Project Report

Accident Alert in Mist

List of group members

Roll Number	Name
AM.EN.U4ELC19025	Chiradeep P
AM.EN.U4ELC19026	Prasana Swaroop
AM.EN.U4ELC19027	Sai Jeevan P
AM.EN.U4ELC19028	Ranit Pradhan
AM.EN.U4ELC19029	Lakshmi Shravika

Abstract

- In the mist areas, the driver in vehicles cannot see the vehicles at front properly. This project is developed for vehicles accident alert in the mist areas.
- The main modules in this project are Infrared transmitter, Infrared receiver, Microcontroller unit and audio section.
- An Infrared signal is produced by using the electronic circuit. The Infrared produced is transmitted along a particular direction.
- The Infrared signal travels and falls on the Obstacles and reflected back. This setup is connected at the front of the vehicle.
- In the electronic circuit, there is an Infrared receiver. The reflected Infrared signal can be received through the receiver if there is any obstacles.
- The microcontroller counts the time taken between the transmission and the reception and thus calculates the distance. If there are obstacles in front of the vehicle, it produces a warning signal.
- Thus, the driver can easily stop the vehicle. This project is very useful for vehicles in the mist areas.

Real world challenge identified

Real world problem identified is **Accidents caused due to mist** on roads shown in figures below and how vehicle drivers are affected due to this dangerous situation and how does mist affect driver's vision.

