Department of Electrical & Computer Engineering

COEN 6711
MICROPROCESSOR AND ITS APPLICATIONS.

Project Report on
"Automatic Braking System".

Submitted to:

Dr.Abdelaziz Trabelsi

Ganesh Santhar 40010625
Kamalpreet Grewal 40010401
Jivanjot Singh Bajaj 40030329
Opeyemi Oyedepo 27216386

Date Submitted: 03/12/2016

Abstract

This project work presents an ultrasonic automatic braking system for forward collision avoidance with accelerator pedal disengagement mechanism. In our present world, safety has become important aspects of automobile design. Automation is the key, which keep the safety at our fingers. Accidents happen with the automobile vehicles that cause serious injury or even death. One common cause is the failure to apply the brakes is such critical situation or some inefficient braking system.

An automobile braking system is an important innovation to the automobile domain that can assist drivers to brake a car while avoiding imminent obstacle that could cause collision and other fatal condition. The system is based on an ultrasonic emitter and receiver that helps in producing and receiving the ultrasonic waves to determine the distance between a car and an obstacle. This can be used to provide some level of safety in an automobile system. This project will focus on a design that can help stop the automobile by disabling the acceleration and optionally turning away from the direction of the obstacle.

There are ultrasonic sensors that will be mounted on the vehicle, the signals are transmitted constantly from it, and the reflected signals are received back from the obstacles if any. A decoder module in the firmware converts the received signals into relative distance between the obstacle and vehicle. A safety module monitors the calculated distance and if the distance measured crosses above defined safety limit, with the help of a Motor driver, which is integrated to the FRDM-KL25Z to drive the vehicle, the FRDM-KL25Z firmware will stop the motor driving function and disable acceleration.

The core component and the brain is the Microprocessor Arm Cortex M0+ KL25Z from Arm, which interfaces with other part of the system to provide the desired functionality. Design, procedures and implementation details are discussed and further improvement are suggested.

Table of Contents

List of Tables

List of Figures

1. Introduction

The number of accidents have become rampant today and these accidents are mostly caused due to delay of driver to hit the brake and results in forward collision.

The act of driving requires a high level of co-ordinations and sometimes human error do occur, for example, reversing accidents are very common. This can be avoided if there the system has a braking system that alerts the driver about the distance from obstacles in the reversing vehicle pathway.

Based on the current fast development in the areas of real time embedded systems, there has been a tremendous increase in the number of Automobiles, as they have become a major tool of transportation in the current society. Automobile safety system becomes very important as its number soars nowadays.

Important features and benefits of the system includes, safety of drivers and avoiding accidents by stopping the vehicle in the shortest distance. The sensors can quickly sense obstructions there by acting instantaneously in case of emergency with the least effort from the driver

1.1. Aim and objectives

The aim of this project is to implement an Automatic Braking system based on the FRDM-KL25Z, this microcontroller will receive a signal about an obstacle detected by an ultrasonic Sensor, stop motor driving function, and disable acceleration through the integrated motor Driver IC. The system is required to stop the vehicle within a smallest possible distance. Of importance is that, there is little or no effort on the part of the driver to achieve this functionality. Specific objective in the project includes.

1. To explore in detail all the features and functionality of FRDM-KL25Z.
2. Understanding the interface of FRDM-KL25Z, with the other important components.
3. To develop the functional modules of the braking system such as the obstacle detection, motor driver and the user interface using a wireless technology
4. The performance of the system should adhere to the standards of road safety guidelines
5. To present a Final report and Demonstration of the working prototype

1.2 Motivation

The use of various types of automobile has become an integral part of human activity. The movement of people from one location to another is a typical scenario of the daily use of automobile. Thus, the number of automobiles is increasing day by day. However, some uncertain and unpredictable situations such as accidents do occur from time to time in an increasing manner. Accidents do occur every time and everywhere and can result into severe damages, serious injury or even cause death. These accidents are usually caused by several factors such as the delay of the drivers to apply the brake at the quickest possible time or some kind of failure on the part of the braking system. This project is designed to develop a new system that can solve this problem where drivers may not brake manually but the vehicles can stop automatically when detecting obstacles.

1.3 Scope of Project

This project focus on the implementation of an ultrasonic automatic braking system for forward collision avoidance with accelerator pedal disengagement mechanism. The system will be emulated using two-wheeled Robot Chassis vehicle. Furthermore, The Arm cortex M0+ being the core of the system design will interface with other important component such as the sensor, Motor IC and Bluetooth module as a user interface.

2. System Design

This section describes Automatic Braking System design, which aims to emulate the real-world scenario of automobile collision prevention concept. To demonstrate it's sophisticated and safety feature to the Target Industry, this project makes use of robot chassis Vehicle as Target Application. The Automatic Braking System comprises of three functional modules, which enables it to achieve its goal. The Modules are

1. Steering Module – conducts the driver module through user interface
2. Motor Driver Module – drives the RC vehicle
3. Obstruction detection Module – detects the obstacle

All the above modules work in parallel to emulate the target scenario automatic collision prevention.

