"CS684 Embedded Systems Course Project Report, CSE Dept, IIT Bombay, November 2010".

Automated Electronic Toll Tax Collection System

Group-14

Pushkar Kothavade (09307924) pushkarbk@ee.iitb.ac.in
Mugdha Nazare (09307915) mugdhaun@ee.iitb.ac.in
Ashish Pardhi (09305049) ashishpardhi@cse.iitb.ac.in

Instructor : Prof. Kavi Arya

Department of Computer Science Engineering
Indian Institute of Technology Bombay
Mumbai -400076.

Table of Contents

Introduction

1.1 Problem Statement

Automated Electronic Toll Tax Collection System (AETC) is implanted using FIREBIRD V robots. AETC system detects vehicles passing through toll plaza and automatically deducts the toll tax from the vehicle owner's prepaid account and sends intimation to owner through SMS on GSM registered mobile number. Communication between vehicle and toll plaza takes place through ZigBee. Details of vehicle (vehicle ID, balance, number of times violation happed in the past, latest speed of the vehicle, latest timestamp) are stored in the database. AETC System takes photographs of license plates and reads the license number using OCR (Optical Character Recognition) image processing algorithm.

1.2 Requirements Specification

- USB Camera
- GSM Module with valid SIM inserted
- Minimum Two ZigBee enabled FireBird-V Robots
- Three ZigBee modules (as shown in the picture-10)
- One Windows based system with minimum three USB ports and Matlab installed
- One Linux based system with serial port through which the GSM module is accessed and Matlab installed
- GCC compiler on Linux system
- X-CTU software to program ZigBee modules
- White line with three black patches (as shown in pictures below)

1.3 System Design, Flowchart and State Chart

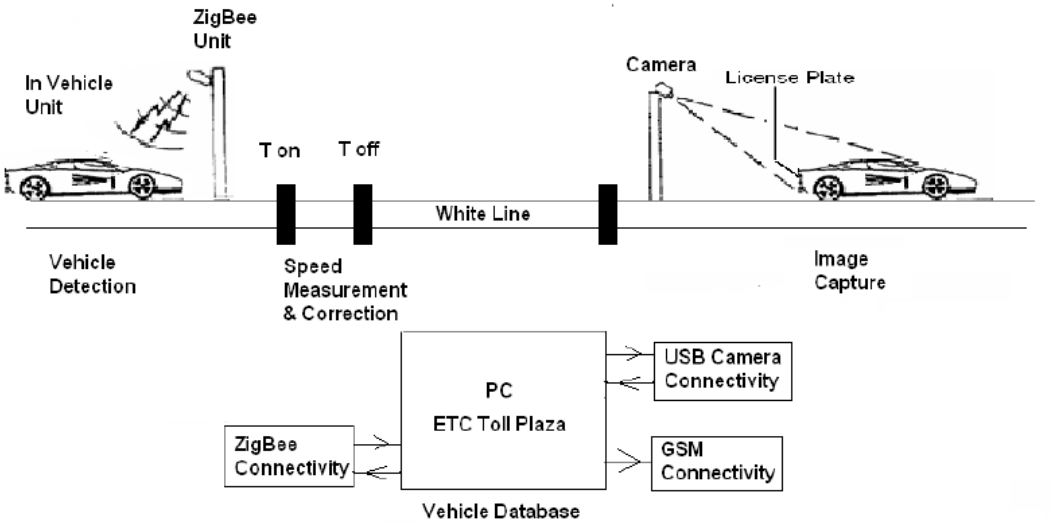


Figure 1.1: System Architecture

Working of the above system is explained with the help of flowchart in brief as below.

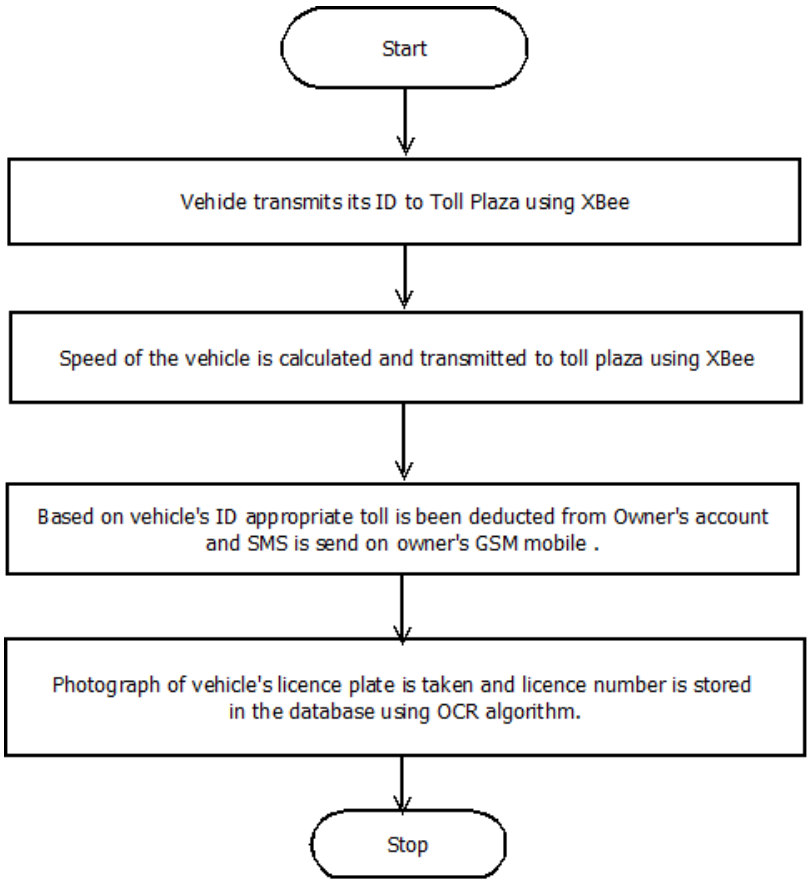


Figure 1.2: System Flowchart

- FireBird-V robots are used as vehicles.

Figure 1.3: FireBird-V Robot

- License plate is attached to one of the sides of the robot as shown in picture below.

Figure 1.4: License Plate

- Vehicles are programmed to follow white line.

Figure 1.5: White Line

- First two black patches (from left-according to system architecture) are placed for speed measurement and speed correction.
- When vehicle crosses first black patch, it starts the timer and stops the timer when it reaches to second black patch.
- Based on the timer value and known distance between the two black patches, speed of the vehicle is calculated and send to Toll plaza.

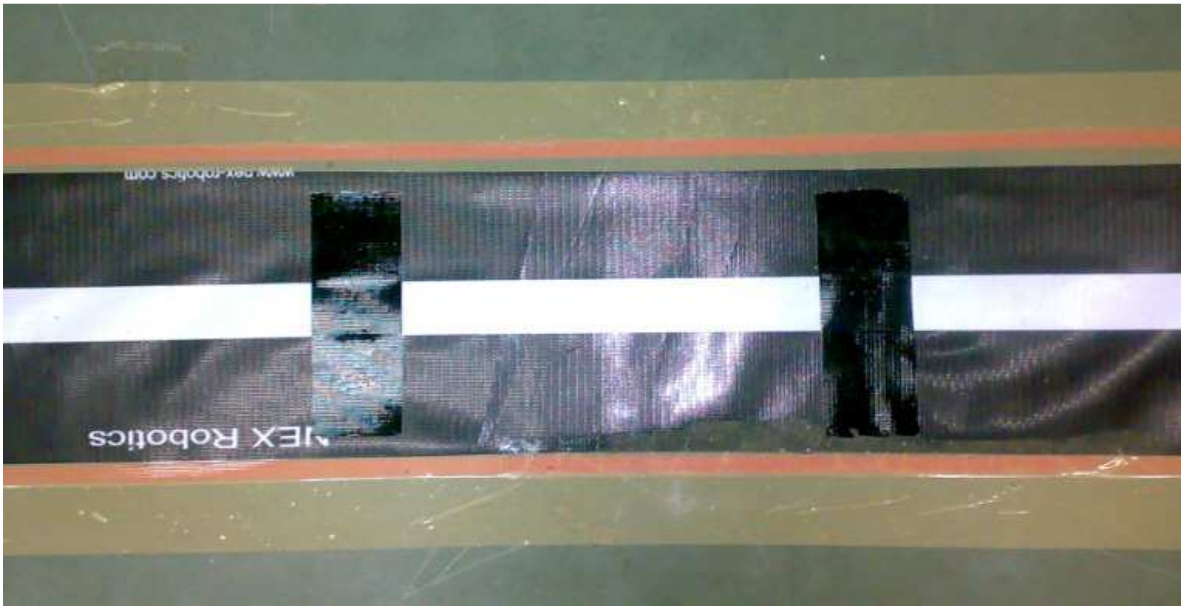


Figure 1.6: First Two Black Patches For Speed Measurement

- Third black patch is used to synchronize camera operation. Vehicle stops at third black patch then camera is activated by toll plaza to take photograph. After image is captured, toll plaza issues command to vehicle to move ahead.

