Indian Institute of Technology, Gandhinagar

EE 404 Embedded Systems - Spring 2014

Vehicle crash detection and mitigation - Project Report

- Ishan Upadhyay (11110038), Parth Gudhka (11110062), Shashank Heda (11110094)

Abstract

The project is aimed at making a micro-controller based vehicle crash detection device that tries to reduce the severity of the consequences of a crash by sending the coordinates of the crash to a predefined set of numbers set by the user. This could be used to send the location to a close relative and/or a hospital for example to minimize the precious time taken to respond to a crash, and thus probably save lives.

Problem/Project Statement

The aim of the project is to reduce the time taken in getting information of a vehicle crash to a well-wisher of a crash victim thereby reducing the response time and saving precious moments in getting the crash victims to the required health care.

Project Relevance/Applications

This design is a system which can detect accidents in significantly less time and sends the basic information to first aid centre within a few seconds covering geographical coordinates and the time at which a vehicle accident had occurred. This alert message is sent to the rescue team in a short time, which will help in saving the valuable lives. A switch is also provided in order to terminate the sending of a message in rare case where there is no casualty, this can save the precious time of the medical rescue team. When the accident occurs the alert message is sent automatically to the rescue team and to the police station. The message is sent through the GSM module and the location of the accident is detected with the help of the GPS module.

This application provides the optimum solution to poor emergency facilities provided to the roads accidents in the most feasible way.

Project Analysis

Detecting an accident is usually done using an accelerometer. When a crash happens, the deceleration of a car is very high, and much higher than usual vehicle accelerations. Hence, when the acceleration crosses a certain limit, the accident mitigation system, which consists of the GPS and GSM, is triggered. Till then, the processor keeps checking whether the threshold value of the accelerometer has been crossed.

Once the mitigation system is triggered, the Atmel processor serially sends commands to the GPS module to receive the coordinate's information. The GPS sends a lot of data including date, time, coordinates etc. Hence, the processor is left with the task of parsing the information

and extracting just the latitude and longitude. Once this is done, the processor sends commands to the GSM module to set it up for sending text messages and then sends a customised message with the coordinates and instructions to send the SMS to a predefined set of phone numbers.
An overall block diagram and interfacing schematic can be seen in Figure 1 and 2 respectively.