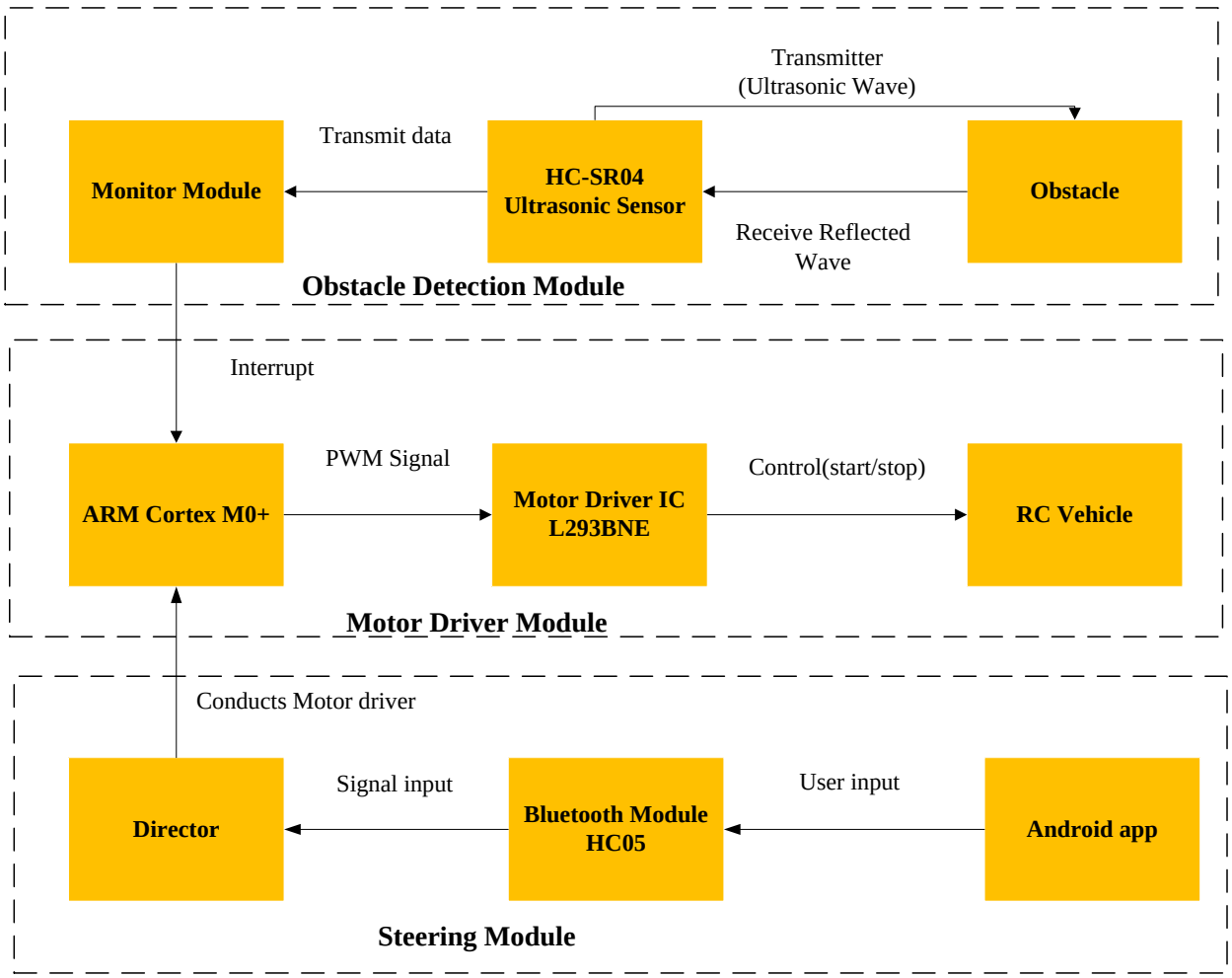

Figure 1: Automatic Braking System Design

2.1 Steering Module

The primary functionality of this module is to provide user a way to control the RC vehicle. It realizes it through employing wireless technology Bluetooth. From the above figure, Steering Module encompasses below

- FRDM KL25Z (Arm Cortex M0+ Microcontroller)
- Director software
- Bluetooth Module HC-05
- Android app

The Microcontroller controls driving Module based on the inputs from the Bluetooth module.

Direct software is a software program which works between the Bluetooth device and the Microcontroller and process the inputs from HC 05. It sits inside the motor driver software component, suitable location to direct driving software. It stops directing the motor if it receives interrupt from Obstruction detection module and disable the acceleration control from the user towards the obstacle. It prevents the acceleration until the interrupt is removed.

The Bluetooth Module is interfaced as serial device with the Microcontroller. Detailed information of interfacing Bluetooth module HC05 with microcontroller and its pin configuration is available in the Hardware interface section.

Android app, the primary user interface that gets inputs from the user and displays current condition to user. It has a glossy graphical interface with forward, left, right, reverse and stop button to get inputs of steering as shown in the figure. It also has buttons and interfaces to connect Bluetooth device and display the status. It performs so by establish wireless connection with the Bluetooth module HC 05.

2.2 Motor Driver Module

This module implements the driving functionality of the Automatic Braking system. It achieves the requirements by combining software and hardware components listed below

- FRDM KL25Z (Arm Cortex M0+ Microcontroller)
- Motor driver Software
- Dual Bridge Motor Driver IC L293DNE
- Robot Chassis Vehicle

The Kl 25Z Microcontroller generates PWM signal to drive Motor Drive IC and in turn Robot Chassis Vehicle.

Motor Driver software is a software Program that manipulates the PWM signal for driving Moto driver IC in Forward, Left, Right, Reverse and Stop directions.

Motor Driver IC L293DNE takes PWM signal from KL 25Z and sends electrical signals to dc motors. It can drive two motors at the same time. Its interface to Microcontroller and the DC motors of RC vehicle can be found at Hardware interface section.

