**AUTOMATIC VEHICLE ACCIDENT ALERT SYSTEM**

**TEAM NAME** : DREAM CRUSHERS

**TEAM MEMBERS:**

1.G.SHIVAPRASAD SAGAR

2.ARCHANA

3.ARUNA

**ABSTRACT**

As the usage of vehicles is increasing drastically,

the hazards due to vehicles is also increased. The main cause for accidents is high speed, drunk and drive, diverting minds, over stress and due to electronic gadgets. This paper deals with accident detection system that occurs due to carelessness of the person who is driving the vehicle. This introduces accident alerting system which alerts the person who is driving the vehicle.

If the person is not in a position to control the vehicle then the accident occurs. Once the accident occurs to the vehicle this System will send information to registered mobile number.

**HARDWARE COMPONENTS**

1. Arduino UNO
2. Buzzer
3. Accelerometer
4. GPS
5. GSM

**Software required:**

1. Arduino IDE

**Block Diagram**

**CLOUD**

**GPS+GSM**

**BUZZER**

**Arduino UNO**

**Accelerometer**

**Mobile**

**Arduino UNO**



**Arduino** is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control both physically and digitally. Its products are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as do-it-yourself (DIY) kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

**Accelerometer**

Image result for Accelerometer


An **accelerometer** is a device that measures proper acceleration. Proper acceleration, being the acceleration (or rate of change of velocity) of a body in its own instantaneous rest frame, is not the same as coordinate acceleration, being the acceleration in a fixed coordinate system. For example, an accelerometer at rest on the surface of the Earth will measure an acceleration due to Earth's gravity, straight upwards (by definition) of g ≈ 9.81 m/s2. By contrast, accelerometers in free fall (falling toward the center of the Earth at a rate of about 9.81 m/s2) will measure zero.

Accelerometers have multiple applications in industry and science. Highly sensitive accelerometers are components of inertial navigation systems for aircraft and missiles. Accelerometers are used to detect and monitor vibration in rotating machinery. Accelerometers are used in tablet computers and digital cameras so that images on screens are always displayed upright. Accelerometers are used in drones for flight stabilisation. Coordinated accelerometers can be used to measure differences in proper acceleration, particularly gravity, over their separation in space; i.e., gradient of the gravitational field. This gravity gradiometry is useful because absolute gravity is a weak effect and depends on local density of the Earth which is quite variable.

**Global Positioning System (GPS)**



The **Global Positioning System** (**GPS**), originally **Navstar GPS**, is a satellite-based radionavigation system owned by the United States government and operated by the United States Air Force. It is a global navigation satellite system that provides geolocation and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. Obstacles such as mountains and buildings block the relatively weak GPS signals. The GPS does not require the user to transmit any data, and it operates independently of any telephonic or internet reception, though these technologies can enhance the usefulness of the GPS positioning information. The GPS provides critical positioning capabilities to military, civil, and commercial users around the world.

While originally a military project, GPS is considered a dual-use technology, meaning it has significant civilian applications as well.

GPS has become a widely deployed and useful tool for commerce, scientific uses, tracking, and surveillance. GPS's accurate time facilitates everyday activities such as banking, mobile phone operations, and even the control of power grids by allowing well synchronized hand-off switching.

**GSM**

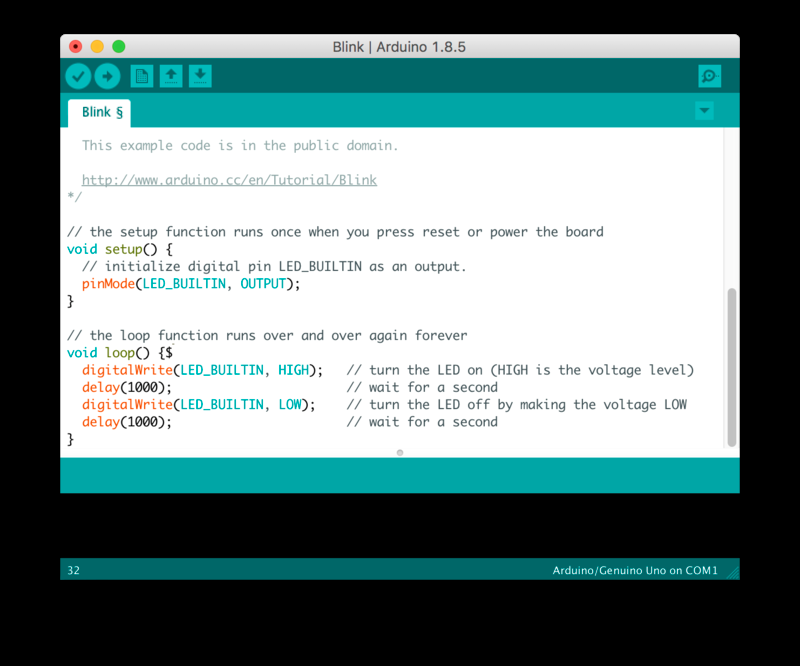


The **Global System for Mobile Communications** (**GSM**) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile devices such as mobile phones and tablets. It was first deployed in Finland in December 1991.By the mid-2010s, it became a global standard for mobile communications achieving over 90% market share, and operating in over 193 countries and territories.2G networks developed as a replacement for first generation (1G) analog cellular networks. The GSM standard originally described a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via General Packet Radio Service (GPRS), and Enhanced Data Rates for GSM Evolution (EDGE).