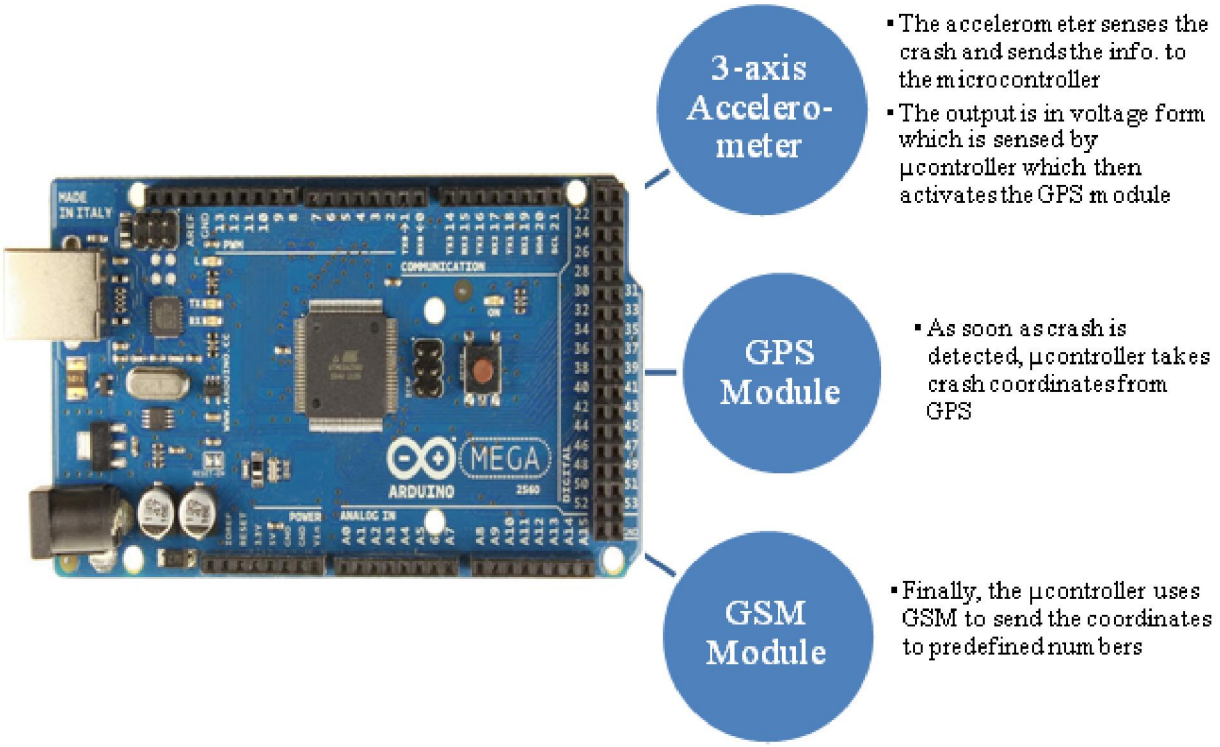

Figure 1: Block Diagram for the Crash detection and mitigation strategy

Schematic

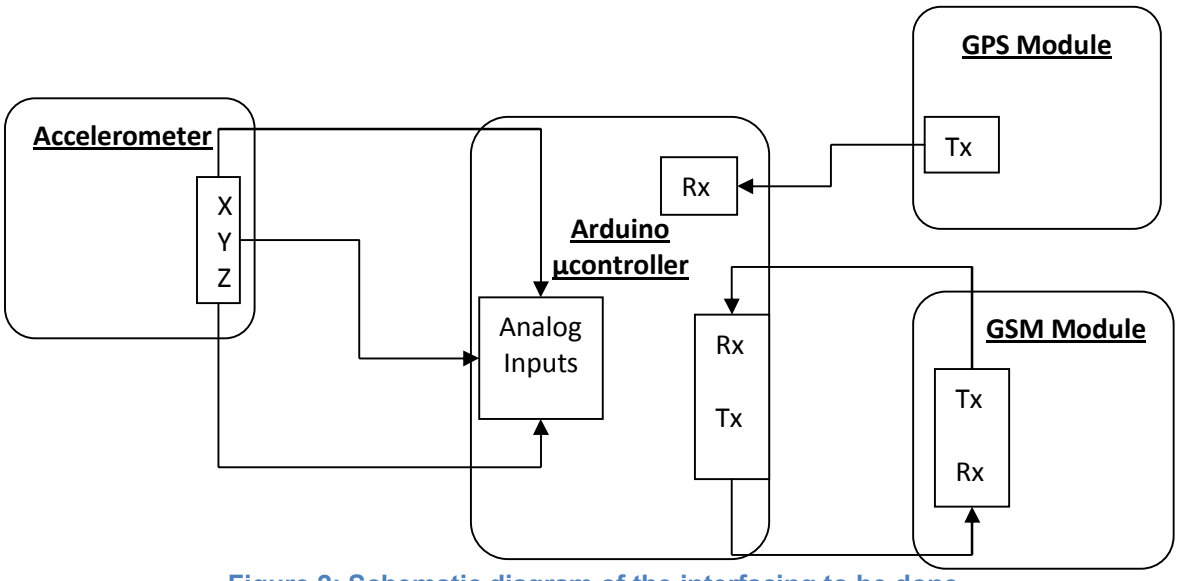

Figure 2: Schematic diagram of the interfacing to be done

Components and their Description

We would require the following components for completing the project:

1. **Arduino Mega 2560**

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator and a USB connector through which it is usually programmed. The advantage of using the Mega 2560 is because it has a large number of I/O pins which is quite useful here, as we have at least three devices to interface. In case an LCD is included, no other board would have enough pins to handle all the devices. The Arduino is burned using standard Arduino software available at http://arduino.cc/en/Main/Software

2. **Low Power, 3-Axis ±3 g Accelerometer - ADXL335**

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The accelerometer returns analog input voltages for x, y and z directions to the Arduino board proportional to the acceleration in the corresponding directions, and the Arduino then converts it into 10-bit data. So, the readings from the accelerometer would range from 0-1023. As soon as we get a value above a certain threshold (for demonstration purposes, this threshold is kept 400), we trigger the crash mitigation part of the cycle, i.e. using GPS and GSM to send SMS to a concerned person. When an acceleration above the range of the accelerometer is experienced, we receive a zero voltage. Hence, we must also include that as a triggering signal.