Robot chassis Vehicle emulates the automobiles from target scenario. Which is the target application here. The project concept is show cased in its motional experiments. It is powered with

two dc motors to drive two wheels of the vehicle. This vehicle hosts entire components of the project components in it. The top rack of the vehicle carries the Microcontroller, Bluetooth module and Ultrasonic sensor. The bottom one contains the power supplies to motor and Microcontroller such as AA batteries and Power banks.

2.3 Obstruction Detection Module

This is the core module of ABS which senses the obstacles to RC vehicle and feedbacks the obstacle detect data to Microcontroller. The Module It achieves the objective by combining software and hardware components listed below

- FRDM KL25Z (Arm Cortex M0+ Microcontroller)
- Monitor Module
- Ultrasonic Sensor HC SR04

The Microcontroller plays an important role in this module by generating trigger signals to and receiving echoed signals from Ultrasonic sensor.

Decoder Module is a software Program that reads the feedback data from Ultrasonic sensor and converts the data in seconds to meaningful distance units. It also tracks the distance data for safety minimum value. If the distance happens to be below safety minimum value it interrupts Motor driver software to brake the RC Vehicle.

Ultrasonic Sensor works on generating ultrasonic waves to detect the obstacle in front of it. As mentioned above it gets trigger signal from microcontroller to perform its operation. The generated waves travel in front until it finds an obstacle and the waves are reflected to sensor. The sensor captures the time of this process and feedback to Kl 25Z.

3. Software Description

To realize the real world scenario the software design utilizes free RTOS as its base software to provide real time computation support. The main application software of ABS runs on top of the RTOS. The architecture of the software is given below:

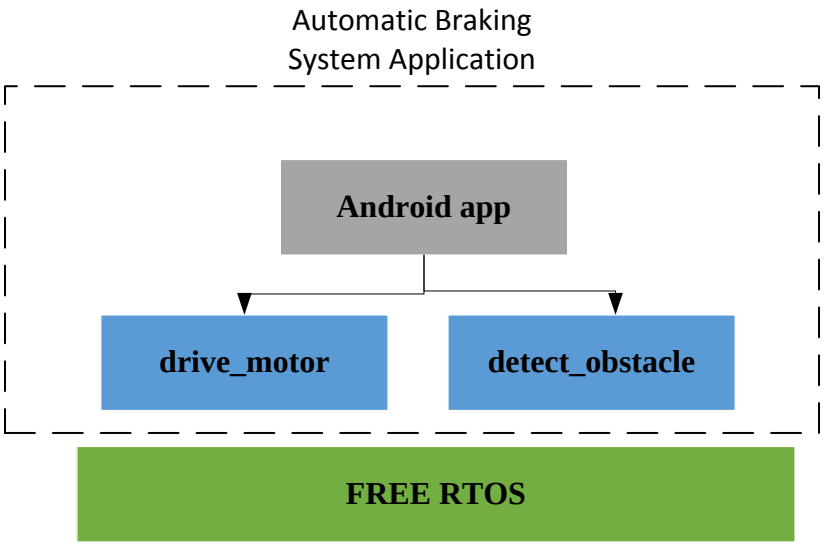

Figure 2: ABS Software Architecture

The main application process consists of three software modules detect_obstacle, driver_motor and director. The software modules driver_motor and detect_obstacle are implemented as parallel threads and they work in tandem. The module director is a software function sits inside motor driver software receives input from Bluetooth module.

3.1 Software Tools

The software is programmed in embedded C++. The open source cross compiler GCC arm is used to build the application software and free RTOS. The application employs Mbed library as the predominant external library package. The online compiler from mbed was used through the development process as well.

For the software, we used MBED online compiler.

Benefits of using MBED Compiler:

- USB drag and drop programming interface
- Entry-level online Compiler
- High-level peripheral abstraction
- Easy to use C/C++ SDK
- Lots of published libraries and projects

4. Hardware Description

Some of the important activities of our implementation includes the understanding of functional behavior of the component and how they are to interface with our Microcontrollers. The implementation of the hardware and interfacing design were also crucial. Other activities were carried out in relation to testing to ensure that we are able to achieve the implementation

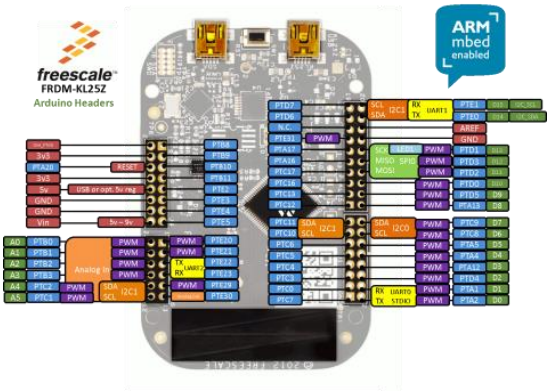

Figure 3: FRDM KL25Z

4.1 FRDM KL25Z Processor

The Freedom KL25Z is an ultra-low-cost development platform built on ARM® Cortex®-M0+ processor.it has very easy access to MCU I/O, battery-ready, low-power operation, a standard-based form factor with expansion board options and a built-in debug interface for flash programming and run-controller

