```
// TTGO T-Call pin definitions
#define MODEM_RST            5
#define MODEM_PWKEY          4
#define MODEM_POWER_ON       23
#define MODEM_TX             27
#define MODEM_RX             26
#define I2C_SDA              21
#define I2C_SCL              22


#include <TinyGPS++.h> //https://github.com/mikalhart/TinyGPSPlus

#define BLYNK_PRINT Serial
#define BLYNK_HEARTBEAT 30
#define TINY_GSM_MODEM_SIM800
#define SIM800L_IP5306_VERSION_20190610

#include <TinyGsmClient.h> // https://github.com/vshymanskyy/TinyGSM
#include <BlynkSimpleSIM800.h> //https://github.com/blynkkk/blynk-library
#include <Wire.h>
#include "utilities.h"

//Buttons
#define buttonpin 34

// Emergency Number and Message
String message = "Accident Alert!! I'm at this location ";
String message1 = "Location ";
String message2 = "I am not safe!! Location ";
String mobile_number = "+91XXXXXXXXXX";  //Enter your Phone Number
String message_with_data;
```

```
// Variables for storing GPS Data
float latitude;
float longitude;
float speed;
float satellites;
String direction;


 const int MPU_addr = 0x68; // I2C address of the MPU-6050
 int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
 float ax = 0, ay = 0, az = 0, gx = 0, gy = 0, gz = 0;
 boolean accident = false; //stores if a accident has occurred
 boolean trigger1 = false; //stores if first trigger (lower threshold) has occurred
 boolean trigger2 = false; //stores if second trigger (upper threshold) has occurred
 boolean trigger3 = false; //stores if third trigger (orientation change) has occurred
 byte trigger1count = 0; //stores the counts past since trigger 1 was set true
 byte trigger2count = 0; //stores the counts past since trigger 2 was set true
 byte trigger3count = 0; //stores the counts past since trigger 3 was set true
 int angleChange = 0;


// Set serial for GPS Module
#define SerialMon Serial

// Hardware Serial for builtin GSM Module
#define SerialAT Serial1

// Define the serial console for debug prints, if needed
#define TINY_GSM_DEBUG SerialMon

// set GSM PIN, if any
```

```
#define GSM_PIN ""

const char apn[]  = "airtelgprs.com";    //Change with your network provider APN
const char user[] = "";
const char pass[] = "";

//Change this with your Blynk Credentials
#define BLYNK_TEMPLATE_ID "XXXXXXXXXXXX"
#define BLYNK_TEMPLATE_NAME "XXXXXXXXXXXX"
#define BLYNK_AUTH_TOKEN "XXXXXXXXXXXXXXXXXXXXXXX"

const char auth[] = BLYNK_AUTH_TOKEN;

//static const int RXPin = 4, TXPin = 5;
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;

BlynkTimer timer;

TinyGsm modem(SerialAT);

unsigned int move_index = 1;

void setup()
{
  // Set console baud rate
  Serial.begin(9600);
  delay(10);

  // Keep power when running from battery
  Wire.begin(I2C_SDA, I2C_SCL);
```

```
bool   isOk = setPowerBoostKeepOn(1);
SerialMon.println(String("IP5306 KeepOn ") + (isOk ? "OK" : "FAIL"));

// Set-up modem reset, enable, power pins
pinMode(MODEM_PWKEY, OUTPUT);
pinMode(MODEM_RST, OUTPUT);
pinMode(MODEM_POWER_ON, OUTPUT);

pinMode(buttonpin, INPUT);
digitalWrite(buttonpin, LOW);

digitalWrite(MODEM_PWKEY, LOW);
digitalWrite(MODEM_RST, HIGH);
digitalWrite(MODEM_POWER_ON, HIGH);

// Set GSM module baud rate and UART pins
SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);
delay(3000);

// Restart takes quite some time
// To skip it, call init() instead of restart()
SerialMon.println("Initializing modem...");
modem.restart();

String modemInfo = modem.getModemInfo();
SerialMon.print("Modem: ");
SerialMon.println(modemInfo);

// Unlock your SIM card with a PIN
//modem.simUnlock("1234");

SerialMon.print("Waiting for network...");
```

```
 if (!modem.waitForNetwork(240000L)) {
  SerialMon.println(" fail");
  delay(10000);
  return;
 }
 SerialMon.println(" OK");

 if (modem.isNetworkConnected()) {
  SerialMon.println("Network connected");
 }

 SerialMon.print(F("Connecting to APN: "));
 SerialMon.print(apn);
 if (!modem.gprsConnect(apn, user, pass)) {
  SerialMon.println(" fail");
  delay(10000);
  return;
 }
 SerialMon.println(" OK");
 //  ss.begin(GPSBaud);
 Blynk.begin(auth, modem, apn, user, pass, "blynk.cloud", 8080);
 timer.setInterval(5000L, checkGPS);


  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

}

void checkGPS()
```

```
{
  if (gps.charsProcessed() < 10)
  {
    //Serial.println(F("No GPS detected: check wiring."));