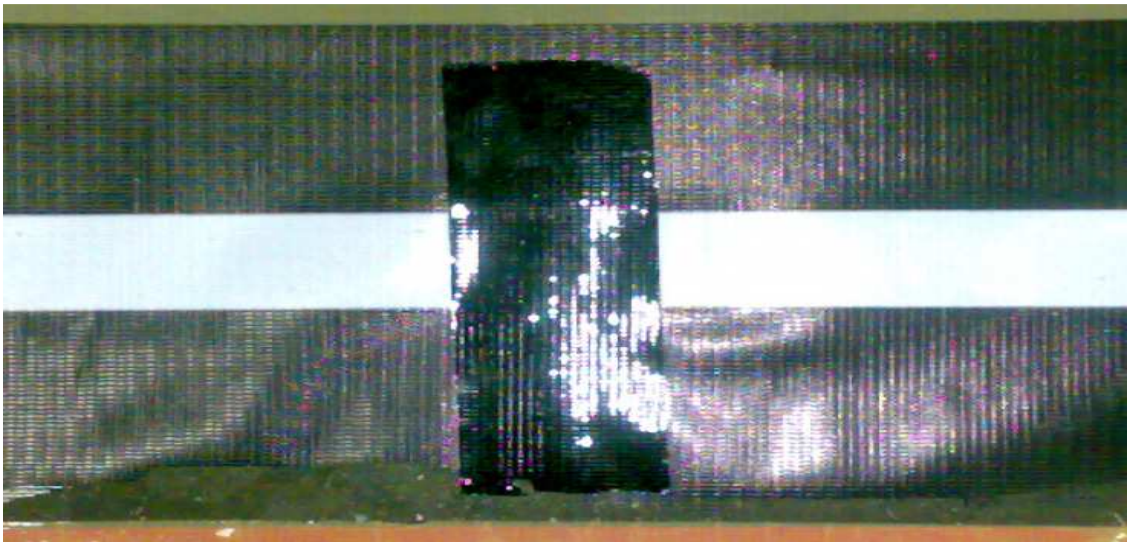


Figure 1.7: Third Black Patch For Camera Synchronization

- Toll plaza is implemented using two systems. One system runs on Windows platform and another runs on Linux platform. USB camera is attached to Windows based system.

Figure 1.8: Image Capture Mechanism (Top View)

Figure 1.9: Image Capture Mechanism (Side View)

- GSM module is connected to Linux system through serial port.
- SIM card is inserted into GSM module.
- Antenna is connected to GSM module to send and receive signals.
- AT commands document used for GSM communication is included in deliverables.

Figure 1.10: GSM Module

- ZigBee modules – One pair of ZigBee modules communicates between two systems and another pair of ZigBee modules communicates between Windows based system and ZigBee enabled transmitter receiver module unit.
- X-CTU software is used to program ZigBee modules. User manual of ZigBee is included in deliverables.

Figure 1.11: ZigBee Module

Apart from this, following things are taken off the shelf in our project.

- Image processing algorithm (GOCR) is already developed piece of code which is freeware.
- Part of the C code for FireBird-V robot has already been developed in our previous assignments.

State chart for the entire system working and meaning of the signals used in it are given below.

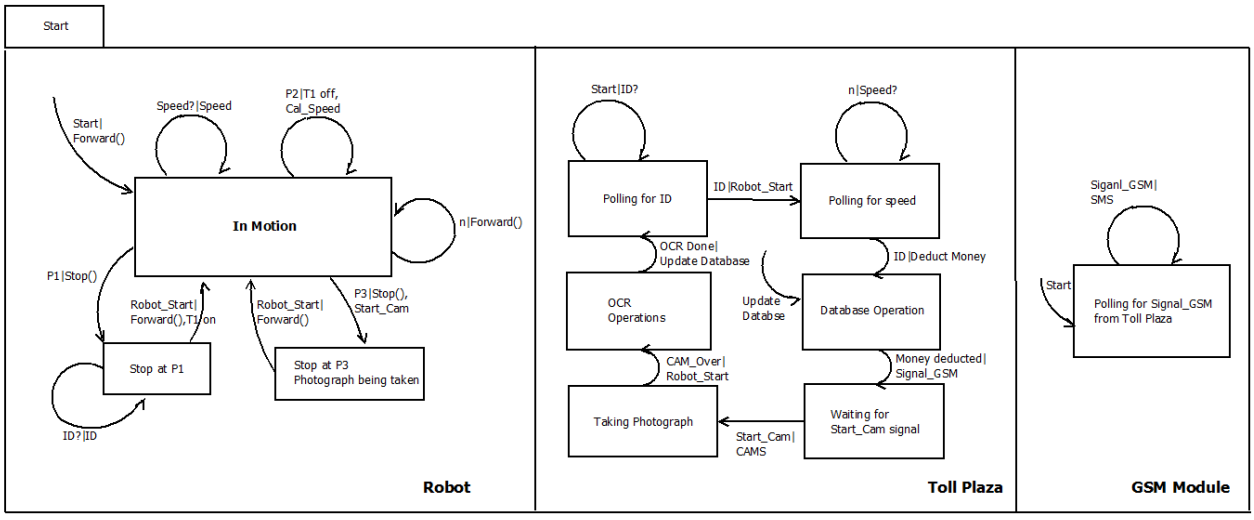


Figure 1.12: State Chart of the entire system

Siganl	Meaning
Start	Start the Robot, Toll plaza and GSM module
ID?	Toll plaza asks for Robot ID
ID	Robot sends his ID to toll plaza
Robot_Start	Toll plaza sends start command to robot
T1 on	Timer 1 on
T1 off	Timer 1 off
P1	Black patch 1 is sensed by robot
P2	Black patch 2 is sensed by robot
P3	Black patch 3 is sensed by robot
Forward()	Robot moves in the forward direction
Stop()	Robot stops
Cal_Speed	Robots calculates its speed
Speed?	Toll plaza asks for Robot Speed
Speed	Robot sends his speed to toll plaza
Start_Cam	Robot asks toll plaza to start the Camera
CAMS	Toll plaza sends start command to camera
CAM_Over	Camera indicates toll plaza that photo is been taken
Deduct Money	Deduct money from owner's account
Money deducted	Money is deducted from owner's account
Signal_GSM	Toll plaza sends signal to GSM module to send appropriate SMS on owner's mobile
Update Database	Update the database giving information regarding toll deduction
OCR Done	OCR algorithm have performed its operation
SMS	Send SMS on owner's mobile

1.4 Assumptions

- Each vehicle has ZigBee enabled transmitter receiver module.
- For this prototype, we have taken into account only GSM mobile users.
- A specific license plate and alphanumeric format is assumed.

1.5 Limitations

- Each vehicle has to stop at first black patch till the time Toll plaza allows it to go in order to synchronize with Matlab based operations.
- Each vehicle has to stop at third black patch in order to synchronize camera operation. This limitation can be overcome by using advanced camera which can capture pictures of moving object.
- Accuracy of image processing algorithm depends on light conditions

2. Present Status

- We have successfully demonstrated the working system and have achieved all proposed tasks.

2.1 Individual Roles and Contibutions

Task List and Work Division					
Task	**Person**	**Update**	**Challenges**	**Status**	**Completion Date**
Image Processing of License Plate	All	Test-1: Testing Of OCR Algorithm In Matlab	Works Fine On Computer Generated Images	Done	1/10/2010
Image Processing of License Plate	All	Test-2: Testing Of OCR Algorithm In Actual Scenario	Problems With Detection	Done	28/10/2010
Documentation	All	First Draft Is Submitted	-	Done	22/09/2010
Presentation	All	Done With First Presentation	-	Done	27/09/2010
Setup Preparation and Test Plan Execution	All	Setup Is Ready	-	Done	29/10/2010
Implementation of Toll Plaza on PC Using MATLAB	Pushkar Kothavade	We Will Upload Version-1.0 Of Code On SVN soon	-	Done	19/10/2010
Implementation of Database in MS-Excel	Pushkar Kothavade	Matlab Can Automatically Read and Write to MS-Excel Sheet	-	Done	25/09/2010
Access to ZigBee Module (Serial Communication)	Pushkar Kothavade	Matlab Can Access ZigBee Module	-	Done	30/09/2010
Access to Camera (USB Communication)	Pushkar Kothavade	Image Capture Mechanism Using USB Camera	-	Done	03/10/2010
Programming of Vehicle	Mugdha Nazare	Almost Done	-	Done	29/10/2010
Speed Calculation using Timer	Mugdha Nazare	Black Patch Detection & Speed Measurement	-	Done	02/10/2010
White Line Follower Code	Mugdha Nazare	Implemented White Line Follower Code With Black Patch Detection	-	Done	04/10/2010
ZigBee communication (Vehicle Side))	Mugdha Nazare	Tested ZigBee Communication Between Vehicle and Toll plaza	-	Done	07/10/2010
Display on LCD code	Mugdha Nazare	It Is Working Fine	-	Done	12/10/2010
Implementation Database management System	Ashish Pardhi	Integration of Database With System	-	Done	23/10/2010
Implementation of GSM Communication	Ashish Pardhi	Basic testing on GSM Module Using HyperTerminal	-	Done	8/10/2010
Implementation of GSM Communication	Pushkar Kothavade	C code written to access GSM Module	-	Done	11/10/2010
Implementation of GSM Communication	Ashish Pardhi	Integration of C code into MATLAB using MEX File	Not Working	Failed	17/10/2010
Implementation of GSM Communication	Pushkar Kothavade	Tried Shell Scripting Using Command Line Terminal Software	Not Working	Failed	19/10/2010
Implementation of GSM Communication	Pushkar, Mugdha, Ashish	Planning To Implement GSM C code on Separate Machine	-	Done	2/11/2010