The key Features of the Processor includes-

- MKL25Z128VLK4 MCU – 48 MHz, 128 KB flash, 16 KB SRAM, USB OTG (FS), 80LQFP
- Capacitive touch "slider," MMA8451Q accelerometer, tri-color LED
- Easy access to MCU I/O
- Sophisticated OpenSDA debug interface
- Mass storage device flash programming interface (default) – no tool installation required to evaluate demo apps
- P&E Multilink interface provides run-control debugging and compatibility with IDE tools
- Open-source data logging application provides an example for customer, partner and enthusiast development on the OpenSDA circuit
- mbed enabled

4.2 HC-SR04 Ultrasonic Sensor

The HC-SR04 sensor is an ultrasonic sensor with 4 pin and generally used because of its incredible price point. Its pins are as follows:

1. Vcc (+5V DC supply)
2. Trig (TTL input, to trigger a measurement)
3. Echo (TTL output, pulse proportional to distance)
4. GND (ground)

Ultrasonic sensors (also known as transceivers when they both send and receive, but more generally called transducers) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

Product features:
Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time × velocity of sound (340M/S) / 2, λ

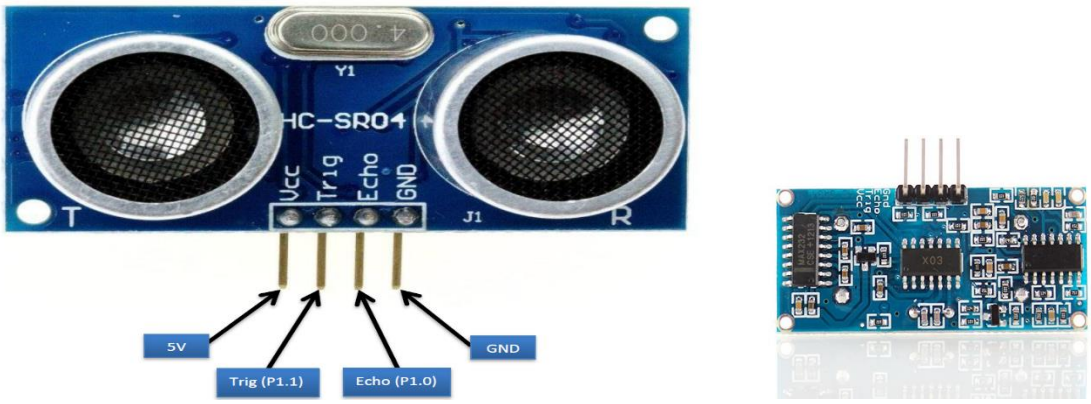

Figure 4: Front and Back View of Ultrasonic Sensor Pinout

Timing diagram:

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion.

You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

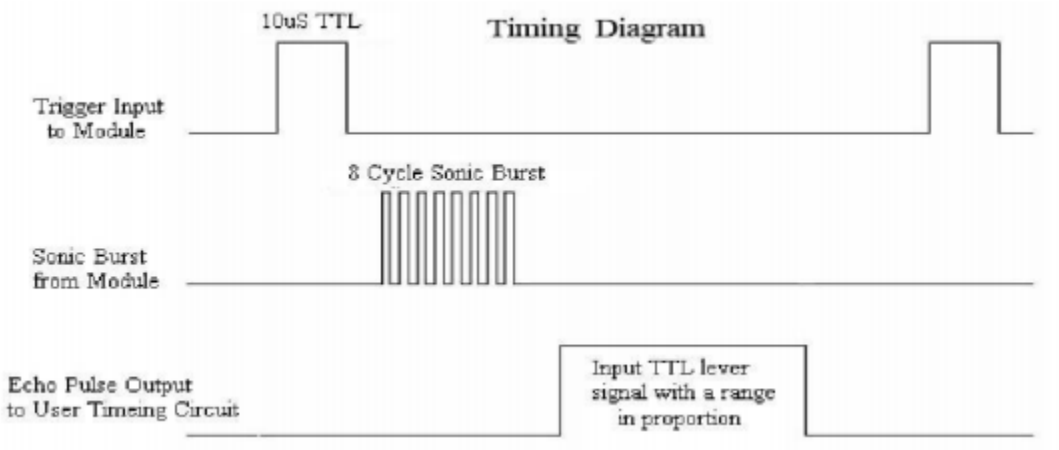

Figure 5: Ultrasonic Sensor timing

4.3 Voltage Divider

The HC-SR04 uses TTL (5V) supply voltage and logic levels. The FRDM-KL25Z processor has a 3.3V voltage, but the board provides 5V on the header. The HC-SR04 can use 3.3V levels on the Trig signal, but provides a 5V signal on the Echo pin. To get the signal to the 3.3V level, simple voltage divider with a 21k Ohm and 15k Ohm resistor is used.

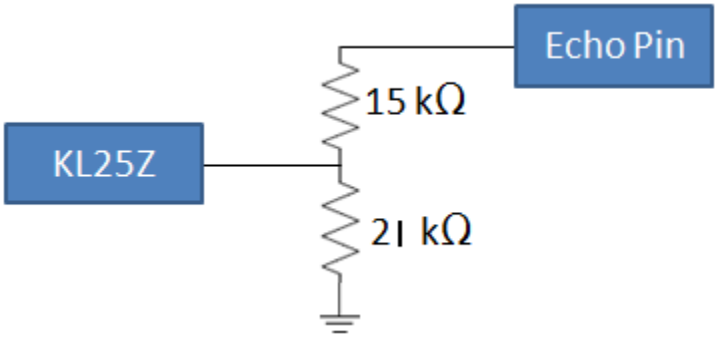

Figure 6: Voltage Division with Resistors to KL25Z

4.4 Robot Car Chassis kit

This chassis is a versatile robot platform featuring two gear motors with 65mm wheels and a rear caster for differential movements. The chassis plates are cut from acrylic with a wide variety of mounting holes for sensors, controllers, power, etc. Simply bolt the two pre-cut platforms together, attach the motors and caster and add your favorite robotics controller. This kit includes all of the parts needed to assemble the chassis as well as a 4xAA battery holder with barrel jack termination.

