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

The `#include` statements are used to include necessary libraries for the code to access various functions and modules.

```
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
```

- `DS3231 rtc(2, 3);` creates an instance of the DS3231 class for accessing the DS3231 real-time clock. The parameters `2` and `3` specify the I2C SDA and SCL pins, respectively.
- `SoftwareSerial mySerial(10, 11);` creates a SoftwareSerial object named `mySerial` for communication with the SIM808 module. The parameters `10` and `11` specify the RX and TX pins, respectively.
- `DFRobot_SIM808 sim808(&mySerial);` creates an instance of the DFRobot_SIM808 class for accessing the SIM808 module. The `&mySerial` parameter specifies the SoftwareSerial object to be used for communication.
- `SoftwareSerial obdSerial(0,1);` creates another SoftwareSerial object named `obdSerial` for communication with the ELM327 module. The parameters `0` and `1` specify the RX and TX pins, respectively.
- `const int chipSelect = 53;` assigns the pin number `53` to the `chipSelect` variable. This pin is used for the SD card module.
- `const int sensorPin = A0;` assigns the pin number `A0` to the `sensorPin` variable. This pin is used for reading the analog sensor value.

```
#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;
```

- `#define PHONE_NUMBER1 "94769439154"` and `#define PHONE_NUMBER2 "94713651522"` are preprocessor directives that define the phone numbers to which SOS messages will be sent.
- `#define MESSAGE_LENGTH 5` defines the length of the SOS message.
- `#define ELM_PORT obdSerial` defines the communication port for the ELM327 module as `obdSerial`.
- `const int threshold = 800;` sets the vibration threshold value.
- `int sensorValue;` declares an integer variable to store the sensor value.
- `char latStr[15];` and `char lonStr[15];` declare character arrays to store latitude and longitude as strings.
- ‘char message[5];

‘ declares a character array to store the SOS message.

- `int messageIndex = 0;` initializes a variable to keep track of the index while constructing the SOS message.
- `char phone[16];` declares a character array to store the phone number for sending messages.
- `char datetime[24];` declares a character array to store the date and time.
- `String date;` and `String time;` declare String objects to store the date and time.
- `char MESSAGE[300];` declares a character array to store the final message to be sent.
- `float lat, lon, speed;` declare variables to store latitude, longitude, and speed.
- `ELM327 myELM327;` creates an instance of the ELM327 class for accessing the ELM327 module.
- `MPU6050 mpu;` creates an instance of the MPU6050 class for accessing the MPU6050 module.
- `int rpm = 0;` initializes the RPM value to 0.
- `float fuel = 0;` initializes the fuel level to 0.
- `int ax, ay, az;` declare variables to store accelerometer data.
- `int gx, gy, gz;` declare variables to store gyroscope data.

```
void setup() {
  Serial.begin(115200);
  rtc.begin();
```

```
  sim808.begin();
  obdSerial.begin(38400);
  mpu.initialize();
  pinMode(sensorPin, INPUT);
  pinMode(chipSelect, OUTPUT);
  if (!SD.begin(chipSelect)) {
    Serial.println("SD initialization failed!");
    return;
  }
  getDateTime();
}
```

- `void setup()` is a function that runs once when the Arduino board is powered on or reset.
- `Serial.begin(115200);` initializes the serial communication at a baud rate of 115200.
- `rtc.begin();` initializes the DS3231 real-time clock.
- `sim808.begin();` initializes the SIM808 module.
- `obdSerial.begin(38400);` initializes the ELM327 module with a baud rate of 38400.
- `mpu.initialize();` initializes the MPU6050 module.
- `pinMode(sensorPin, INPUT);` sets the sensor pin as an input.
- `pinMode(chipSelect, OUTPUT);` sets the chip select pin of the SD card module as an output.
- `if (!SD.begin(chipSelect)) { ... }` checks if the SD card initialization fails and prints an error message if it does.
- `getDateTime();` calls the `getDateTime()` function to retrieve the current date and time.

```
void loop() {
  getGPSData();
  getECUData();
  getAccelerometerData();
  getGyroscopeData();
  sensorValue = analogRead(sensorPin);
  if (sensorValue > threshold) {
    writeToSDCard();
    uploadToServer();
    getMessage();
    sendSMS();
  }
  delay(1000);
}
```

- `void loop()` is a function that runs repeatedly as long as the Arduino board is powered on.
- `getGPSData();` retrieves GPS data (latitude, longitude, and speed) from

the SIM808 module.

- `getECUData();` retrieves ECU data (RPM and fuel level) from the ELM327 module.
- `getAccelerometerData();` retrieves accelerometer data from the MPU6050 module.
- `getGyroscopeData();` retrieves gyroscope data from the MPU6050 module.
- `sensorValue = analogRead(sensorPin);` reads the analog sensor value and assigns

it to the `sensorValue` variable.

- `if (sensorValue > threshold) { ... }` checks if the sensor value is greater than the vibration threshold.
- Inside the `if` statement, the following actions are performed if the threshold is exceeded:
  - `writeToSDCard();` writes data to the SD card.
  - `uploadToServer();` uploads data to a server.
  - `getMessage();` constructs the SOS message.
  - `sendSMS();` sends the SOS message via SMS.
- `delay(1000);` adds a delay of 1 second before the next iteration of the loop.

This is a brief overview of the functions and code snippets in the provided code. Each function performs specific tasks such as initializing modules, retrieving data, processing data, and taking actions based on certain conditions.