3. **GTPA010 GPS Receiver with Embedded Antenna**

The Global Positioning System (GPS) is a satellite based navigation system that sends and receives radio signals. A GPS receiver acquires these signals and provides the user with information. Using GPS technology, one can determine location, velocity and time, 24 hours a day, in any weather conditions anywhere in the world for free. GPS was formally known as the NAVSTAR (Navigation Satellite Timing and Ranging). Global Positioning System was originally developed for military. Because of its popular navigation capabilities and because GPS technology can be accessed using small, inexpensive equipment, the government made the system available for civilian use. The USA owns GPS technology and the Department of Defense maintains it. The basis of the GPS technology is a set of 24 satellites that are continuously orbiting the earth. These satellites are equipped with atomic clocks and send out radio signals as to the exact time and their location. These radio signals from the satellites are picked up by the GPS receiver. Once the GPS receiver locks on to four or more of these satellites, it can triangulate its location from the known positions of the satellites. It is a high performance, low power satellite based model. It is a cost effective and portable system which accurately detects the location.

4. **SIM300 GSM Module**

A GSM module is a modem which accepts any SIM card and acts just like a mobile phone with its own unique phone number. The key difference between a normal cell phone and a GSM module is the fact that a GSM module can be interfaced with other devices such as a PC via RS232 or a microcontroller/microprocessor using UART communications. This is particularly

useful for applications such as data logging, SMS control, remote control etc. In this project we used the SIM300 variant of GSM module and like other variants the module could be controlled by issuing it predefined commands known as “AT” commands. These commands have a fixed syntax of the form *“ATS<n>=<m>”*, where “n” denotes the index of the *“S”* register and “m” is the value assigned to it, also “m” is optional and in case it is missing, a default value is assigned to it.
The SIM300 is designed with power saving technique, the current consumption being as low as 2.5mA in SLEEP mode. The SIM300 is integrated with the TCP/IP protocol, Extended TCP/IP AT commands are developed for customers to use the TCP/IP protocol easily, which is very useful for data transfer applications.

Software: Overall flowchart

A flow chart for the overall process is shown in Figure 3.

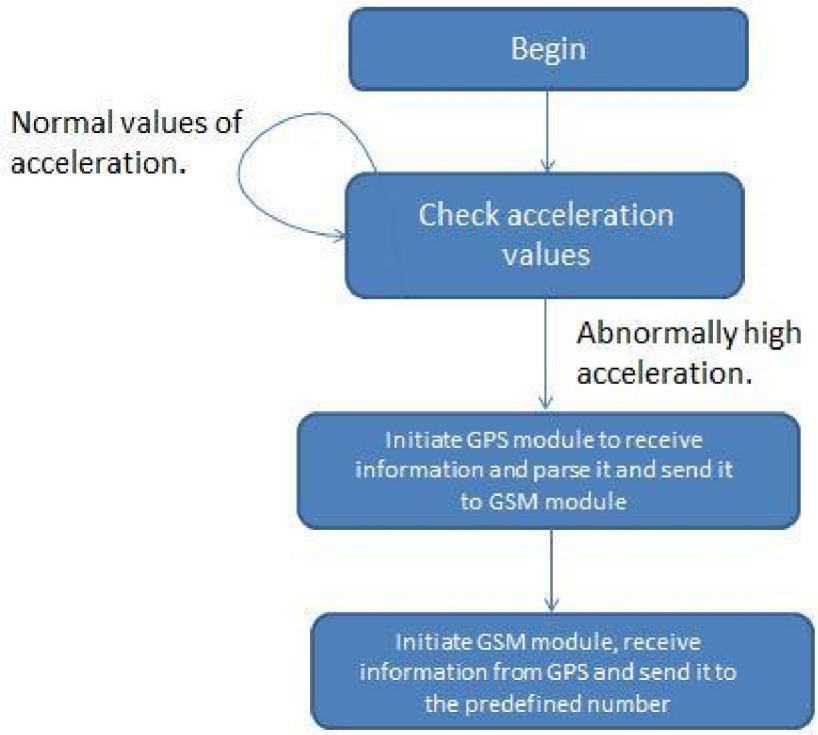

Figure 3: Flow chart guideline for the process to be followed

Results

We observe that on moving the accelerometer rigorously, to simulate an accident, a message is received on the target device in the following format:
Emergency! Accident @ Latitude: 23.065206,N; Longitude: 72.355496,E

A picture of the setup is shown in Figure 4.