Functions and Features of the Car Chassis kit:-

- The Mechanical structure is simple, it is easy to install
- This car is the tachometer encoder
- With a 4 AA battery box (batteries not included)
- Can be used for distance measurement, velocity
- Can use with other devices to realize function of tracing, obstacle avoidance, distance testing, speed testing, wireless remote control , Size: 20 x 14cm (L x W) Wheel size: 6.5 x 2.7cm (Dia. x H)

Figure 7: Robot Car Chassis kit

4.5 Motor Driver - L293DNE

The IC Run four solenoids, two DC motors or one bi-polar or uni-polar stepper with up to 600mA per channel using the L293D. It is a high current Motor driver IC with dual H-bridge for controlling up to two motors at a time. Best for Inductive load de coupling from the main/control unit and the wide voltage range allows for an adaptive voltage range control.

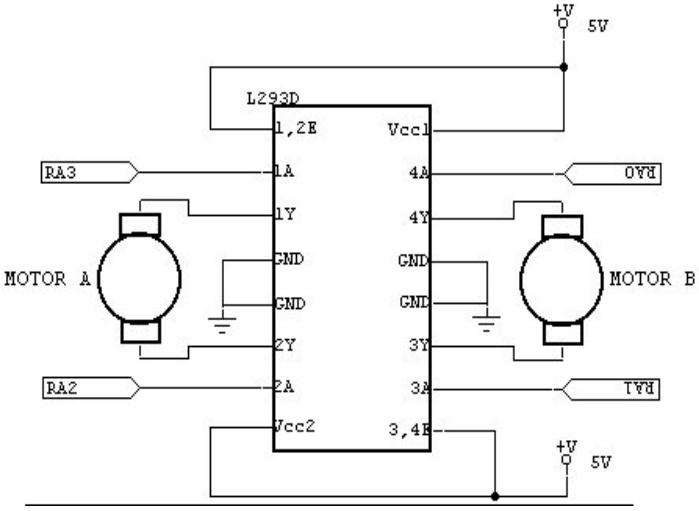

Figure 8: Motor Driver Pin connection

- Wide Supply-Voltage Range: **4.5 V to 36 V**.
- Separate Input-Logic Supply.
- Internal ESD Protection.
- Thermal Shutdown.
- High-Noise-Immunity Inputs.
- Output Current 1 A Per Channel (**600 mA for L293D**).
- Peak Output Current 2 A per Channel (**1.2 A for L293D**).
- Output Clamp Diodes for Inductive Transient Suppression (L293D).

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc 2
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc 1

Table 1: Motor Driver Pin function name

4.6 Bluetooth Module HC 05

Operates from 3.6v to 6v, which means you can use interface this simple device to any micro controllers. The added simplicity of the four pre-soldered connectors results in a very easy n' quick plug and play for any future projects. This device can work up to 10 meters, the voltage supplied must not exceed 6v and it is not reverse polarity protected. Comes preheat-shrink with clear coating to protect the device from anti-static discharge, dust protection and accidental shortage while maintaining an elegant look. Use this 3.6cm x 1.5cm device for any project you may have in mind and rest assure with this Bluetooth based device will consume less power than

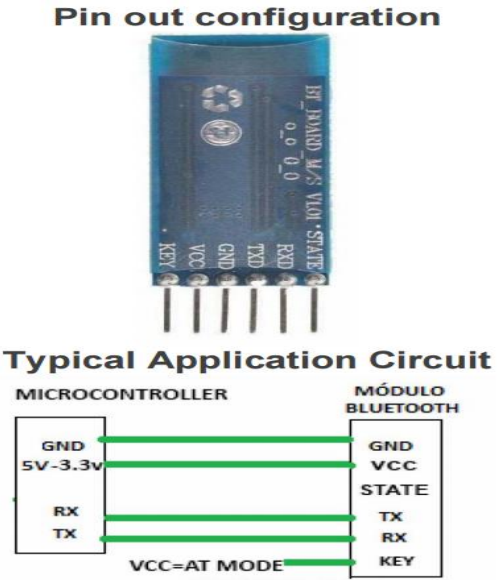

Figure 9: Bluetooth Pin out connection

a traditional WiFi Module will! No Firmware = No Fuss, pairing code 1234 (default) and Logic Levels of RX & TX are 3.3V.