2.2 Implementation Issues

- Integration and synchronization of all the modules such as GSM on Linux, Toll Plaza on windows and Robot was a difficult task, which took considerable amount of time.
- We used several approaches for Incorporation of GSM module. First we tried Matlab to send AT commands to GSM module but GSM module did not respond to commands.
- Second approach we considered was integration of C code into Matlab using MEX function utility, which failed as well. So finally we decided to split toll plaza functionality into two systems, one system is Windows based and other one is Linux based.

2.3 Human Resource

- We have put up 50 man days for this project.

3. Innovation and Creativity in the project

- This is a prototype of the system which demonstrates the feasibility of the automated toll tax collection system. We look forward to implement this system in real world scenario on Indian roadways.

4. Reusability Index of Project

- We encourage interested students to use this basic prototype and build up a more improvised system.
- For their convenience, all the project code is made available along with a readme file which will guide them to make the project running at their end. Also, codes are well commented and are divided into modules which are easy to understand and can be reused independently.
- Equipments used in the project such as robots (ERTS lab, IITB and NexRobotics property), XBee module, etc are easily available in market.

5. Feedback

- It was a good experience to work on already built in and reusable robots made available through ERTS lab, IITB and NexRobotics which made building up the AETC prototype easier.

6. Future Scope

- This system can be implemented on Indian road ways with appropriate modifications and with more sophisticated equipments.
- Instead of Matlab one can consider other software which takes less computation time.
- One can develop mobile based application which can accept AT commands so that we can replace GSM module with mobile.
- More advanced camera which can take clear pictures of moving objects can be used in the project so that vehicles need not stop at third black patch.

7. Learnings

- It was a good experience to work on already built in and reusable robots made available through ERTS lab, IITB and NexRobotics which made building up the AETC prototype easier.
- Got a confidence in robot programming and system level design.
- Learned to use Linux APIs in C program.
- GSM and ZigBee modules programming.

8. Appendices

8.1. Readme

Project:

The objective of this document is to help someone else run the code that is delivered as part of this project.

Project Title: Automated Electronic Toll Tax Collection System

Students: Group14

Name	Roll No.	Email
Pushkar Kothavade	09307924	pushkarbk@ee.iitb.ac.in
Mugdha Nazare	09307915	mugdhaun@ee.iitb.ac.in
Ashish Pardhi	09305049	ashishpardhi@cse.iitb.ac.in

Project Objective

The objective is to make prototype of an Automated Electronic Toll Tax Collection (AETC) system using FIREBIRD V robots. This system detects vehicles passing through toll plaza and automatically deducts the toll tax from the vehicle owner's prepaid account. The same will be intimated to owner through SMS on GSM registered mobile number. Also, this system stores the licence plate number of every vehicle using image processing and calculates the speed of the vehicle.

AETC is the effective solution to avoid traffic jams and to maintain the transparency in transactions at toll plazas across India.

Hardware Platform

1. Firebird V ATMEGA2560
2. GSM module (Serial port communication)
3. XBee communication

Software

1. AVR Studio 4
2. Matlab
3. Gcc complier

Code Description

Code Files

Filename	Purpose	Executes on
AETC_Robot.c	Main Program	Robot
AETC_lcd.c	Contains the abstractions of major operations.	Robot
AETC_TollPlaza.m	Main Program	Toll Plaza PC
AETC_GSM.m	Main Program	GSM Communication system PC
AETC_GSMS.c	Contains message to be sent on a particular number when toll is deducted successfully	GSM Communication system PC
AETC_GSMF.c	Contains message to be sent on a particular number when toll is not deducted successfully	GSM Communication system PC
AETC_Record.xls **AETC_LicNum.xls**	Contains the ID, account balance, timestamp of last time vehicle has crossed the toll plaza and no. of times the vehicle has violated the rule.	Toll Plaza PC

Deliverables

Filename	Contains
C-code.tar.gz	SourceCode of programs to be burnt on Robot.
TollPlaza_PC-interface.tar.gz	Contains Matlab and xls file.
GSM_PC-Interface.tar.gz	Contains Matlab and C files.
ProjectDocuments.tar.gz	Contains Project related doc files.

Execution Instructions

1) Compile the C code AETC_Robot.c given in the zip folder C-code.tar.gz and burn it on Firebird V robot using AVR software interface.

2) Make a white line strip which has two black patches slightly away from the starting position of the robot. The distance between the two black patches can be around 25 to 30 cm. Also keep the third black patch at a sufficient distance from the earlier two black patches. Mount the camera near this third black patch. You will have to adjust the orientation of camera according to the place where the robot would actually stop on this black patch. Also, make sure that light conditions are properly set for the camera so that gocr algorithm will not fail. The setup is shown in the figure on next page for you reference.

Figure 8.1: Image Capture Mechanism

3) Run the matlab code AETC_TollPlaza.m given in the zip folder TollPlaza_PC-interface.tar.gz on windows machine simultaneously with the code AETC_GSM.m given in zip folder GSM_PC-Interface.tar.gz on Linux machine which has a GSM module attached via serial port. Make sure that the two C codes given in GSM_PC-Interface.tar.gz are kept in the same workspace from where Matlab (Linux) is invoked.

In order to check every functionality written in code, we advise you to place robot on white line and start before starting Matlab.

4) You may wish to send multiple robots one after the other. In such case, ensure that you are changing the ID of the robot during programming in AETC_Robot.c file. The ID's should be same as those mentioned in the file AETC_Record.xls which is kept in the same Matlab (Windows) workspace. A sample file is attached in the zip folder TollPlaza_PC-interface.tar.gz.

5) Output of the gocr algorithm can be observed in the file AETC_LicNum.xls which is generated by matlab (Windows) code in the same workspace. Deduction of toll, no. of times the vehicle has violated the rule, speed of the vehicle, timestamp of vehicle last crossing toll plaza can be observed in the AETC_Record.xls file.

6) Make sure that XBee module used for communication are programmed in unicast mode in order to avoid interference of the other XBee module running in parallel. This can be done using a software X-CTU

Coding Guidelines

Please find attached a zip file with this document. Please refer to this code to write your own code.