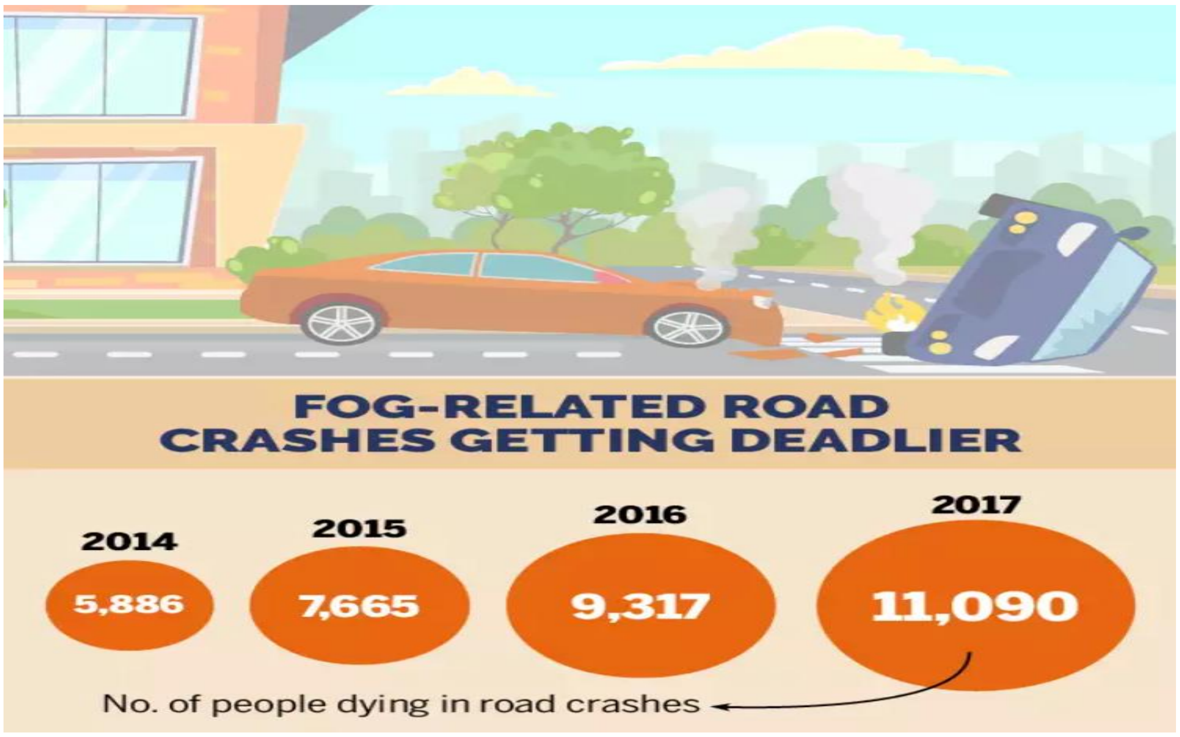
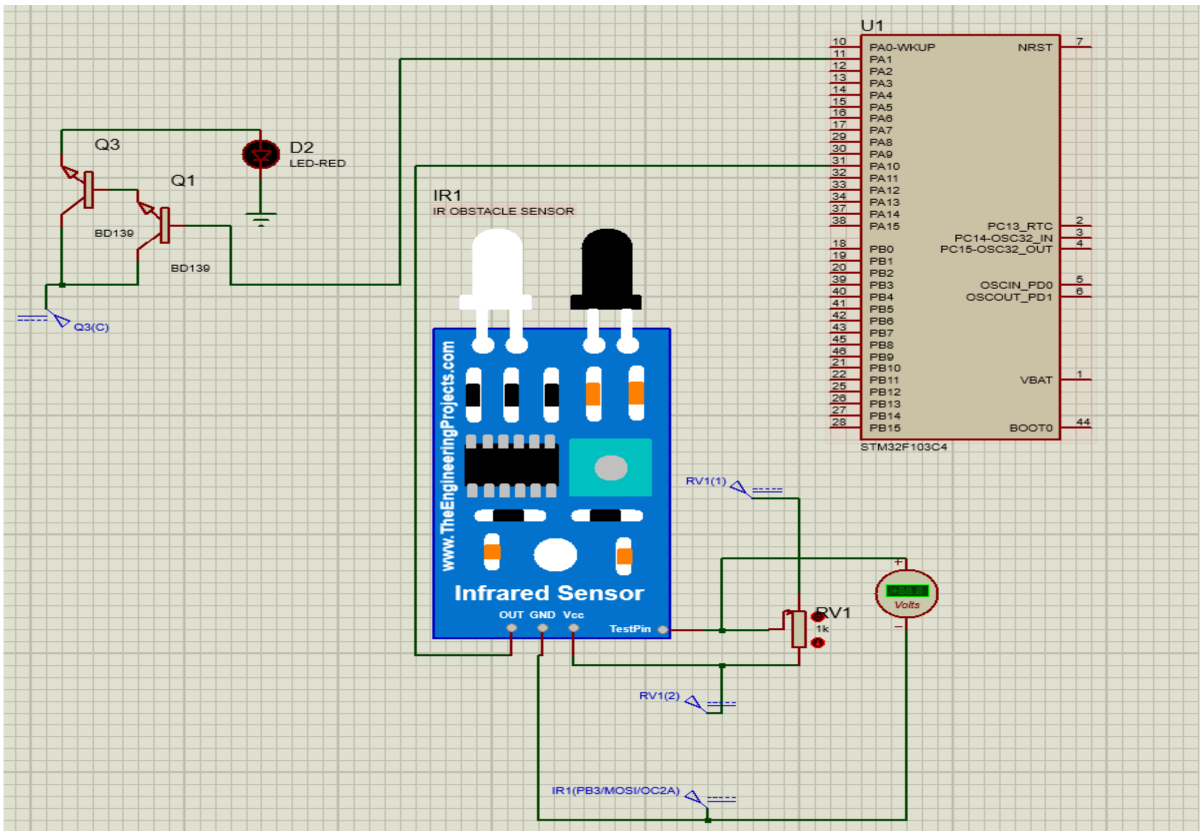


Image showing the increase in number of people dying in accidents due to mist

In mist areas the vehicle drivers can not see the vehicles in front, This obviously leads to accidents. However mist conditions also promote accidents because they effects perceptual judgements of speed and distance.

The effects are the result of reduced contrast. Depth perception can be reduced due to the thickness of fog. Low contrast caused by mist makes it difficult to distinguish between light and dark areas and objects on road.

Proposed solution using STM32F103C4 based embedded platform



We have developed a electronic circuit having Infrared sensor along with STM32F103C4 Micro controller.