4.7 Power Supply Units

There are two main power supply sources to our automatic braking system

- The a 4 -AA finger batteries which powers the Robot Car Chassis kit
- The power bank, which provided power supply to Microcontroller FRDM KL25Z

4.8 Hardware Interfacing

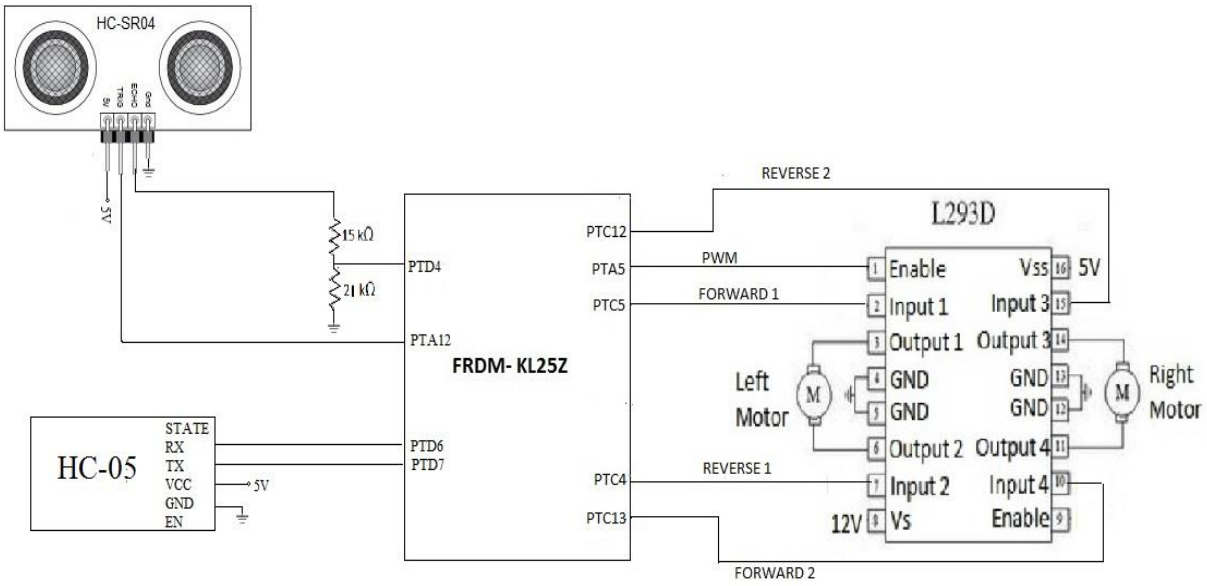

Figure 9: MCU interfacing with Modules

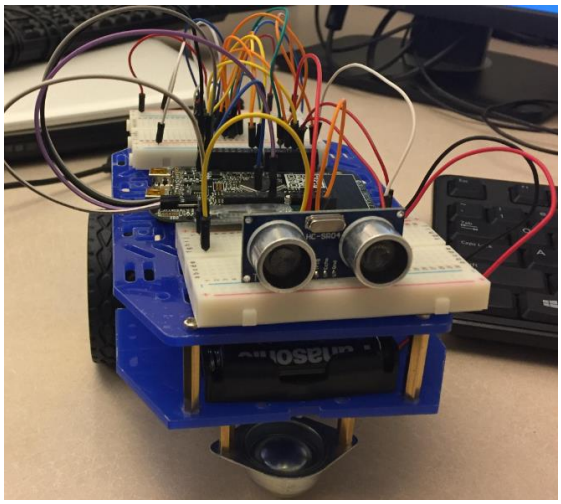

Figure 10: Hardware Implementation of ABS

5. System Testing

Testing is a disciplined process that consists of evaluating the application (including its components) behavior, performance, and robustness -- usually against expected criteria. Expected behavior, performance, and robustness should therefore be both formally described and measurable. Verification and Validation (V&V) activities focus on both the quality of the software/Hardware product and of the engineering process. Testing is most often regarded as a detective measure of quality, it is closely related to corrective measures such as debugging. In practice, embedded system engineers usually find it more productive to enact testing and debugging together, usually as an interactive process. Debugging literally means removing defects ("bugs").”

In our project we performed:

- Unit Testing: where we ensure that the individual parts are working correctly.
- Functional Testing: where the testing of the functions of component or system was done and the question “Can the user perform this “ was answered.
- Regression Testing: was to verify that modifications in the software or the environment have not caused any unintended adverse side effects and that the system still meets its requirements.

Hardware Testing involved testing of individual components by making use of the IEEE Labs:

- Processor: tested if it drew or delivered enough Voltage, Current, Power.
- Sensor: Connected the Ground and Vcc to the Oscilloscope and observed if a square wave was generated when the fingerprint was given as an input.

Developing simple Test cases/runs/plans helped in the Software testing which was further enhanced to perform Hardware Testing.

SL No	Test case ID	Test Case Name	Test description	Expected result	Actual result	Test Status
Functional Test						
1	FUNC_001	Detect_obstacle	TO verify that whether the sensor detects the obstacle	Reflected signal are captured in echo signal	Yes Reflected signals are captured	Pass
2	FUNC_002	Motor_Control	To verify that Motor rotates with the PWM signals from the MCU.	Motor should rotate	Yes the Motor rotates according to the input	Pass
3	FUNC_003	Bluetooth_Control	To verify the wireless connection between the module and the App	The App should pair with the Bluetooth Device	The App and Device connected successfully	Pass
4	FUNC_004	Bluetooth_Data	To verify the transmission of the Data between the App and Device	The Bluetooth software receives the Data	The Bluetooth transmits the data successfully	Pass
Non-functional test						
1	NONFUNC_001	Performance Test	To verify the system stops the motor within 8 seconds when interrupted by the sensor	The Motor motion stops below 8 seconds	The Motor motion stops at 5 seconds	Pass
2	NONFUNC_002	Reliability Test	To verify that the system should	The system executes the	The system was tested working for 5 times for	Pass

			perform the required operation at all the time without failure	automatic collision control every time	the period of seven days successfully	
3	NONFUNC_003	Usability Test	To verify the GUI provided to the user is clear and easy to understand	The user is able to use the system with ease and with less support	The user found the interface usable and very user friendly	Pass

Table 2: Functional and Non-Functional Test Cases

6. Development Process

This section explains the process used in the development of software for the embedded system. The project was developed with most widely used SDLC model, Waterfall Model.

