```
#include <DS3231.h>
#include <DFRobot_sim808.h>
#include <SoftwareSerial.h>
#include <SD.h>
#include "ELMduino.h"
#include <MPU6050.h>
```

- The above lines include necessary libraries for various functionalities used in the code.

```
DS3231 rtc(2, 3);
SoftwareSerial mySerial(10, 11); //sim808 serial
DFRobot_SIM808 sim808(&mySerial);  //Connect RX,TX,PWR,
SoftwareSerial obdSerial(0,1); // RX, TX obd
const int chipSelect = 53;
const int sensorPin = A0;  // analog input pin for the sensor
```

- The above lines define objects and variables used in the code, including DS3231 (real-time clock), SoftwareSerial objects for communication, and pin assignments.

```
#define PHONE_NUMBER1 "94769439154"
#define PHONE_NUMBER2 "94713651522"
#define MESSAGE_LENGTH 5
#define ELM_PORT obdSerial
const int threshold = 800;  // vibration threshold
int sensorValue;
char latStr[15];
char lonStr[15];
char message[5];
int messageIndex = 0;
char phone[16];
char datetime[24];
String date;
String time;
char MESSAGE[300];
float lat, lon, speed;
ELM327 myELM327;
MPU6050 mpu;
int rpm = 0;
float fuel = 0;
int ax, ay, az;
int gx, gy, gz;
```

- The above lines define various constants, variables, and arrays used in the code.

```
void setup() {
  // Initialization code for setup
```

```
}
```

- The `setup()` function is called once when the Arduino board is powered on or reset. It is used for initializing variables, configuring pins, and setting up necessary components.

```
void loop() {
  // Main code to be executed repeatedly in a loop

}
```

- The `loop()` function is called repeatedly after the `setup()` function. It contains the main code that runs continuously.

```
void getECUData(){
  // Function to retrieve data from the ELM327 module

}
```

- The `getECUData()` function retrieves data from the ELM327 module, specifically the RPM (revolutions per minute) and fuel level.

```
void sosTriger(){
  // Function to trigger an SOS message

}
```

- The `sosTriger()` function sends an SOS message to specified phone numbers when a vibration threshold is exceeded.

```
void fileWrite() {
  // Function to write data to an SD card

}
```

- The `fileWrite()` function writes data to an SD card. It opens a file named "data.csv" and appends data such as date, time, latitude, longitude, speed, RPM, fuel level, accelerometer, gyroscope, and sensor value.

```
void gyro(){
  // Function to read gyroscope data from the MPU6050 module

}
```

- The `gyro()` function reads raw accelerometer and gyroscope data from the MPU6050 module and prints it to the Serial Monitor.

```
void upload() {
  // Function to upload data to a remote server

}
```

- The `upload()` function uploads data to a remote server using the SIM808 module. It establishes a TCP connection, formats the data, sends an HTTP POST request to the server, and closes the connection.

```
void getGPSData() {
  //

 Function to retrieve GPS data

}
```

- The `getGPSData()` function retrieves latitude, longitude, and speed data from the SIM808 module.

```
void msg() {
  // Function to send a message with GPS coordinates

}
```

- The `msg()` function sends an SMS message containing GPS coordinates to specified phone numbers.

```
void ShowSerialData() {
  // Function to read and display data from the SIM808 module

}
```

- The `ShowSerialData()` function reads data from the SIM808 module and displays it on the Serial Monitor.

The provided code sets up various components such as the DS3231 real-time clock, SIM808 module for GPS and SMS functionality, ELM327 module for ECU data retrieval, MPU6050 module for gyroscope data, and SD card module for data storage. It retrieves GPS data, ECU data, accelerometer and gyroscope data, and reads a vibration sensor. It then writes the collected data to an SD card, uploads it to a remote server, and sends SMS messages with GPS coordinates if certain conditions are met.