Reinitializing all the leds
      HAL_GPIO_WritePin(LED_GPIO_Port,LED_Pin,GPIO_PIN_RESET);
      HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET);
      HAL_GPIO_WritePin(LED3_GPIO_Port,LED3_Pin,GPIO_PIN_RESET);

      stop();//stoping the car
    }
```

Before While loop this must be written.

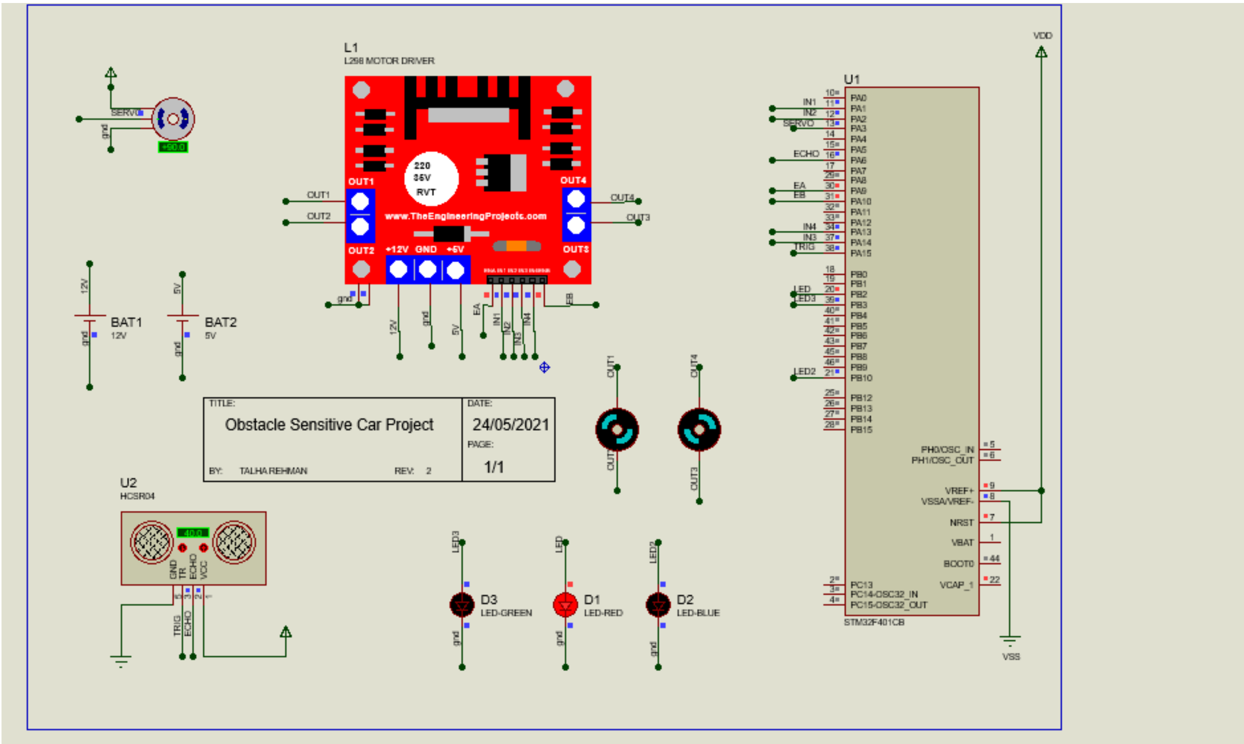
```
/* USER CODE BEGIN 2 */

HAL_TIM_Base_Start(&htim4); //Initialize the Timer 4
HAL_TIM_IC_Start(&htim3,TIM_CHANNEL_1); //start the input capture mode at timer 3 channel 1 i.e PA6 pin of Microcontroller
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_4);//start the PWM generation at Timer2 Channel 4 i.e. PA3 pin of Microcontroller
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2); //Start The Pwm for EA at PA9
HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_3); //Start the PWM for EB at  PA10