Waterfall approach was first SDLC Model used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.

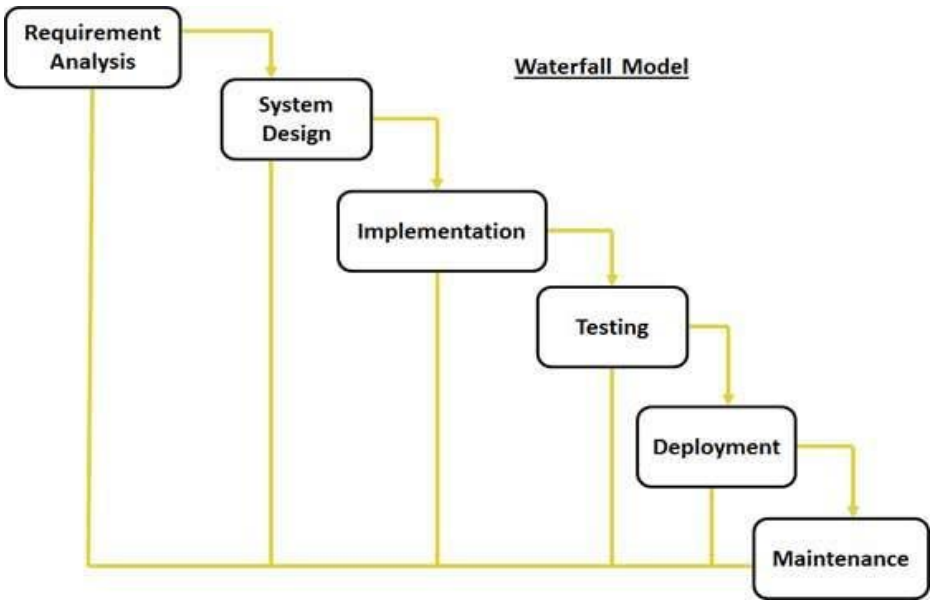

Figure 11: Software Methodology approach.

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed captured in this phase and documented in a requirement specification doc.

- **System Design:** The requirement specifications from first phase studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture.

- **Implementation:** With inputs from system design, the system first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system:** Once the functional and non-functional testing executed, the product is deployed in the customer environment.

- **Maintenance:** There are some issues, which come up in the client environment. To fix those issues patches are released. In addition, to enhance the product some better versions are released. Maintenance performed to deliver these changes in the customer environment.

7. Project Schedule

The below table shows the proposed schedule that is being used to achieve the deliverables required for our Automatic Braking System. All major tasks were identified and executed on time as a Team.

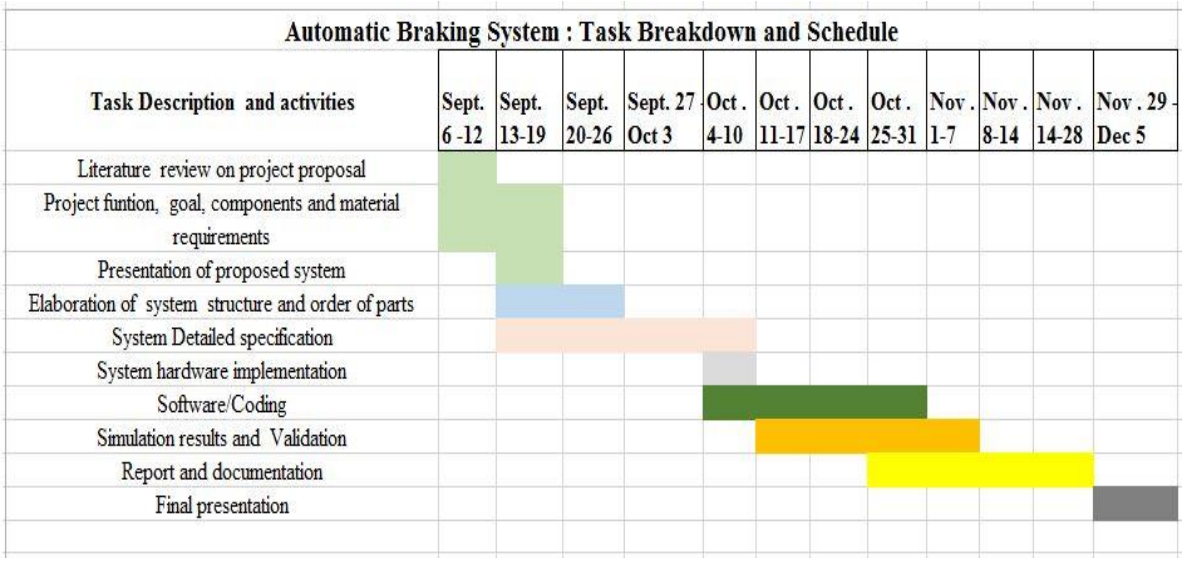

Figure 12: Breakdown of Schedule

8. Project Budget

QUANTITY	DESCRIPTION	TOTAL AMOUNT
1	**FRDM KL25Z**	$38
1	**HC-SR04 Sensor**	$6
1	**Robot Car Chassis kit**	$28
1	**Motor Driver - L293DNE**	$7
1	**HC-05, 3.6V - 6V Bluetooth Module**	$6
2	**BREADBOARD**	$6
1	**Finger BATTERY**	$18
50	**Wires**	$9
2	**Resistors**	$1
1	**Power Bank**	$24
Total		$143