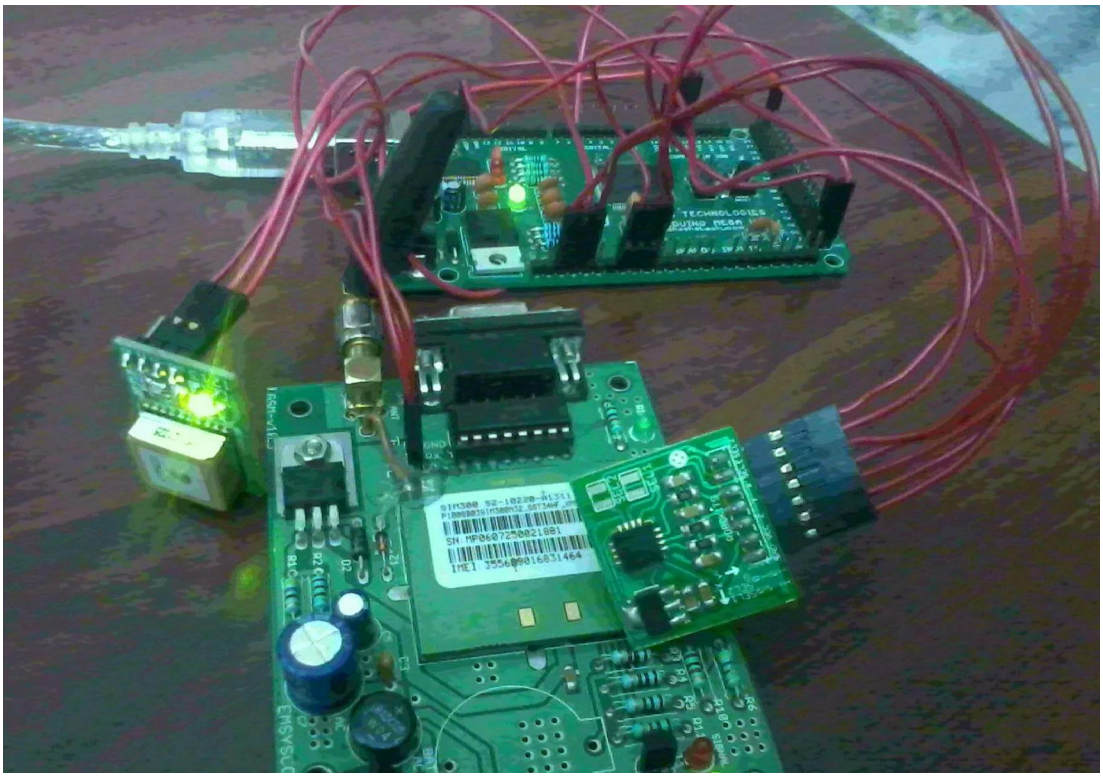

Figure 4: Project Set-up

Future Possibilities:

1. Fabricate project on PCB

To use this circuit systematically, a PCB would be needed, otherwise the setup is a messy tangle of wires. So, creating a PCB would make the circuit much easier to implement. We could add a 16X2 LCD to signal the status to the user about which operation is being carried out. In addition, the current GSM module is powered using a 5V power adapter, which doesn't make it practically usable. So, we would need to include a battery source to power the GSM and the Arduino board both so that they could be made portable and hence be possible used in vehicles.

2. GPS using active antenna

The drawback in the current implementation is that the location that the GPS provides is almost 40 kilometres away from the actual location, and hence not practically usable. We suspect the problem is chiefly due to the embedded antenna not being strong enough to capture signals exactly. To get enough signals, the current module needs a view of the sky, whereas in other external active antenna type GPS modules, the signal received is much stronger, and it can work in much harsher conditions. We could try using the active antenna type GPS to get more accurate location coordinates.

Acknowledgement

We express our deep gratitude to Dr. Ramesh Gaonkar, Visiting Professor, Department of Electrical Engineering, Indian Institute of Technology, Gandhinagar, India, Mr. Abhishek Upadhyay, Teaching Assistant (Embedded Systems), PhD, IIT Gandhinagar and Ms. Sneha Ved, Teaching Assistant (Embedded Systems), PhD, IIT Gandhinagar for their excellent guidance and scrutiny provided throughout the project. This project would have been impossible without their guidance.