8.2 Program Source Code
8.2.1.AETC_Robot.c

```
/**
 * @file AETC_Robot.c
 * Program for speed calculation, white line follower and obstacle detection.
 * @author Puskar Kothavade, Ashish Pardhi, Mugdha Nazare, IIT Bombay
 * @date 10/Oct/2010
 * @version 1.0
 *
 * @section LICENSE
  * Copyright (c) 2010. ERTS Lab IIT Bombay
   * All rights reserved.

  * Redistribution and use in source and binary forms, with or without
   *modification, are permitted provided that the following conditions are met:

   * Redistributions of source code must retain the above copyright
    * notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above copyright
   * notice, this list of conditions and the following disclaimer in
   * the documentation and/or other materials provided with the distribution.

   * Neither the name of the copyright holders nor the names of
   * contributors may be used to endorse or promote products derived
   * from this software without specific prior written permission.

   * Source code can be used for academic purpose.
   * For commercial use permission form the author needs to be taken.
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/signal.h>
#include <math.h>
#include "AETC_lcd.c"

#define FCPU 11059200ul ///< defined here to make sure that program works properly
/**
* Functions prototype
*/
```

```
void port_init();
void timer5_init();


void velocity(unsigned char, unsigned char);
void motors_delay();
void timer1_init();
unsigned char ADC_Conversion(unsigned char);
/**
* Global Variables
*/

unsigned char ADC_Value;
unsigned char flag1 = 0, flag2 = 0, f = 0; ///< stores flag values
unsigned char Left_white_line = 0;///< store left white line sensor value
unsigned char Center_white_line = 0;///< store center white line sensor value
unsigned char Right_white_line = 0;///<  store right white line sensor value
unsigned char Front_Sharp_Sensor = 0;///< store front sensor value
unsigned char Front_IR_Sensor = 0;
static unsigned int speed_cnt = 0;///< store speed count
unsigned int vehicle_stop = 0;
static unsigned int timer_flag = 0;
unsigned int LID_Transmit = 0;///< store transmitted ID value

unsigned char data;///< store data
float speed;///< store speed
unsigned char speed_int, speed_dec;
unsigned int RegSpeed = 150;
unsigned char StartTheBot = 0;

/*
* ISR Routine to Increment the speed counter
*/
ISR(TIMER1_OVF_vect)
{
        speed_cnt++;
}

/*
* Function to configure LCD port
*/
void lcd_port_config (void)
{
 DDRC = DDRC | 0xF7; //all the LCD pin's direction set as output
 PORTC = PORTC & 0x80; // all the LCD pins are set to logic 0 except PORTC 7
}
```

```
/*
* ADC pin configuration
*/
void adc_pin_config (void)
{
 DDRF = 0x00;
 PORTF = 0x00;
 DDRK = 0x00;
 PORTK = 0x00;
}

/*
* Function to configure ports to enable robot's motion
*/
void motion_pin_config (void)
{
 DDRA = DDRA | 0x0F;
 PORTA = PORTA & 0xF0;
 DDRL = DDRL | 0x18;   //Setting PL3 and PL4 pins as output for PWM generation
 PORTL = PORTL | 0x18; //PL3 and PL4 pins are for velocity control using PWM.
}

/*
* Function to Initialize PORTS
*/
void port_init()
{
        lcd_port_config();
        adc_pin_config();
        motion_pin_config();
}

/*
* Timer 5 initialised in PWM mode for velocity control
* Prescale:64
* PWM 8bit fast, TOP=0x00FF
* Timer Frequency:674.988Hz
*/
void timer5_init()
{
        TCCR5B = 0x00;          ///< Stop
        TCNT5H = 0xFF;          ///< Counter higher 8-bit value to which OCR5xH value is compared
with
```

```
	TCNT5L = 0x01;		///< Counter lower 8-bit value to which OCR5xH value is compared
with


	OCR5AH = 0x00;		///< Output compare register high value for Left Motor
	OCR5AL = 0xFF;		///< Output compare register low value for Left Motor
	OCR5BH = 0x00;		///< Output compare register high value for Right Motor
	OCR5BL = 0xFF;		///< Output compare register low value for Right Motor
	OCR5CH = 0x00;		///< Output compare register high value for Motor C1
	OCR5CL = 0xFF;		///< Output compare register low value for Motor C1
	TCCR5A = 0xA9;		///< {COM5A1=1, COM5A0=0; COM5B1=1, COM5B0=0; COM5C1=1
COM5C0=0}  For Overriding normal port functionalit to OCRnA outputs. {WGM51=0, WGM50=1}
Along With WGM52 in TCCR5B for Selecting FAST PWM 8-bit Mode

	TCCR5B = 0x0B;		///< WGM12=1; CS12=0, CS11=1, CS10=1 (Prescaler=64)
}

/*
* Function For Timer1 Initialisation
*/

void timer1_init()
{
	TCCR1B = 0x00;  ///< STOP
	TCNT1H = 0x00;		///< Timer register higher 8 bit to zero
	TCNT1L = 0x00;		///< Timer register lower 8 bit to zero
	OCR1AH = 0xFF;		///< Output compare register higher 8 bits...not used in this case
	OCR1AL = 0xFF;		///< Output compare register lower 8 bits.....not used in this case
	OCR1BH = 0x00;		///< Output compare register higher 8 bits...not used in this case
	OCR1BL = 0x00;		///< Output compare register lower 8 bits.....not used in this case
	OCR1CH = 0x00;		///< Output compare register higher 8 bits...not used in this case
	OCR1CL = 0x00;		///< Output compare register lower 8 bits.....not used in this case
	TCCR1A = 0x00;		///< Using channel A only, but not using output compare mode.
	WGM: normal mode COM10:1=00
	TCCR1C = 0x00;		///< No FOC
	TIMSK1 = 0x01;		///< Timer overflow interrupt enabled

	SREG = SREG | 0x80; ///< Setting I flag of status register to one to globally enable all
interrupts

}

/*
* Function For ADC Initialisation
*/
void adc_init()
{
```

```
	ADCSRA = 0x00;
	ADCSRB = 0x00;			///< MUX5 = 0


	ADMUX = 0x20;			///< Vref=5V external --- ADLAR=1 --- MUX4:0 = 0000
	ACSR = 0x80;
	ADCSRA = 0x86;			///< ADEN=1 --- ADIE=1 --- ADPS2:0 = 1 1 0
}




/*
* Function For ADC Conversion
*/
unsigned char ADC_Conversion(unsigned char Ch)
{
	unsigned char a;
	if(Ch>7)
	{
		ADCSRB = 0x08;
	}
	Ch = Ch & 0x07;
	ADMUX= 0x20| Ch;
	ADCSRA = ADCSRA | 0x40;		///< Set start conversion bit
	while((ADCSRA&0x10)==0);	///< Wait for conversion to complete
	a=ADCH;
	ADCSRA = ADCSRA|0x10; ///< clear ADIF (ADC Interrupt Flag) by writing 1 to it
	ADCSRB = 0x00;
	return a;
}

/**
* Function To Print Sesor Values At Desired Row And Coloumn Location on LCD
*/

void print_sensor(char row, char coloumn,unsigned char channel)
{

	ADC_Value = ADC_Conversion(channel);
	lcd_print(row, coloumn, ADC_Value, 3);
}

/**
* Function for velocity control.
*/
```

```
void velocity (unsigned char left_motor, unsigned char right_motor)
{


        OCR5AL = (unsigned char)left_motor;
        OCR5BL = (unsigned char)right_motor;
}

/**
* Function used for setting motor's direction.
*/
void motion_set (unsigned char Direction)
{
 unsigned char PortARestore = 0;

 Direction &= 0x0F;              ///<  removing upper nibbel for the protection
 PortARestore = PORTA;                  ///<  reading the PORTA original status
 PortARestore &= 0xF0;                  ///< making lower direction nibbel to 0
 PortARestore |= Direction; ///< adding lower nibbel for forward command and restoring the
PORTA status
 PORTA = PortARestore;                  ///< executing the command
}

void forward (void)
{
  motion_set (0x06);
}

void stop (void)
{
  motion_set (0x00);
}


/**
* Function To Initialize UART0
* desired baud rate:9600
* actual baud rate:9600 (0.0%)
* char size: 8 bit
* parity: Disabled
*/
void uart0_init(void)
{
 UCSR0B = 0x00; ///< disable while setting baud rate
 UCSR0A = 0x00;
 UCSR0C = 0x06;
 UBRR0L = 0x47; ///< set baud rate lo
```

```
 UBRR0H = 0x00; ///< set baud rate hi
 UCSR0B = 0x98;


}

/*
* Function For Devices Initialisation
*/
void init_devices (void)
{
        cli(); ///< Clears the global interrupts
        port_init(); ///< initialise the ports
        adc_init();
        uart0_init();
        timer5_init();
        timer1_init();
        sei();   //Enables the global interrupts
}

/**
* Usart receiver ISR
*/

SIGNAL(SIG_USART0_RECV)
{
        unsigned char data;
        data = UDR0; ///< making copy of data from UDR0 in data variable

        if(data == 'Z' && timer_flag==1)
        {
                StartTheBot=1;
        }

        if(data =='O')                          ///< To Regulate the speed
        {
                forward();
                RegSpeed = 150;
        }

        if(data =='I')                          ///< To transmit ID to Laptop
        {
                if (LID_Transmit == 0 && timer_flag==1)
                {
                   UDR0 = '2' ;       /// Change this number for different vehicles
                LID_Transmit = 1;
                }
```

```
	}


	if(data == 'X')                                  ///< To transmit speed int value
	{
		if(timer_flag>=3)
		{
			UDR0 = speed_int;
		}
	}

	if(data == 'Y')                             ///< To transmit speed value after decimal point
	{
		if(timer_flag>=3)
		{
			UDR0 = speed_dec;
		}
	}

	if(data == 'V')              ///< IF third black patch comes then transmit 1 otherwise 0.
	{
		if(timer_flag==4)
		{
			UDR0 = '1';
		}
	}
	if(data == 'D')               ///< Image capture process over. Vehicle can go ahead now.
	{
		if(timer_flag==4)
		{
			vehicle_stop=1;
		}
	}
}

/**
* Main Function
*/
int main()
{
	init_devices();
	lcd_set_4bit();
	lcd_init();
	while(1)
	{
```

```
Left_white_line = ADC_Conversion(3);          ///< Getting data of Left WL Sensor
Center_white_line = ADC_Conversion(2);///< Getting data of Center WL Sensor


Right_white_line = ADC_Conversion(1);        ///< Getting data of Right WL Sensor
Front_Sharp_Sensor = ADC_Conversion(11);///< Getting data of Front Sharp sensor
Front_IR_Sensor = ADC_Conversion(6);    ///<  Getting data of Front IR sensor


flag1=0;
flag2=0;


print_sensor(1,1,3);    ///< Prints value of White Line Sensor1
print_sensor(1,5,2);    ///< Prints Value of White Line Sensor2
print_sensor(1,9,1);    ///< Prints Value of White Line Sensor3
lcd_print(1, 13, timer_flag, 4);

if((Center_white_line<0x28))
{
        flag1=1;
        forward();
        velocity(RegSpeed,RegSpeed);
}


if((Left_white_line>0x28) && (flag1==0))
{
        flag1=1;
        forward();
        velocity(100,50);
}


if((Right_white_line>0x28) && (flag1==0))
{
        flag1=1;
        forward();
        velocity(50,100);
}


        if((Center_white_line>0x28) && (Left_white_line>0x28) &&
(Right_white_line>0x28))
        {

                forward();
```

```
                velocity(RegSpeed,RegSpeed);
                if (timer_flag == 0)            ///< vehicle at first black patch


                {
                        timer_flag=1;
                        while(StartTheBot==0)
                        {
                                stop();
                        }

                        if (StartTheBot == 1)
                        {
                                forward();
                                velocity(RegSpeed,RegSpeed);
                                TCCR1B = 0x01;
  ///< Timer 1 start witrh  no prescaler
                                while(Center_white_line>0x28 && Left_white_line>0x28 &&
Right_white_line>0x28)//wait till the time the entire red line is crossed.
                                {
                                        Left_white_line = ADC_Conversion(3);
///< Getting data of Left WL Sensor
                                        Center_white_line = ADC_Conversion(2);
///< Getting data of Center WL Sensor
                                        Right_white_line = ADC_Conversion(1);
///< Getting data of Right WL Sensor
                                }

                                timer_flag=2;
                        }
                }
                else if (timer_flag == 2)       ///< vehicle at second black patch
                {
                        TCCR1B = 0x00;   ///< Timer 1 stop
                        timer_flag=3;
                        lcd_print(2, 1, TCNT1H, 4);
                        lcd_print(2, 6, TCNT1L, 4);
                        lcd_print(2, 11, speed_cnt, 5);
                        speed = 27/ (((unsigned int) speed_cnt * 65536 * 90.9E-9) +
((unsigned int) TCNT1H * 256 * 90.9E-9) + ((unsigned int) TCNT1L * 90.9E-9));
                        f=1;
                        speed_int= speed;
                        speed_dec=(speed-speed_int)*100;

                }

                else if (timer_flag ==3)        ///< vehicle at third black patch
```

```
                        {
                                timer_flag=4;

                                while (vehicle_stop==0)    ///< Capturing photo in progress
                                {
                                        stop();
                                }

                                if(vehicle_stop==1)
  ///< photograph taken. Vehicle can go ahead now
                                {
                                        forward();
                                }

                        }
                }


                if((Front_Sharp_Sensor>0x80)|| (Front_IR_Sensor<0xF0)) ///< Obstacle detection
                {
                        flag2=1;
                        stop();
                }


        }
}
/**
* End of Program.
*/
```