    Blynk.virtualWrite(V4, "GPS ERROR");
  }
}

void loop()
{
  while (Serial.available() > 0)
  {
    if (gps.encode(Serial.read()))
      displayInfo();
  }



  if (digitalRead (buttonpin) == HIGH) //Triggered with Push Button
    {
      Serial.println("I am not Safe");
      message_with_data = message2 + "Latitude = " + String(latitude, 6) + "Longitude = " +
String(longitude, 6) + " Link: https://maps.google.com/maps?&z=15&mrt=yp&t=k&q=" +
String(latitude, 6) + "," + String(longitude, 6);
      modem.sendSMS(mobile_number, message_with_data);
      message_with_data = "";
      modem.callNumber(mobile_number);
      delay(2000);
    }

  Blynk.run();
  timer.run();
```

```
  mpu_read();
  ax = (AcX - 2050) / 16384.00;
  ay = (AcY - 77) / 16384.00;
  az = (AcZ - 1947) / 16384.00;
  gx = (GyX + 270) / 131.07;
  gy = (GyY - 351) / 131.07;
  gz = (GyZ + 136) / 131.07;
  // calculating Amplitute vactor for 3 axis
  float Raw_Amp = pow(pow(ax, 2) + pow(ay, 2) + pow(az, 2), 0.5);
  int Amp = Raw_Amp * 10;  // Mulitiplied by 10 bcz values are between 0 to 1
  Serial.println(Amp);
if (Amp <= 2 && trigger2 == false) { //if AM breaks lower threshold (0.4g)
trigger1 = true;
Serial.println("Level 1 Trigger");
}
if (trigger1 == true) {
trigger1count++;
if (Amp >= 12) { //if AM breaks upper threshold (3g)
      trigger2 = true;
      Serial.println("Level 2 Trigger");
      trigger1 = false; trigger1count = 0;
    }
  }
  if (trigger2 == true) {
    trigger2count++;
    angleChange = pow(pow(gx, 2) + pow(gy, 2) + pow(gz, 2), 0.5); Serial.println(angleChange);
    if (angleChange >= 30 && angleChange <= 400) { //if orientation changes by between 80-100
degrees
trigger3 = true; trigger2 = false; trigger2count = 0;
Serial.println(angleChange);
Serial.println("Level 3 Trigger");
}
```

```
}
if (trigger3 == true) {
trigger3count++;
if (trigger3count >= 10) {
      angleChange = pow(pow(gx, 2) + pow(gy, 2) + pow(gz, 2), 0.5);
      //delay(10);
      Serial.println(angleChange);
      if ((angleChange >= 0) && (angleChange <= 10)) { //if orientation changes remains between 0-10 degrees
accident = true; trigger3 = false; trigger3count = 0;
Serial.println(angleChange);      }
else { //user regained normal orientation
trigger3 = false; trigger3count = 0;
Serial.println("TRIGGER 3 DEACTIVATED");
}
}
}
if (accident == true) { //in event of a Accident Detection
Serial.println("ACCIDENT DETECTED");
accident = false;
     Serial.println("Accident Alert!!");
     message_with_data = message + "Latitude = " + String(latitude, 6) + "Longitude = " + String(longitude, 6) + " Link: https://maps.google.com/maps?&z=15&mrt=yp&t=k&q=" + String(latitude, 6) + "," + String(longitude, 6);
     modem.sendSMS(mobile_number, message_with_data);
     message_with_data = "";
     modem.callNumber(mobile_number);
     delay(2000);
}
if (trigger2count >= 6) { //allow 0.5s for orientation change
    trigger2 = false; trigger2count = 0;
    Serial.println("TRIGGER 2 DECACTIVATED");
  }
```

```
  if (trigger1count >= 6) { //allow 0.5s for AM to break upper threshold
    trigger1 = false; trigger1count = 0;
    Serial.println("TRIGGER 1 DECACTIVATED");
  }
  delay(100);
}

void mpu_read()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
  AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
 }


void displayInfo()
{
 if (gps.location.isValid() )
 {
   latitude = (gps.location.lat());
   longitude = (gps.location.lng());
   speed = gps.speed.kmph();
   Blynk.virtualWrite(V1, String(latitude, 6));
   Blynk.virtualWrite(V2, String(longitude, 6));
```

```
    Blynk.virtualWrite(V3, speed);
    Blynk.virtualWrite(V6, "https://maps.google.com/maps?&z=15&mrt=yp&t=k&q=" +
String(latitude, 6) + "," + String(longitude, 6));
    Blynk.virtualWrite(V9, String(longitude, 6), String(latitude, 6));
  }

}
BLYNK_WRITE(V7) // this command is listening when something is written to V7
{
  int pinValue = param.asInt(); // assigning incoming value from pin V7 to a variable

  if (pinValue == 1)
  {
  Serial.println("Sent Location");
   message_with_data = message1 + "Latitude = " + String(latitude, 6) + "Longitude = " +
String(longitude, 6) + " Link: https://maps.google.com/maps?&z=15&mrt=yp&t=k&q=" +
String(latitude, 6) + "," + String(longitude, 6);
   modem.sendSMS(mobile_number, message_with_data);
   message_with_data = "";
  }
}
```