Table 3: Expense Table

9. Project Repository

As a requirement for the project, we decided to use a collaborative and tracking technique too so that we can easily monitor and manage the progress we were making on our project at a specific time. We are using the Github open source as our collaborative tool that gives every member an opportunity to update on the deliverables in a flexible manner. It was very helpful in achieving our goals**.**

https://github.com/gsanthar/Automatic-Braking-System

10. Conclusion

The Primary goal of this project is to design and develop Automatic Braking System, which prevents automobile mishaps. As a Final Point, this report demonstrates successful analysis, design and implementation of the Automatic Braking System with proposed requirements. It is believed that the system created will be an important safety feature in automobile industry.

This Project utilizes various innovative technologies in hardware and software design of the embedded system. This document represents project accomplishments adhering to stringent embedded system requirements such as Cost, Power and Size.

The author is satisfied that the criteria set down at the outset of the project has been fulfilled and ultimately the project has been a success.

11. References

1. http://www.ibm.com/developerworks/rational/library/459.html
2. http://www.farnell.com/datasheets/1651277.pdf
3. https://mcuoneclipse.com/2013/01/01/tutorial-ultrasonic-ranging-with-the-freedom-board/
4. http://www.rakeshmondal.info/L293D-Motor-Driver
5. http://www.micropik.com/PDF/HCSR04.pdf
6. http://www.ti.com/lit/ds/symlink/l293.pdf
7. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

12. Appendix

12.1 Source code

```
/*
* Authors: Ganesh Santhar,Kamal,Jivanjot,Opeyemi
 * File Name: main.cpp
 * Description: Automatic Braking System functionality is implemented
 *         here with help of FreeRTOS and thread function detect_obstacle,
 *         detect_driver.
 *
 * Language:  C++
 * Version: 1.0
 * Date Created: 04/october/2016
 * Date modified: 02/december/2016
 */

#include "mbed.h"
#include "rtos.h"

#include "motordriver.h"
#include "hcsr04.h"

DigitalOut myled(LED1); /* Interface for LED device */
Serial pc(USBTX,USBRX); /* Interface for Serial Terminal */
Serial blue(D14,D15);   /* Interface for Bluetooth Module */
HCSR04  usensor(PTA12,PTD4); /* Interface for Ultrasonic Sensor */

/* Interface for dc motors */
Motor A(PTA5, PTC5, PTC4 , 1); // pwm, fwd, rev, can brake
Motor B(PTD5, PTC13, PTC12, 1); // pwm, fwd, rev, can brake

unsigned int dist, b; // global variable for interrupt signal
Thread *thread2; // Thread pointer variable for driver_motor thread


/* definition of thread functions */
void detect_obstacle();
void drive_motor();

/*
 * Funtion Name: detect_obstacle
 * Parameters: None
 * Description: A thread which runs always to detect obstacle by tracking
 *             output from sensor.
```

```
 *
 * Returns:  None
 */
void detect_obstacle()
{

      static int a=0;
       while(1){

          usensor.start();
          wait_ms(200);
          dist=usensor.get_dist_cm();// reflected signal from obstacle recorded in distance
units
          while ( dist < 12 ) {
             pc.printf("dist = %d %d\r\n",dist,a);
             a++;
             if ( a == 3 ) { // Interrupt triggered if code enters here
                b = 1;
                a = 0;
                break;
                }
          }
          a = 0;

          pc.printf("dist: %d\r\n",dist);
       }
}


/*
 * Funtion Name: drive_motor
 * Parameters: None
 * Description: A thread which drives the motor by generating PWM signal
 *              and gets input from bluetooth serial interface to
 *              trigger and stop pwm signal.
 *
 * Returns:  None
 */
void drive_motor()
{
   unsigned char inp;

   while(1) {

      if(blue.readable()>0) {
      inp=blue.getc();
```

```
        }
        if ( b == 1 ) { // Upon interrupt brakes the motor
           inp = 'E';
           b = 0;
           blue.putc(dist);
        }

        pc.printf("Input key : %c %d\r\n",inp,b);
        switch(inp) {

           case 'A': // drives motor in forward direction
              A.speed(255);
              B.speed(255);
              break;
           case 'D': // drives motor in left direction
              A.speed(255);
              B.speed(0);
              break;
           case 'C': // drives motor in Right direction
              A.speed(0);
              B.speed(255);
              break;
           case 'E': // stops motor  motiton
              A.speed(0);
              B.speed(0);
              wait(1);
              A.coast();
              B.coast();
              break;
           case 'B': // drives motor in Reverse direction
              A.speed(-255);
              B.speed(-255);
              break;
           default:
              break;

          }
        }
}
```

```
/*
 * Funtion Name: main
 * Parameters: None
 * Description: This will runs as main process and Creates the Threads detect_obstacle
 *                  and driver_motor.This function will run as along as board is up.
 *
 * Returns:  None
 */

int main ()
{
   blue.baud(9600);
   Thread thread(detect_obstacle);
   thread2 = new Thread(drive_motor);
   while (true) {
      Thread::wait(5000);
      pc.printf("Obstacle detected at %d cm \r\n",dist);
      fflush(stdout);
      }
}
```