8.2.2 AETC_lcd.c

```
/**
 * @file AETC_lcd.c
 * Program for printing data on LCD.
 * @author Puskar Kothavade, Ashish Pardhi, Mugdha Nazare, IIT Bombay
 * @date 10/Oct/2010
 * @version 1.0
 *
 * @section LICENSE
  * Copyright (c) 2010. ERTS Lab IIT Bombay
   * All rights reserved.

  * Redistribution and use in source and binary forms, with or without
   *modification, are permitted provided that the following conditions are met:

   * Redistributions of source code must retain the above copyright
    * notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above copyright
   * notice, this list of conditions and the following disclaimer in
   * the documentation and/or other materials provided with the distribution.

   * Neither the name of the copyright holders nor the names of
   * contributors may be used to endorse or promote products derived
   * from this software without specific prior written permission.

   * Source code can be used for academic purpose.
   * For commercial use permission form the author needs to be taken.
 */

#include <avr/io.h>
#include <avr/delay.h>
#include <util/delay.h>

/// set the cpu frequency
#define FCPU 11059200ul
/// set command input
#define RS 0
/// writing to LCD
#define RW 1
/// set Enable pin
#define EN 2
```

```
#define lcd_port PORTC



#define sbit(reg,bit)    reg |= (1<<bit)
#define cbit(reg,bit)    reg &= ~(1<<bit)

void init_ports();
void lcd_reset_4bit();
void lcd_init();
void lcd_wr_command(unsigned char);
void lcd_wr_char(char);
void lcd_home();
void lcd_cursor(char, char);
void lcd_print(char, char, unsigned int, int);
void lcd_string(char*);

unsigned int temp;
unsigned int unit;
unsigned int tens;
unsigned int hundred;
unsigned int thousand;
unsigned int million;

int i;


/**
* Function to Reset LCD
*/

void lcd_set_4bit()
{
        _delay_ms(1);

        cbit(lcd_port,RS);                      //RS=0 --- Command Input
        cbit(lcd_port,RW);                      //RW=0 --- Writing to LCD
        lcd_port = 0x30;                        //Sending 3
        sbit(lcd_port,EN);                      //Set Enable Pin
        _delay_ms(5);                           //Delay
        cbit(lcd_port,EN);                      //Clear Enable Pin

        _delay_ms(1);

        cbit(lcd_port,RS);                      //RS=0 --- Command Input
        cbit(lcd_port,RW);                      //RW=0 --- Writing to LCD
        lcd_port = 0x30;                        //Sending 3
```

```
        sbit(lcd_port,EN);                  //Set Enable Pin
        _delay_ms(5);                       //Delay


        cbit(lcd_port,EN);                  //Clear Enable Pin

        _delay_ms(1);

        cbit(lcd_port,RS);                  //RS=0 --- Command Input
        cbit(lcd_port,RW);                  //RW=0 --- Writing to LCD
        lcd_port = 0x30;                    //Sending 3
        sbit(lcd_port,EN);                  //Set Enable Pin
        _delay_ms(5);                       //Delay
        cbit(lcd_port,EN);                  //Clear Enable Pin

        _delay_ms(1);

        cbit(lcd_port,RS);                  //RS=0 --- Command Input
        cbit(lcd_port,RW);                  //RW=0 --- Writing to LCD
        lcd_port = 0x20;                    //Sending 2 to initialise LCD 4-bit mode
        sbit(lcd_port,EN);                  //Set Enable Pin
        _delay_ms(5);                       //Delay
        cbit(lcd_port,EN);                  //Clear Enable Pin


}

/**
* Function to Initialize LCD
*/
void lcd_init()
{
        _delay_ms(1);

        lcd_wr_command(0x28);               //LCD 4-bit mode and 2 lines.
        lcd_wr_command(0x01);
        lcd_wr_command(0x06);
        lcd_wr_command(0x0E);
        lcd_wr_command(0x80);

}


/**
* Function to Write Command on LCD
*/
void lcd_wr_command(unsigned char cmd)
```

```
{
        unsigned char temp;


        temp = cmd;
        temp = temp & 0xF0;
        lcd_port &= 0x0F;
        lcd_port |= temp;
        cbit(lcd_port,RS);
        cbit(lcd_port,RW);
        sbit(lcd_port,EN);
        _delay_ms(5);
        cbit(lcd_port,EN);

        cmd = cmd & 0x0F;
        cmd = cmd<<4;
        lcd_port &= 0x0F;
        lcd_port |= cmd;
        cbit(lcd_port,RS);
        cbit(lcd_port,RW);
        sbit(lcd_port,EN);
        _delay_ms(5);
        cbit(lcd_port,EN);
}

/**
* Function to Write Data on LCD
*/
void lcd_wr_char(char letter)
{
        char temp;
        temp = letter;
        temp = (temp & 0xF0);
        lcd_port &= 0x0F;
        lcd_port |= temp;
        sbit(lcd_port,RS);
        cbit(lcd_port,RW);
        sbit(lcd_port,EN);
        _delay_ms(5);
        cbit(lcd_port,EN);

        letter = letter & 0x0F;
        letter = letter<<4;
        lcd_port &= 0x0F;
        lcd_port |= letter;
        sbit(lcd_port,RS);
        cbit(lcd_port,RW);
```

```
	sbit(lcd_port,EN);
	_delay_ms(5);


	cbit(lcd_port,EN);
}

void lcd_home()
{
	lcd_wr_command(0x80);
}


/**
* Function to Print String on LCD
*/
void lcd_string(char *str)
{
	while(*str != '\0')
	{
		lcd_wr_char(*str);
		str++;
	}
}

/**
* Position the LCD cursor at "row", "column".
* @param row no. of rows
* @param column no. of columns
*/

void lcd_cursor (char row, char column)
{
	switch (row) {
		case 1: lcd_wr_command (0x80 + column - 1); break;
		case 2: lcd_wr_command (0xc0 + column - 1); break;
		case 3: lcd_wr_command (0x94 + column - 1); break;
		case 4: lcd_wr_command (0xd4 + column - 1); break;
		default: break;
	}
}

/**
* Function To Print Any input value upto the desired digit on LCD
*/
void lcd_print (char row, char coloumn, unsigned int value, int digits)
{
```

```
unsigned char flag=0;
if(row==0||coloumn==0)


{
        lcd_home();
}
else
{
        lcd_cursor(row,coloumn);
}
if(digits==5 || flag==1)
{
        million=value/10000+48;
        lcd_wr_char(million);
        flag=1;
}
if(digits==4 || flag==1)
{
        temp = value/1000;
        thousand = temp%10 + 48;
        lcd_wr_char(thousand);
        flag=1;
}
if(digits==3 || flag==1)
{
        temp = value/100;
        hundred = temp%10 + 48;
        lcd_wr_char(hundred);
        flag=1;
}
if(digits==2 || flag==1)
{
        temp = value/10;
        tens = temp%10 + 48;
        lcd_wr_char(tens);
        flag=1;
}
if(digits==1 || flag==1)
{
        unit = value%10 + 48;
        lcd_wr_char(unit);
}
if(digits>5)
{
        lcd_wr_char('E');
}
```

```
}


/**
* End of Program.
*/
```