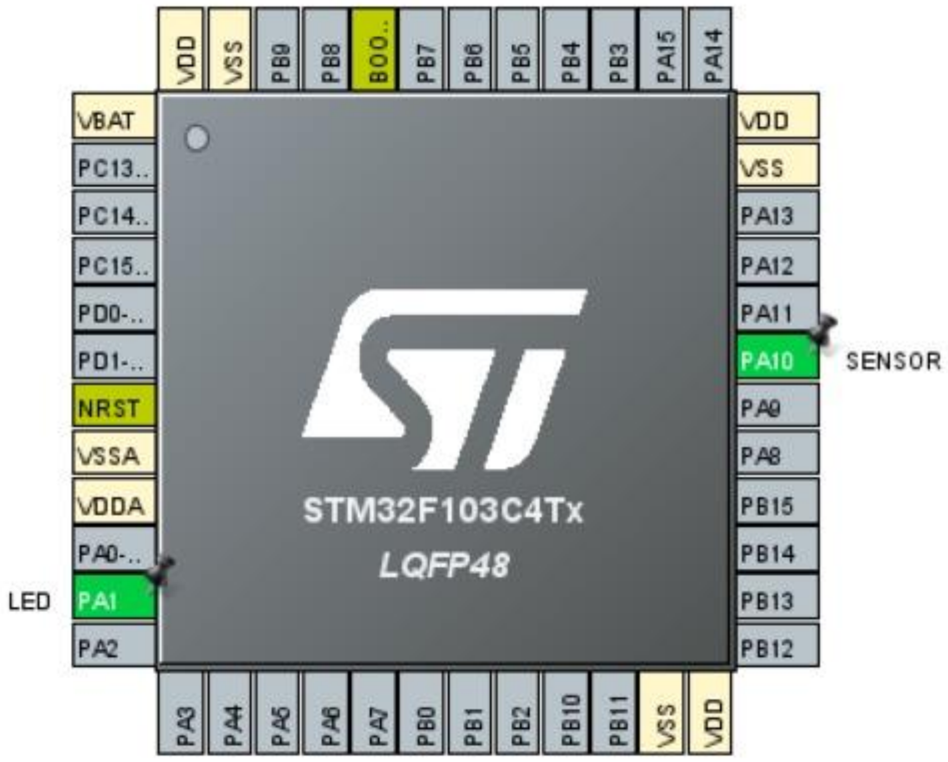
An Infrared signal is produced by using the electronic circuit. The Infrared produced is transmitted along a particular direction. The Infrared signal travels and falls on the Obstacles and reflected back. This setup is connected at the front of the vehicle.

In the electronic circuit, there is a Infrared receiver. The reflected Infrared signal can be received through the receiver if there is any obstacles.