Appendix

```
//Code for vehicle crash detection and mitigation
#include <string.h>
#include <ctype.h>
int   ledPin = 13;                  // LED test pin
int   rxPin = 0;                    // RX PIN
int   txPin = 1;                    // TX TX
char  linea[300] = "";
char  comandoGPR[7] = "$GPRMC";
int   cont=0;
int   bien=0;
int   conta=0;
int   indices[13];
char  c[50];
char  d[50];
int   byteGPS=-1;
int   flag =0;
char* cell_number = "+919876543210"; // Phone number to send SMS

void setup() {
  // initialize all serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT); // Initialize LED pin
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);
  for (int i=0;i<300;i++){ // Initialize a buffer for received data
    linea[i]=' ';
  }
}

// the loop routine runs over and over
void loop() {
  // Read the input on analog pin 0,1 and 2
  int x = analogRead(A0);
  int y = analogRead(A1);
  int z = analogRead(A2);
  // Print out the values read from accelerometer
  Serial.print("X= ");
  Serial.print(x);
  Serial.print("; Y= ");
  Serial.print(y);
  Serial.print("; Z= ");
  Serial.println(z);
  // If accident is detected, the accelerometer's x and y values go
  // above 400
  // or if the range of values is exceeded, it displays zero.
```

```
  // Therefore, the three condition in the for loop.
  if(( x>=400 )||( y>=400 )||(x*y == 0)){
    Serial.println("Accident Detected!");
    GPScall();  // Call function that takes coordinates from GPS
  }
  delay(2000);  // delay in between reads for stability
}


//function for receiving input from GPS module, extracting relevant
//information and creating the message

void GPScall()
{
  char lat[200]="Emergency! Accident @ Latitude: ";
  while(flag!=1){
    char dummy[200]="Emergency! Accident @ Latitude: ";
    strcpy(lat,dummy);     // initialize the "lat" string at every new
iterations
    char lon[200] ="";  Longitude: ";
    digitalWrite(ledPin, LOW);
    byteGPS=Serial3.read();// Read a byte of the serial port
    if (byteGPS == -1)           // See if the port is empty yet
    {
       delay(100);
      digitalWrite(ledPin, LOW);
    }

    else
    {
      digitalWrite(ledPin, LOW);
      // If there is serial port data, it is put in the buffer
      linea[conta]=byteGPS;
      conta++;

      if (byteGPS==13)
      {
      /*
      If the received byte is = to 13, end of transmission note: the
      actual end of transmission is <CR><LF> (i.e. 0x13 0x10)
      */
        digitalWrite(ledPin, LOW);
        cont=0;
        bien=0;
        // The following for loop starts at 1 as
        // the first byte is <LF> (0x10) from the previous
        //transmission.
        for (int i=1;i<7;i++)
        {
          // Verifies if the received command starts with $GPR
          if (linea[i]==comandoGPR[i-1])
          {
```

```
            bien++;
          }
        }
        if(bien==6){
          // If yes, continue and process the data
          for (int i=0;i<300;i++){
            if (linea[i]==','){
              // check for the position of the  "," separator
              // note: again, there is a potential buffer overflow
              // here!
              indices[cont]=i;
              cont++;
            }

            if (linea[i]=='*'){
              // ... and the "*"
              indices[12]=i;
              cont++;
            }
          }

          int i=0;
          for(int j = indices[2]; j< indices[4]-1; j++){
            c[i++] = linea[j+1];
          }
          c[i]='\0';
          int k=0;
          for(int j = indices[4]; j< indices[6]-1; j++){
            d[k++] = linea[j+1];
          }
          d[k]='\0';

          if((c[10]=='N')&&(d[11]=='E')){
            strcat(lat,c);
            strcat(lon,d);
            strcat(lat,lon);
            flag =1;
           // having generated the final message to be send, the
           // sendSMS function is invoked
            sendSMS(lat,cell_number);
            break;
          }
        }
        conta=0;                  // Reset the buffer
        for (int i=0;i<300;i++){
          linea[i]=' ';
        }
      }
    }
  }
}
```

```
//Function to issue commands to the gsm module

void sendCommand(char* atCommand){
  Serial.println("Checking");
  uint8_t x=0;
  char response[100];
  char c;
  unsigned long previous;

  memset(response, '\0', 100);   // Initialize the string

  delay(100);

  // Clean the input buffer

  while( Serial2.available() > 0) Serial2.read();
  Serial2.print(atCommand);      // Send the AT command
  x = 0;
  previous = millis();

  // Following loop waits for 2 seconds for the answer
  do{
    if(Serial2.available() != 0){
      // if there is data in the UART input buffer, read it and check
     // for
      c = Serial2.read();
      response[x++] = c;
    }
    // Waits for the answer with time out
  }
  while(((millis() - previous) < 2000));
  Serial.println(response);
}

//Function for sending an sms

void sendSMS(char* message, char* number){
  char c[20] sms_command;
  Serial.println(lat);
  Serial.println("Starting...");
  sendCommand("AT+CMGF=1\r\n");
  sprintf(sms_string,"AT+CMGS=\"%s\"\r\n", number);
  sendCommand(sms_string);
  sendCommand(message);
  sendCommand("\n");
  Serial2.print((char)0x1a);
  sendCommand("\n");
}
```