8.2.3 AETC_TollPlaza.m

```
%%-------------------------------------------------------------------------------
% Code_TollPlaza.m:- Main Program for ZIGBEE Communication
% Between TollPlaza and Vehicle
%
%-------------------------------------------------------------------------------
% Author:
%                 Puskar Kothavade
%                 Ashish Pardhi
%                 Mugdha Nazare
%-------------------------------------------------------------------------------

%   Copyright (c) 2010. ERTS Lab IIT Bombay
%   All rights reserved.

%   Redistribution and use in source and binary forms, with or without
%   modification, are permitted provided that the following conditions are met:

%   * Redistributions of source code must retain the above copyright
%     notice, this list of conditions and the following disclaimer.
%
%   * Redistributions in binary form must reproduce the above copyright
%     notice, this list of conditions and the following disclaimer in
%     the documentation and/or other materials provided with the
%     distribution.

%   * Neither the name of the copyright holders nor the names of
%     contributors may be used to endorse or promote products derived
%     from this software without specific prior written permission.

%   * Source code can be used for academic purpose.
%         For commercial use permission form the author needs to be taken.
%-------------------------------------------------------------------------------
```

```
%%%%%%%%%%%% Toll Plaza (Windows Based Computer) %%%%%%%%%%%%%
%% Start %%

clc;                        % Clear Screen.
warning off all;                % Do Not Show Any Warnings.
c = '0'                     % Initialise Counter.
while(1)                    % Run Following Program Continuously.


%% Communication Module Through ZigBee %%

s = serial('COM13');                % Define COM Port Object To Communicate To Vehicle.
fopen(s);                   % Open COM port.
id = '';                    % Initialise id Variable.
w = size(id);                   % Find Out Whether id Variable is empty or Non-empty
while ( w(1) == 0)
fprintf(s,'%c','I');            % Send character 'I' to vehicle to get the Vehicle ID.
id = fscanf(s)                  % Read vehicle ID.
w = size(id);
end
fprintf(s,'%c','Z');            % Toll plaza starts communicating with vehicle
Start = 1
speed1 = '';                    % Procedure To Calculate Speed Of The Vehicle.
m = size(speed1);
while( m(1)== 0)
fprintf(s,'%c','X');
speed1 = fscanf(s);
m = size(speed1);
end

speed2 = '';
n = size(speed2);
while( n(1)== 0)
fprintf(s,'%c','Y');
speed2 = fscanf(s);
n = size(speed2);
end

speed1 = speed1 - 48 + 48           % Speed1 Is Number Left Of Decimal Point.
speed2 = speed2 - 48 + 48           % Speed2 Is Number Right Of Decimal Point.
TotalSpeed = speed1 + (0.01 * speed2)   % TotalSpeed = Speed1.Speed2

Port2 = serial('COM12');            % Define COM Port Object To Communicate To Linux Based
```

```
Computer.
fopen(Port2);                    % Open COM Port.


%% Database Access %%
id = id - 48;
if (id > 0 && id < 5)
   z = xlsread('AETC_Record.xls');        % Read Excel Sheet Having User Database.
   if  (z(id,2) >= 100)          % Minimum balance has to be 100 rupees.
      z(id,2) = z(id,2) - 100;    % Deduct 100 Rupees From Account.
      z(id,3) = 0;              % For Successful Transaction Make Status = 0.
      id = id + 48;
      fprintf(Port2,'%c',id);     % Send Vehicle's ID and No violation Status To Linux Based PC.
      fprintf(Port2,'%c','S');
   else
      z(id,2) = 0;              % If Balance Is Less Than 100 Rupees Then Deduct Remaining Balance.
      z(id,3) = z(id,3)+1;       % Make status = 1 If Violation Is Detected.
      violate = 1;              % Set violation Variable Equals To One.
      id = id + 48;
      fprintf(Port2,'%c',id);     % Send Vehicle's ID and Violation Status To Linux Based PC.
      fprintf(Port2,'%c','F');
   end ;
   id = id - 48;
   z(id,4) = TotalSpeed;
   clk = clock;
   z(id,5) = clk(3);
   z(id,6) = clk(2);
   z(id,7) = clk(1);
   z(id,8) = clk(4);
   z(id,9) = clk(5);
   z(id,10) = clk(6);
   xlswrite('AETC_Record.xls', z);        % Update The Database.
end;
fclose(Port2);                  % Close Port.

%% Image Capture %%

v = ''                        % Initialise Variable v
k = size(v);
while ( k(1) == 0)
fprintf(s,'%c','V');             % Send Character 'V' To Vehicle To Get The Flag Value.
v = fscanf(s)                  % If Flag 'v' is one, Tt Means That Vehicle Has Crossed Third Black Patch.
k = size(v);
end
c = c + 1;
a = [c '.' 'j' 'p' 'g'];
A = ['A' c];
```

```
vid=videoinput('winvideo',2);       % Start USB Camera To Capture Image.
triggerconfig(vid,'manual');


config = triggerinfo(vid);
set(vid,'FramesPerTrigger',1);
set(vid,'TriggerRepeat', Inf);
start(vid);
for i=1:20
   trigger(vid);
   im= getdata(vid,1);
end
stop(vid),delete(vid),clear vid;
cd LicensePlatePhotos ;
imwrite(im,a);
fprintf(s,'%c','D');

%% Image Transformation %%

mm = imread(a);                  % Convert Image From .jpg To .pbm Format.
mm=imcomplement(mm);             % Complement The Image.
cd ..;                           % Change Directory.
imwrite(mm,'C:\Users\pushkarbk\Documents\MATLAB\license.pbm');

%% GOCR %%

[status, result] = dos('gocr048.exe -m 4 C:\Users\pushkarbk\Documents\MATLAB\license.pbm');
% Apply Optical Character Recognition (OCR) Algorithm On Captured Image Of License Plate.
xlswrite('AETC_LicNum.xls', result, 'AETC_LicNum',A);

%% End %%
fclose(s);                       % Close Port.
end                              % End Of While(1) Loop.


%%-------------------------------------
% End of Program.
%%-------------------------------------
```