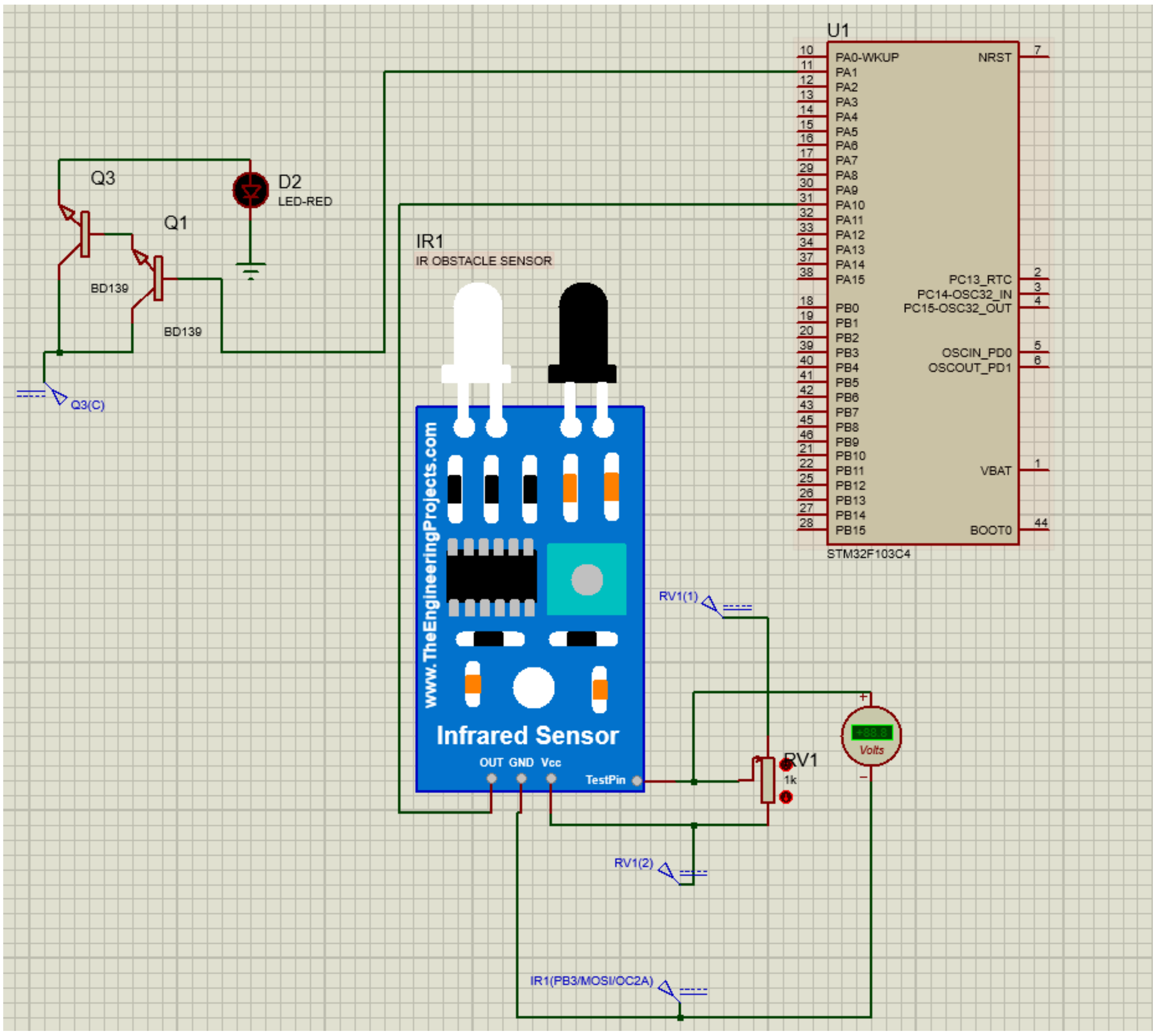
The microcontroller counts the time taken between the transmission and the receiption and thus calculates the distance. If there is obstacles in front of the vehicle, it produces a warning signal. Thus the driver can easily stop the vehicle.

Here in our electronic circuit IR sensor takes the distance of the obstacle and then by processing the distance data collected and checking if the distance is safe or not an alert will be generated to initimate the vehicle driver.

Details of electronic circuit(s) developed as part of proposed solution



- In STM32CubeMX software we selected STM32F103C4 as microcontroller.
- Then configured the GPIO pins (PA10 as GPIO input from the sensor and PA1 as output to the LED).
- Without making any other configurations we have generated the code that directly goes to ARM Keil.
- Further logics that are required to work on Microcontroller are written in ARM Keil.
- Then in Proteus we have built the following circuit.