/* USER CODE END 2 */
```

b. Simulation PART

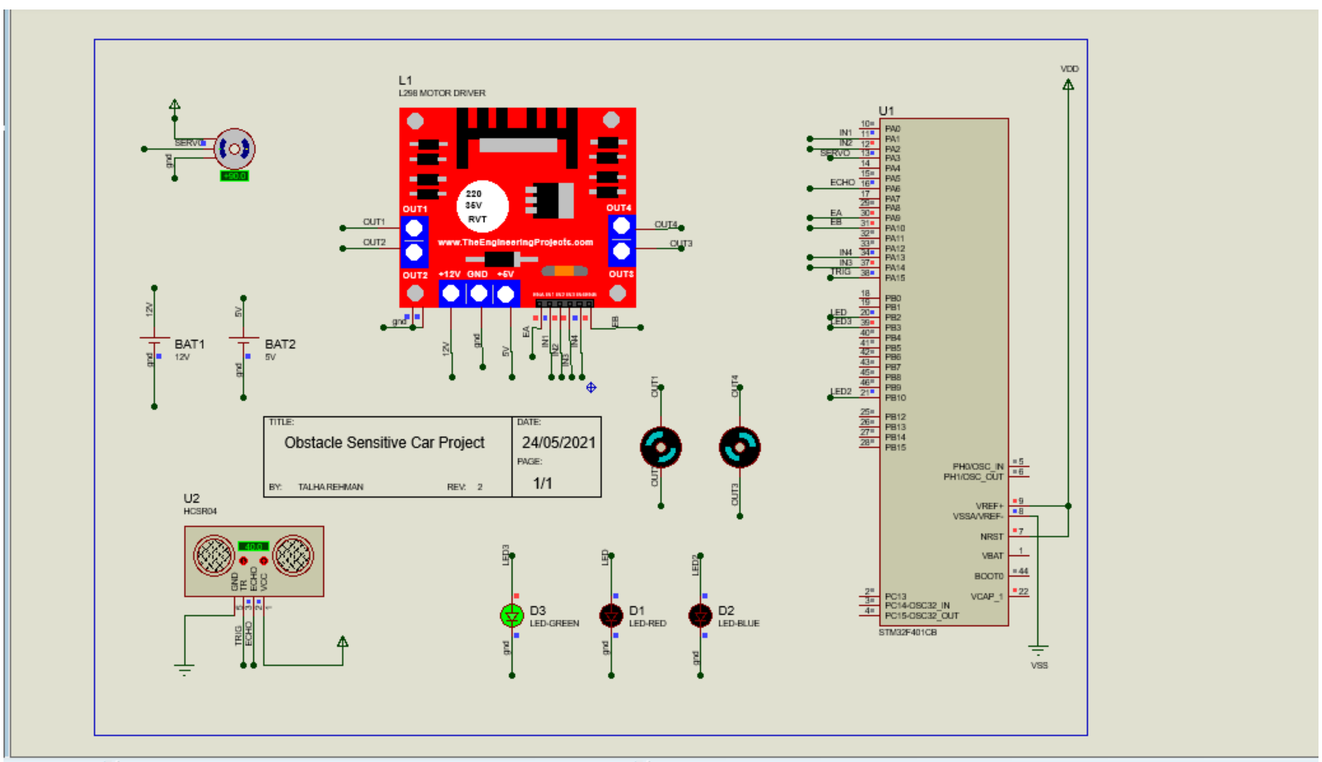
The following image is of simulation in the No way condition see all the LEDs are toggling.



Stop condition.


L1
L298 MOTOR DRIVER
U1
BAT1
12V
BAT2
5V
TITLE:
Obstacle Sensitive Car Project
DATE:
24/05/2021
PAGE:
1/1
BY: TALHA REHMAN
REV: 2
U2
HCSR04
D3
LED-GREEN
D1
LED-RED
D2
LED-BLUE
STM32F401CB

Left turn


L1
L298 MOTOR DRIVER
U1
BAT1
12V
BAT2
5V
TITLE:
Obstacle Sensitive Car Project
DATE:
24/05/2021
PAGE:
1/1
BY: TALHA REHMAN
REV: 2
U2
HCSR04
D3
LED-GREEN
D1
LED-RED
D2
LED-BLUE
STM32F401CB

Conclusion

The Microcontroller is useful in controlling the motor speed and direction accurately and we can have a deep insight of what is happening and can control and alter it to meet our needs. Furthermore, we can interface many other peripheral devices and sensors to design projects that may be the future tech or practical life helping instruments like home automation and Obstacle sensitive cars. Using this motor control technique can not only confine to only cars but this can be taken to much higher and complex scales like Robot Arms, manufacturing units, machines, bionic instruments (arms and limbs control by interfacing sensors and motors to work on the signal receive from body to provide desired motion). We can think of many other useful applications of this and can apply creativity for our own benefit. Now, we have worked out some basic implementation to get hands on with microcontroller and use its different alternate functionalities for our desired objective. Moreover, the software like STM32cubeMx and Proteus have provided us with environment that makes us more easy to get use to with Microcontroller and run simulation test before deploying our algorithm into the Hardware prototype.

Simulation limitations:

There were some unnecessary delays and some of them are used by us in order to keep the record of simulations. In real world and in building the hardware we will be using fewer delay to increase the efficiency of our car in detecting the objects.

References

1. https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet
2. https://controllerstech.com/hc-sr04-ultrasonic-sensor-and-stm32/
3. https://controllerstech.com/servo-motor-with-stm32/
4. https://controllerstech.com/create-1-microsecond-delay-stm32/

Timer as PWM, Input Capture Mode helping videos

5. https://youtu.be/koXHl_-PFqM
6. https://youtu.be/vDvbO_BAYvc
7. https://youtu.be/z3VBbj42NPs

Book:

ARM® Microprocessor Systems Cortex®-M Architecture, Programming, and Interfacing by Muhammad Tahir and Kashif Javed

Appendix

The following are the functions used in our project.

```c
//Forward Moving Function//
void Forward(void)
{
  htim1.Instance->CCR2 = 90;
  htim1.Instance->CCR3 = 90;
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_RESET);
  HAL_Delay(50);
}
//BACKWARD moving Function//
void Backward(void)
{
  htim1.Instance->CCR2 = 90;
  htim1.Instance->CCR3 = 90;
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_SET);
  HAL_Delay(50);
}
//Turning Right Function//
void Right(void)
{
  htim1.Instance->CCR2 = 30;
  htim1.Instance->CCR3 = 30;
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_SET);
  HAL_Delay(50);
}
```

```c
//Turning Right Function//
void Right(void)
{
  htim1.Instance->CCR2 = 30;
  htim1.Instance->CCR3 = 30;
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_SET);
  HAL_Delay(50);
}
//Turning Left Function//
void Left(void)
{
    htim1.Instance->CCR2 = 30;
  htim1.Instance->CCR3 = 30;
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_RESET);
  HAL_Delay(50);
}
//Stop Function//
void stop(void)
{
  HAL_GPIO_WritePin(IN1_GPIO_Port,IN1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN2_GPIO_Port,IN2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN3_GPIO_Port,IN3_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(IN4_GPIO_Port,IN4_Pin,GPIO_PIN_RESET);
  HAL_Delay(50);
}

/* USER CODE END 4 */
```