8.2.4. AETC_GSM.m

```
%%-------------------------------------------------------------------------------
% Code_GSM.m:- Main Program for GSM Communication
% Between TollPlaza and Vehicle Owner
%
%-------------------------------------------------------------------------------
% Author:
%                 Puskar Kothavade
%                 Ashish Pardhi
%                 Mugdha Nazare
%-------------------------------------------------------------------------------

%   Copyright (c) 2010. ERTS Lab IIT Bombay
%   All rights reserved.

%   Redistribution and use in source and binary forms, with or without
%   modification, are permitted provided that the following conditions are met:

%   * Redistributions of source code must retain the above copyright
%     notice, this list of conditions and the following disclaimer.
%
%   * Redistributions in binary form must reproduce the above copyright
%     notice, this list of conditions and the following disclaimer in
%     the documentation and/or other materials provided with the
%     distribution.

%   * Neither the name of the copyright holders nor the names of
%     contributors may be used to endorse or promote products derived
%     from this software without specific prior written permission.

%   * Source code can be used for academic purpose.
%         For commercial use permission form the author needs to be taken.
%-------------------------------------------------------------------------------

% Electronic Toll Tax Collection System %

clc;
warning off all;
s = serial('/dev/ttyUSB0');                         % Open Serial Port.
fopen(s);
```

```
while(1)

   id = fscanf(s)                          % Read Serial Port
   w = size(id);                           % Find Out Whether Variable id Empty Or Non-
empty

   if (w(1) > 0)                           % If Variable id Is Non-empty Then Only Check
Further Code.

      if (id(1) == '1')
         if(id(2) == 'S')
            unix('./id1s');                % Execute Precompiled 'idls' File.
         end
         if(id(2) == 'F')
            unix('./id1f');                % Execute Precompiled 'idlf' File.
         end
      end

      if (id(1) == '2')
         if(id(2) == 'S')
            unix('./id2s');
         end
         if(id(2) == 'F')
            unix('./id2f');
         end
      end

      if (id(1) == '3')
         if(id(2) == 'S')
            unix('./id3s');
         end
         if(id(2) == 'F')
            unix('./id3f');
         end
      end

      if (id(1) == '4')
         if(id(2) == 'S')
            unix('./id4s');
         end
         if(id(2) == 'F')
            unix('./id4f');
         end
```

```
      end


   end

 end

fclose(s);                                  % Close Serial Port.
```

8.2.5 AETC_GSMS.c

```
/**
 * @file AETC_GSMS.c
 * Program for serial communication between Owner and TollPlaza through GSM Module.
 * Send Success Message
 * @author Puskar Kothavade, Ashish Pardhi, Mugdha Nazare, IIT Bombay
 * @date 10/Oct/2010
 * @version 1.0
 *
 * @section LICENSE
  * Copyright (c) 2010. ERTS Lab IIT Bombay
   * All rights reserved.

  * Redistribution and use in source and binary forms, with or without
   *modification, are permitted provided that the following conditions are met:

   * Redistributions of source code must retain the above copyright
    * notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above copyright
   * notice, this list of conditions and the following disclaimer in
   * the documentation and/or other materials provided with the distribution.

   * Neither the name of the copyright holders nor the names of
   * contributors may be used to endorse or promote products derived
   * from this software without specific prior written permission.

   * Source code can be used for academic purpose.
   * For commercial use permission form the author needs to be taken.
 */
#include<stdio.h>
#include<string.h>
#include<malloc.h>
#include<unistd.h>
#include<fcntl.h>
```

```
#include<errno.h>
#include<termios.h>


#include <sys/time.h>
#include <sys/types.h>

/**
* Open port for serial communication.
* display error messsage if unable to open.
*/
int port_open(void)
{
 int fd;
  fd = open("/dev/ttyS0",O_RDWR | O_NOCTTY | O_NDELAY);
  if(fd == -1)
  {
    perror("Unable to open the port: /dev/ttyS0");
  }
  else
  {
    fcntl(fd,F_SETFL,0);
  }
  return(fd); //return file descriptor
}

/**
* Configure port
* @param fd File Descriptor to access serial port
*/

void port_config(int fd)
{
  struct termios settings;

  tcgetattr(fd,&settings);

  cfsetispeed(&settings,9600);
  cfsetospeed(&settings,9600);

  settings.c_cflag |= (CLOCAL | CREAD);

  settings.c_cflag &= ~PARENB;
  settings.c_cflag &= ~CSTOPB;
  settings.c_cflag &= ~CSIZE;
  settings.c_cflag |=  CS8;
```

```
  tcsetattr(fd,TCSANOW,&settings);
}


/**
*Generate Delay of two seconds.
*
*/

void del_2s(void)      // Delay for wait
{
  unsigned char r, s;
  for(r=0;r<125;r++)
    for(s=0;s<255;s++);

   void del_1s(void)
  {
   unsigned char p, q;
   for(p=0;p<125;p++)
     for(q=0;q=125;q++);
  }
}


/**
*Send Message to register owner
* @param fd File Descriptor to access serial port
* @param *c character pointer
*/
void write_data(int fd,char *c)
{
  char s[2] = "";
  int n;
  char *charPointer;

  charPointer = (char *)malloc(sizeof(char) * 10);//Allocate memory

  sprintf(charPointer,"%x",26);

  n = write(fd, c, strlen(c));
  if(n<0)
  {
    fputs("write() of data failed! \n",stderr);
  }
  printf("sent String: %d\n",n);

   n = write(fd, "\nAT\r", 4);
```

```
  sleep(2);



  n = write(fd, "AT+IPR=9600\r",12);
  sleep(2);

  n = write(fd, "AT+CSQ\r", 7);
  sleep(2);

  n = write(fd, "AT+CMGF=1\r", 10);
  sleep(2);

  n = write(fd, "AT+CMGS=\"9619821002\"\r", 21);
  sleep(5);
  n = write(fd, "Dear Customer, 100 Rupees Have Been Deducted From Your Account. Thank
You!\n", 76);
  n = write(fd, "\x1a", 3);
  sleep(2);

  n = write(fd, "\rAT+CMGR=1\n", 11);

}

/**
* Close port for reuse
* @param fd File Descriptor to access serial port
*/
void port_close(int fd)
{
 close(fd);
}

int main(void)
{
 int fd;///< File Descriptor to access serial port
 fd = port_open(); // Open Port
 port_config(fd); //Configure Port
 write_data(fd,""); //Write Message
 del_2s(); //Generate Delay
 port_close(fd); //Close Port
 return 0; //Return
}
```

8.2.6. AETC_GSMF.c

```
/**
 * @file AETC_GSMF.c
 * Program for serial communication between Owner and TollPlaza through GSM Module.
 * Send Fail Message
 * @author Puskar Kothavade, Ashish Pardhi, Mugdha Nazare, IIT Bombay
 * @date 10/Oct/2010
 * @version 1.0
 *
 * @section LICENSE
  * Copyright (c) 2010. ERTS Lab IIT Bombay
   * All rights reserved.

  * Redistribution and use in source and binary forms, with or without
   *modification, are permitted provided that the following conditions are met:

   * Redistributions of source code must retain the above copyright
    * notice, this list of conditions and the following disclaimer.

   * Redistributions in binary form must reproduce the above copyright
   * notice, this list of conditions and the following disclaimer in
   * the documentation and/or other materials provided with the distribution.

   * Neither the name of the copyright holders nor the names of
   * contributors may be used to endorse or promote products derived
   * from this software without specific prior written permission.

   * Source code can be used for academic purpose.
   * For commercial use permission form the author needs to be taken.
 */


#include<stdio.h>
#include<string.h>
#include<malloc.h>
#include<unistd.h>
#include<fcntl.h>
#include<errno.h>
#include<termios.h>
#include <sys/time.h>
#include <sys/types.h>
```

```
/**
* Open port for serial communication.
* display error messsage if unable to open.
*/
int port_open(void)
{
 int fd;
  fd = open("/dev/ttyS0",O_RDWR | O_NOCTTY | O_NDELAY);
  if(fd == -1)
  {
    perror("Unable to open the port: /dev/ttyS0");
  }
  else
  {
    fcntl(fd,F_SETFL,0);
  }
  return(fd); //return file descriptor
}

/**
* Configure port
* @param fd File Descriptor to access serial port
*/

void port_config(int fd)
{
  struct termios settings;

  tcgetattr(fd,&settings);

  cfsetispeed(&settings,9600);
  cfsetospeed(&settings,9600);

  settings.c_cflag |= (CLOCAL | CREAD);

  settings.c_cflag &= ~PARENB;
  settings.c_cflag &= ~CSTOPB;
  settings.c_cflag &= ~CSIZE;
  settings.c_cflag |=  CS8;

  tcsetattr(fd,TCSANOW,&settings);
}
```

```
/**
*Generate Delay of two seconds.
*
*/


void del_2s(void)      // Delay for wait
{
  unsigned char r, s;
  for(r=0;r<125;r++)
    for(s=0;s<255;s++);

   void del_1s(void)
  {
   unsigned char p, q;
   for(p=0;p<125;p++)
     for(q=0;q=125;q++);
  }
}



/**
*Send Message to register owner
* @param fd File Descriptor to access serial port
* @param *c character pointer
*/
void write_data(int fd,char *c)
{
  char ch;
  char s[2] = "";
  int n;
  char *charPointer;

  charPointer = (char *)malloc(sizeof(char) * 10);

  sprintf(charPointer,"%x",26);

  n = write(fd, c, strlen(c));
  if(n<0)
  {
    fputs("write() of data failed! \n",stderr);
  }
```

```
  printf("sent String: %d\n",n);



   n = write(fd, "\nAT\r", 4);
   sleep(2);

   n = write(fd, "AT+IPR=9600\r",12);
   sleep(2);

   n = write(fd, "AT+CSQ\r", 7);
   sleep(2);

   n = write(fd, "AT+CMGF=1\r", 10);
   sleep(2);

   n = write(fd, "AT+CMGS=\"9619821002\"\r", 21);
   sleep(5);
   n = write(fd, "Dear Customer, Your Account Does Not Have Sufficient Balance To Use Our
Service. Thank You!\n", 93);
   n = write(fd, "\x1a", 3);
   sleep(2);

   n = write(fd, "\rAT+CMGR=1\n", 11);

}

/**
* Close port for reuse
* @param fd File Descriptor to access serial port
*/
void port_close(int fd)
{
  close(fd);
}

int main(void)
{
  int fd;
  fd = port_open(); // Open Port
  port_config(fd); //Configure Port
  write_data(fd,""); //Write Message
  del_2s(); //Generate Delay
  port_close(fd); //Close Port
  return 0; //Return
}
```