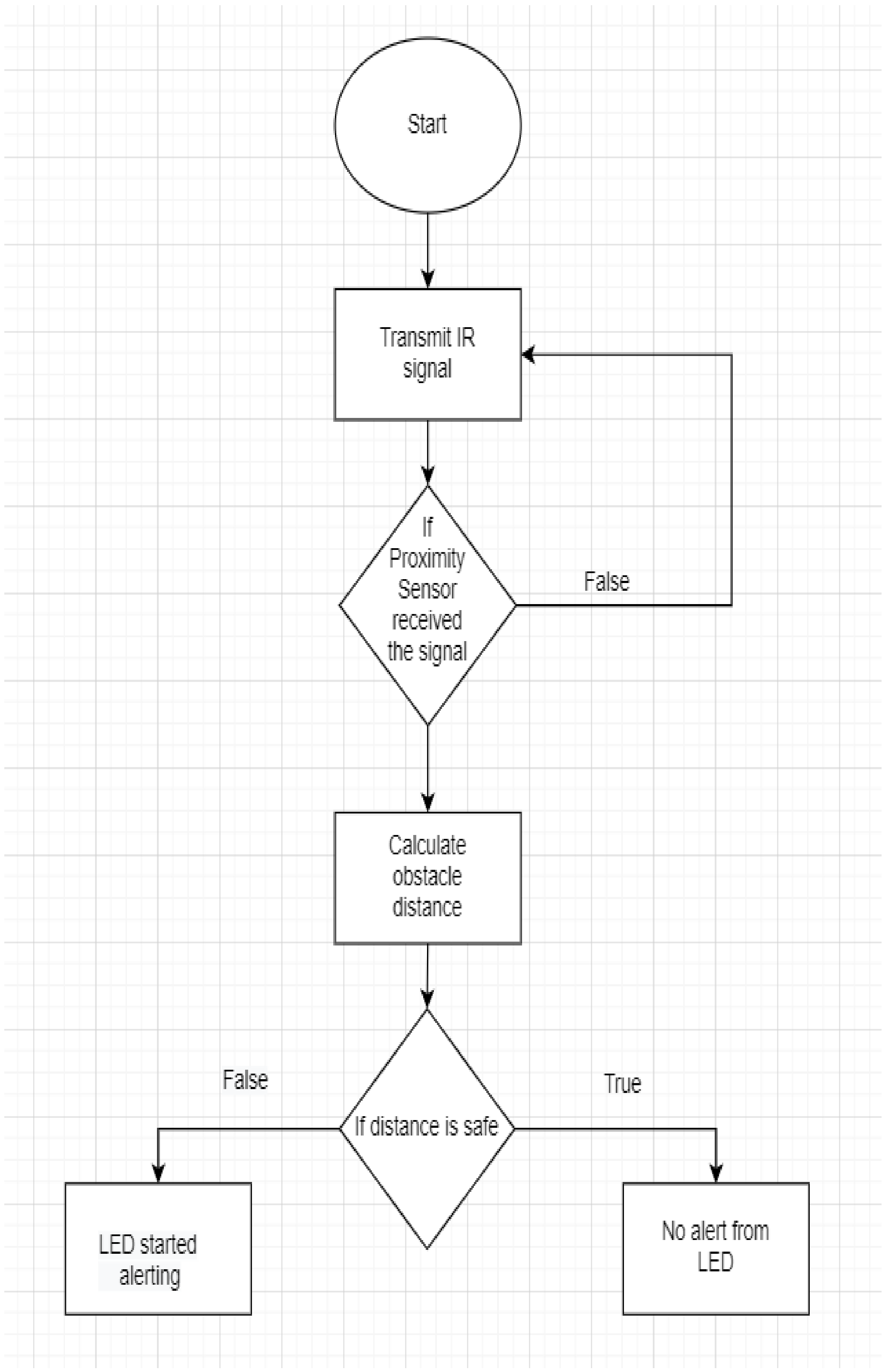
We have used STM32F103C4, IR Sensor, Potentiometer, Voltmeter, LED and Transistors. Here LED is of 5 volts and IR Sensor of 9 volts.

We have used 3 software platforms STM32CubeMX for configuring STM32F103C4, ARM Keil for writing the required logics as C program, and Proteus 8 for setting up the circuit with STM32F103C4 for implementing the required logic with the sensors, IR Sensor for calculating distance of the objects in mist, LED for alerting and Potentiometer (Voltage regulator) for mimicking the distance for simulation purpose.

Details of peripherals and other aspects utilised from STM32F103C4

The peripherals we have used are GPIO pin PA1 as output to LED and PA10 as input from the sensor. Other aspects like clock frequency and all we kept default for this circuit.

Flowchart developed for the program



Program developed as part of the solution

```
/* USER CODE BEGIN Header */
/**
  **************************
  * @file           : main.c
  * @brief          : Main program body
  **************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause
license,
  * the "License"; You may not use this file except in compliance with
the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  **************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------
--------*/
#include "main.h"

/* Private includes ----------------------------------------------------
--------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------
--------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------
--------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------
--------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------
--------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------
--------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
--------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration-----------------------------------------------------------
--------*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
           if(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_10)==1) //Sensor Active
           {
                HAL_GPIO_WritePin(GPIOA,GPIO_PIN_1,1); //LED ON
           }
           else
           {
                HAL_GPIO_WritePin(GPIOA,GPIO_PIN_1,0); //LED OFF
           }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) !=
HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : LED_Pin */
  GPIO_InitStruct.Pin = LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : SENSOR_Pin */
  GPIO_InitStruct.Pin = SENSOR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(SENSOR_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error
return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and
line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n",
file, line) */
  /* USER CODE END 6 */
}
```

```
#endif /* USE_FULL_ASSERT */

/******** (C) COPYRIGHT STMicroelectronics **END OF FILE*/
```