**BUZZER**

**Buzzer** or **beeper** is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (*piezo* for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

**Arduino IDE**



The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2

A *sketch* is a program written with the Arduino IDE. Sketches are saved on the development computer as text files with the file extension **.ino**. Arduino Software (IDE) pre-1.0 saved sketches with the extension **.pde**

**Hardware Connections**

**A close up of a device

Description automatically generated**

**Program**

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include<SoftwareSerial.h>

#include<Servo.h>

#define sw D1

String msg91\_data;

String data,url,line;

int l;

SoftwareSerial mySerial(D2,D3); // incoming sensor data

SoftwareSerial mySerial1(D5 ,D6); // retriew mesg from these pins

String talkBackCommand,tbRequest,hostm,msgurl,url1;

Servo servo;

int randomNumber;

int State ;

const char\* ssid = "MLRITM-PLACEMENT";//Enter the ssid of your router

const char\* password = "Mlritm@123";//Enter the password of your router

void setup() {

Serial.begin(115200);

pinMode(D1,INPUT);

mySerial.begin(115200);

mySerial1.begin(115200);

servo.attach(D4);

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

// l=1;

// msg91();

}

void loop()

{

Sensor\_Data();

delay(1000);

thingspeak();

delay(1000);

getTalkBack(); // Get the command issued from mobile app/web app

if(talkBackCommand == "9999") {

servo.write(180);

Serial.println("srvoo");

delay(30000);

servo.write(0);

}

else{

}

delay(10000);

msg91();

}

/\*......................................Sensor Data.....................................\*/

void Sensor\_Data()

{

State = digitalRead(D1);

Serial.println(State);

if ( State == 0){

url1="/api/sendhttp.php?mobiles=8008795449&authkey=282211AgHyIyq1REzu5d0dff9d&route=4&sender=TESTIN&message=HELP&country=91";

msgurl = url1;

msggg();

}

delay(1000);

}

/\*.....................................Uploading to Cloud...............................\*/

void getTalkBack()

{

const char\* host = "api.thingspeak.com";

const String talkBackAPIKey = "USQX4JVP6FZ5XKVS";

const String talkBackID = "33125";

WiFiClient client; // Initialize a Wi-Fi client

tbRequest="GET /talkbacks/"+ talkBackID + "/commands/execute?api\_key=" + talkBackAPIKey;

Serial.println(tbRequest);

if (client.connect("api.thingspeak.com", 80))

{

client.println(tbRequest);

if (client.connected())

{

talkBackCommand="";

while(client.connected() && !client.available()) delay(10); //waits for data

while (client.connected() || client.available())

{

talkBackCommand = client.readStringUntil('\n');

}

Serial.print("Command -> ");

Serial.println(talkBackCommand);

Serial.println();

}

client.stop();

Serial.flush();

}

}

void thingspeak(){

const char\* thingspeak\_host = "api.thingspeak.com";

const String privateKey = "Y93DUNPL1V9QBDKV";

if(mySerial.available()>0){

Serial.println("data available");

if(mySerial.find('$')){

Serial.println("$ available");

data = mySerial.readStringUntil('\*');

Serial.println(data);

Serial.print("connecting to ");

Serial.println(thingspeak\_host);

// Use WiFiClient class to create TCP connections

WiFiClient client;

const int httpPort = 80;

if (!client.connect(thingspeak\_host, httpPort)) {

Serial.println("connection failed");

return;

}

// We now create a URI for the request

String thingspeak\_url = "/update";

thingspeak\_url += "?api\_key=";

thingspeak\_url += privateKey;

thingspeak\_url+= data;

Serial.print("Requesting thingspeak\_url: ");

Serial.println(thingspeak\_url);

// This will send the request to the server

client.print(String("GET ") + thingspeak\_url + " HTTP/1.1\r\n" +

"Host: " +thingspeak\_host + "\r\n" +

"Connection: close\r\n\r\n");

delay(1000);

// Read all the lines of the reply from server and print them to Serial

while(client.available()){

String line = client.readStringUntil('\r');

Serial.print(line);

}

}

}

Serial.println();

Serial.println("closing connection");

}

void msg91(){

url1="/api/sendhttp.php?mobiles=8008795449&authkey=282211AgHyIyq1REzu5d0dff9d&route=4&sender=TESTIN&message=HELLO BOSS&country=91";

msgurl = url1;