9. Presentation

Automated Electronic Toll Tax Collection System (AETC)

Guided By : Prof. Kavi Arya
Course Project CS684
Group14
Pushkar Kothavade (09307924)
Mugdha Nazare (09307915)
Ashish Pardhi (09305049)

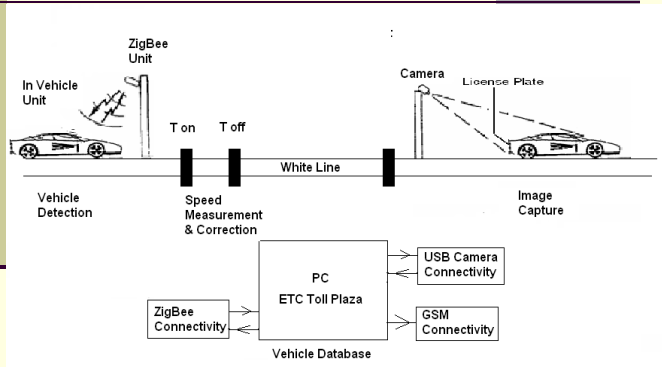
Outline

- Introduction
- System Architecture
- System Flowchart and State chart
- Assumptions
- Limitations
- Challenges
- Reusability
- Future Scope
- Implementation Details
- References

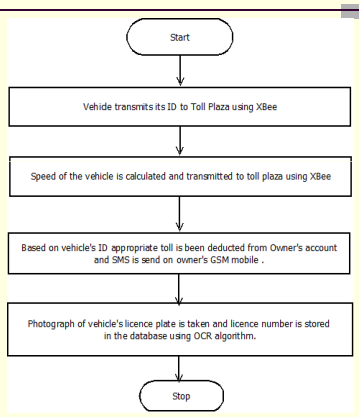
Introduction

- Automated Electronic Toll Tax Collection (AETC) system is proposed to avoid traffic congestion near toll plazas due to manual toll tax collection system.
- Automatic money deduction from prepaid account.
- Facilitates uninterrupted flow of traffic.

System Architecture



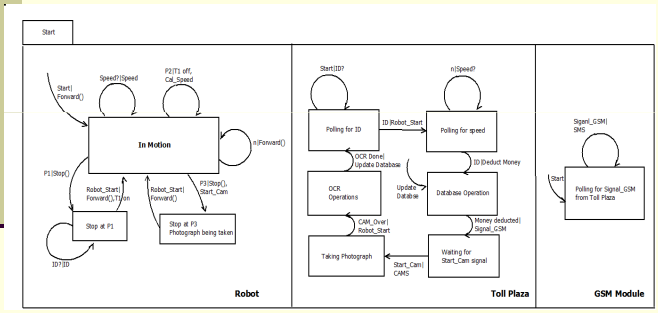
System Flowchart



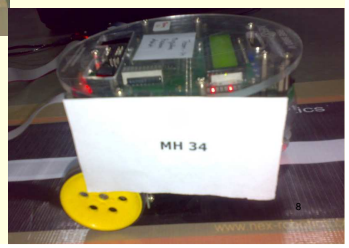
System Architecture

- Speed calculation mechanism.
- Money deduction from prepaid account.
- Acknowledgment of payment on registered GSM mobile.
- Insufficient cash in the account is treated as violation
- Character recognition algorithm to read the characters present in the photograph.
- Recognized characters are stored in database and can be used for law enforcement.

State chart of the system



Robot Photos



Assumptions

- Each vehicle has ZigBee enabled transmitter receiver module and posses a unique ID through which toll plaza identifies the vehicle.
- The vehicle owner uses GSM subscribed mobile.
- A specific license plate and alphanumeric format is assumed.

Limitations

- Delay introduced at the first black patch in order to synchronize with Matlab based operations.
- Delay introduced at third black patch in order to synchronize with camera operation.
- Accuracy of image processing algorithm depends on light conditions

Challenges

- Integration and synchronization of all the modules such as GSM module, Toll Plaza on PC and Robot
- Accessing GSM module using Matlab
- Positioning camera with appropriate light condition and settings such as brightness, contrast, etc.

Reusability

- Interested students can use this basic prototype and build up a more improvised system.
- All the project codes are made available along with a readme file.
- Codes are well commented and are divided into modules which can be reused independently.
- Equipments used in the project other than robot (ERTS lab, IITB and NexRobotics property) are easily available.

Future Scope

- System can be implemented on Indian road ways with appropriate modifications and with more sophisticated equipments.
- More advanced computational softwares may be used in order to reduce the computational time.
- Mobile based application can be developed which accept AT commands and replaces GSM module with mobile.
- More advanced camera capable of taking clear picture of moving object can be used.

Work Division and Milestones

Task List and Work Division

Task	Person	Update	Challenges	Status	Completion Date
Image Processing of License Plate	All	Test-1: Testing Of OCR Algorithm In Matlab	Works Fine On Computer Generated Images	Done	1/10/2010
Image Processing of License Plate	All	Test-2: Testing Of OCR Algorithm In Actual Scenario	Problems With Detection	Done	28/10/2010
Documentation	All	First Draft Is Submitted	-	Done	22/09/2010
Presentation	All	Done With First Presentation	-	Done	27/09/2010
Setup Preparation and Test Plan Execution	All	Setup Is Ready	-	Done	29/10/2010
Implementation of Toll Plaza on PC Using MATLAB	Pushkar Kothavade	We Will Upload Version-1.0 Of Code On SVN soon	-	Done	19/10/2010
Implementation of Database in MS-Excel	Pushkar Kothavade	Matlab Can Automatically Read and Write to MS-Excel Sheet	-	Done	25/09/2010
Access to ZigBee Module (Serial Communication)	Pushkar Kothavade	Matlab Can Access ZigBee Module	-	Done	30/09/2010
Access to Camera (USB Communication)	Pushkar Kothavade	Image Capture Mechanism Using USB Camera	-	Done	03/10/2010
Programming of Vehicle	Mugdha Nazare	Almost Done	-	Done	29/10/2010
Speed Calculation using Timer	Mugdha Nazare	Black Patch Detection & Speed Measurement	-	Done	02/10/2010
White Line Follower Code	Mugdha Nazare	Implemented White Line Follower Code With Black Patch Detection	-	Done	04/10/2010
ZigBee communication (Vehicle Side))	Mugdha Nazare	Tested ZigBee Communication Between Vehicle and Toll plaza	-	Done	07/10/2010
Display on LCD code	Mugdha Nazare	It Is Working Fine	-	Done	12/10/2010
Implementation Database management System	Ashish Pardhi	Integration of Database With System	-	Done	23/10/2010
Implementation of GSM Communication	Ashish Pardhi	Basic testing on GSM Module Using HyperTerminal	-	Done	8/10/2010
Implementation of GSM Communication	Pushkar Kothavade	C code written to access GSM Module	-	Done	11/10/2010
Implementation of GSM Communication	Ashish Pardhi	Integration of C code into MATLAB using MEX File	Not Working	Failed	17/10/2010
Implementation of GSM Communication	Pushkar Kothavade	Tried Shell Scripting Using Command Line Terminal Software	Not Working	Failed	19/10/2010
Implementation of GSM Communication	Pushkar, Mugdha, Ashish	Planning To Implement GSM C code on Separate Machine	-	Done	2/11/2010

Implementation Details

- Communication Interfaces
  - ZigBee (802.15.4) module.
  - GSM module.
- Hardware Interfaces
  - USB camera.
- Software Interfaces
  - Database (Implemented on Microsoft Excel)
  - MATLAB-7.4 (For Toll plaza implementation).

References

- The Making of Singapore's Electronic Road Pricing System by A P G Menonand, Dr Chin Kian Keong Land Transport Authority Singapore Proceeding of the International Conference on Transaction Into The Next Millenium Singapore, 9-11 Sept 1998, Centre for Transaction Studies , Nanyang Technological University.
- Firebird V Hardware & Software manuals.
- MATLAB Help
- www.zigbee.org
- NexRobotics Manuals for image processing

Thank You !!