Simulation outputs and analysis

Following images are the OUTPUTS at two different conditions:

1. If Obstacle at safe distance → LED OFF
2. If Obstacle at unsafe distance → LED ON

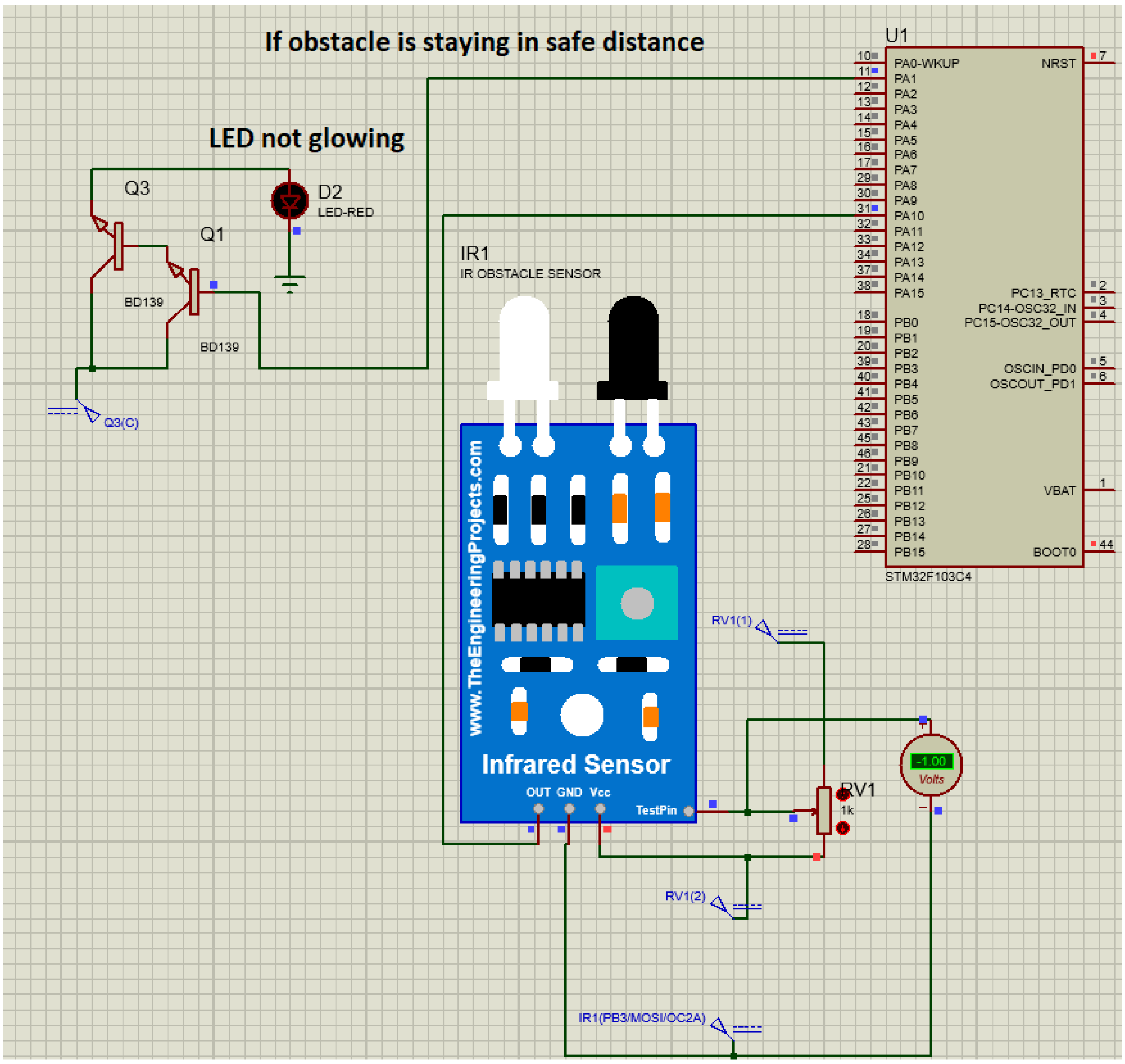


Image with LED not glowing since obstacle is in safe distance

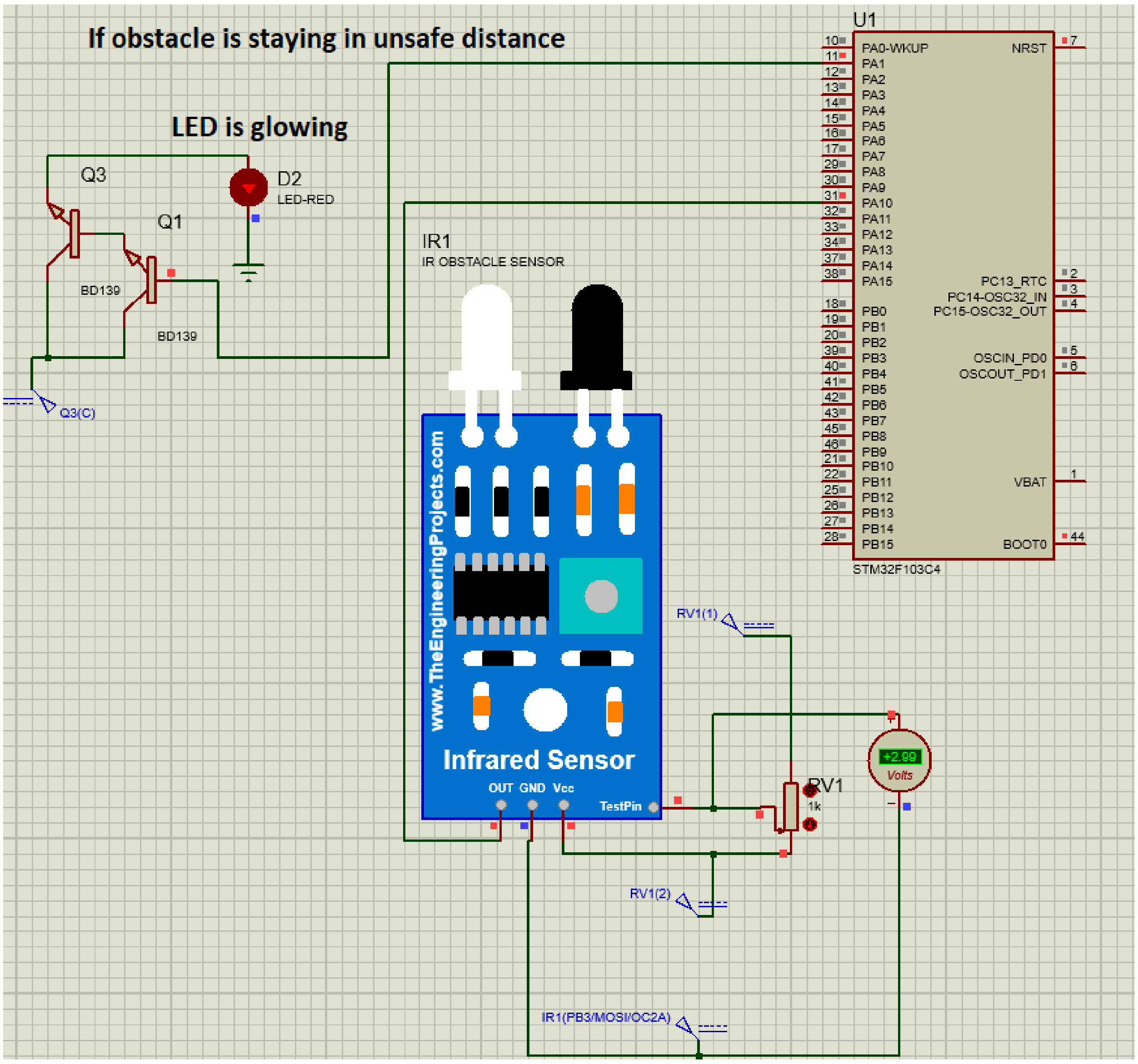


Image with LED glowing since obstacle is not in safe distance

Here we used a Red LED as a visual alarming component, in the real world besides LED we can also keep buzzer as a sound alarming feature.

Conclusion

In this project we have developed an electronic circuit which can detect the objects at front of vehicles in mist conditions to avoid accidents. This is easy to setup and low budget electronic circuit. Nowadays modern vehicles having Night vision, GPS features which is one of the best solution for the particular problem, night vision also use the same theory of IR.