// thingspeak\_host="api.msg91.com";

if( mySerial1.available()>0){

// msg91\_data = mySerial1.readStringUntil('\*');

// Serial.println(msg91\_data);

if (mySerial1.find('#')){

msg91\_data = mySerial1.readStringUntil('\*');

Serial.println("data is available for message");

Serial.println("found #");

l=1;

Serial.println(msg91\_data);

}}

if(l==1){

msggg();

}

l=0;

}

void msggg(){

hostm="api.msg91.com";

Serial.print("connecting to ");

Serial.println(hostm);

// Use WiFiClient class to create TCP connections

WiFiClient client;

const int httpPort = 80;

if (!client.connect("api.msg91.com", httpPort)) {

Serial.println("connection failed");

return;

}

// We now create a URI for the request

Serial.print("msgurl: ");

Serial.println(msgurl);

// This will send the request to the server

client.print(String("GET ") + msgurl + " HTTP/1.1\r\n" +

"Host: " +hostm + "\r\n" +

"Connection: close\r\n\r\n");

delay(1000);

// Read all the lines of the reply from server and print them to Serial

while(client.available()){

String line = client.readStringUntil('\r');

Serial.print(line)

}

}

Program2:

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

TinyGPSPlus gps; // Create an Instance of the TinyGPS++ object called gps

SoftwareSerial ss(9, 10); // rx,tx //gps tx, rx

SoftwareSerial mySerial(7, 8); // 7 to D3-TX and 8 to D2-RX to send sensor and lat and lon data to node mucu

SoftwareSerial mySerial1(4,5); // 4 to D6 on nodemcu , 5 to D5 on nodemcu to send message sting to nodemcu

int x, y, z, bstate2;

int buzzer = 11;

int sw2 = 3;

String value, url1;

double latitude, longitude;

String condition ;

int acc = 0, i = 0, j = 1;

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

mySerial.begin(115200 );

mySerial1.begin(115200);

ss.begin(9600);

pinMode(sw2, INPUT);

pinMode(buzzer, OUTPUT);

digitalWrite(buzzer, HIGH);

}

void loop() {

// put your main code here, to run repeatedly:

x = analogRead(A0);

y = analogRead(A1);

z = analogRead(A2);

Serial.println( "x: "+String(x));

Serial.println( "y: "+String(y));

Serial.println( "z: "+String(z));

bstate2 = digitalRead(sw2);

Serial.println( "BUTTON STATE "+String(bstate2));

latitude = gps.location.lat();

longitude = gps.location.lng();

delay(1000);

smartDelay(1000);

String url1 = "$";

url1 += "&field3=";

url1 += condition;

url1 += "&field1=";

url1 += String(latitude, 6);

url1 += "&field2=";

url1 += String(longitude, 6);

url1 += "\*";

//Serial.println(url1);

if (millis() > 5000 && gps.charsProcessed() < 10)

Serial.println(F("No GPS data received: check wiring"));

if (x >=265 && y > 265 && z>265) {

Serial.println("accident is detected");

acc = 1;

}

// else if (x >380 && y > 390) {

// Serial.println("accident is detected");

// acc = 1;

// }

else {

Serial.println(" no accident is detected");

acc = 0;

}

if (acc == 1)

{

Serial.println("accident happened");

digitalWrite(buzzer, HIGH);

SendMessage();

// tone(buzzer ,1000);

delay(3000);

digitalWrite(buzzer, LOW);

//SendMessage();

// noTone(buzzer);

while (j == 1) {

bstate2 = digitalRead(sw2);

Serial.println( "BUTTON STATE "+String(bstate2));

if ( bstate2 == 0) {

j = 0;

}

else if (i == 5) {

j = 0;

}

i++;

delay(500);

}

if (bstate2 == 0)

{

Serial.println("vehicle is safe");

condition = "0";

mySerial.println(url1);

Serial.println(url1);

}

else if (bstate2 == 1)

{

Serial.println("vehicle is not safe");

condition = "1";

value = "#accident occured at lat ="+String(latitude,6)+"and lon ="+String(longitude,6);

value += "\*";

Serial.println(value);

mySerial1.println(value);

delay(1000);

mySerial.println(url1);

Serial.println(url1);

}

}

j = 1;

i=0;

delay(1000);

}

static void smartDelay(unsigned long ms) // This custom version of delay() ensures that the gps object is being "fed".

{

unsigned long start = millis();

do

{

while (ss.available())

gps.encode(ss.read());

} while (millis() - start < ms);

}

void SendMessage()

{

{

// mySerial.println("AT+CMGS=\"+9848088305\"\r"); // Replace x with mobile number

//delay(1000);

Serial.println("HELP");// The SMS text you want to send

delay(100);

// mySerial.println((char)26);// ASCII code of CTRL+Z

//delay(1000);

value = "#accident occured at lat ="+String(latitude,6)+"and lon ="+String(longitude,6);

value += "\*";

Serial.println(value);

mySerial1.println(value);

delay(1000);

}

}