i have code for ARDUINO UNO AND ADXL345......AND ARDUINO UNO AND GSMSIM900  
  
but i want to run as one full code   
  
how to do this?....this below for ADXL 345 WITH ARDUINO UNO  
#include <SPI.h>  
  
//Assign the Chip Select signal to pin 10.  
int CS=10;  
  
//This is a list of some of the registers available on the ADXL345.  
//To learn more about these and the rest of the registers on the ADXL345, read the datasheet!  
char POWER\_CTL = 0x2D;   //Power Control Register  
char DATA\_FORMAT = 0x31;  
char DATAX0 = 0x32;   //X-Axis Data 0  
char DATAX1 = 0x33;   //X-Axis Data 1  
char DATAY0 = 0x34;   //Y-Axis Data 0  
char DATAY1 = 0x35;   //Y-Axis Data 1  
char DATAZ0 = 0x36;   //Z-Axis Data 0  
char DATAZ1 = 0x37;   //Z-Axis Data 1  
  
//This buffer will hold values read from the ADXL345 registers.  
char values[10];  
//These variables will be used to hold the x,y and z axis accelerometer values.  
int x,y,z;  
  
void setup(){   
  //Initiate an SPI communication instance.  
  SPI.begin();  
  //Configure the SPI connection for the ADXL345.  
  SPI.setDataMode(SPI\_MODE3);  
  //Create a serial connection to display the data on the terminal.  
  Serial.begin(115200);  
    
  //Set up the Chip Select pin to be an output from the Arduino.  
  pinMode(CS, OUTPUT);  
  //Before communication starts, the Chip Select pin needs to be set high.  
  digitalWrite(CS, HIGH);  
    
  //Put the ADXL345 into +/- 4G range by writing the value 0x01 to the DATA\_FORMAT register.  
  writeRegister(DATA\_FORMAT, 0x01);  
  //Put the ADXL345 into Measurement Mode by writing 0x08 to the POWER\_CTL register.  
  writeRegister(POWER\_CTL, 0x08);  //Measurement mode    
}  
  
void loop(){  
  //Reading 6 bytes of data starting at register DATAX0 will retrieve the x,y and z acceleration values from the ADXL345.  
  //The results of the read operation will get stored to the values[] buffer.  
  readRegister(DATAX0, 6, values);  
  
  //The ADXL345 gives 10-bit acceleration values, but they are stored as bytes (8-bits). To get the full value, two bytes must be combined for each axis.  
  //The X value is stored in values[0] and values[1].  
  x = ((int)values[1]<<8)|(int)values[0];  
  //The Y value is stored in values[2] and values[3].  
  y = ((int)values[3]<<8)|(int)values[2];  
  //The Z value is stored in values[4] and values[5].  
  z = ((int)values[5]<<8)|(int)values[4];  
    
  //Print the results to the terminal.  
  Serial.print(x, DEC);  
  Serial.print(',');  
  Serial.print(y, DEC);  
  Serial.print(',');  
  Serial.println(z, DEC);        
  delay(10);   
}  
  
//This function will write a value to a register on the ADXL345.  
//Parameters:  
//  char registerAddress - The register to write a value to  
//  char value - The value to be written to the specified register.  
void writeRegister(char registerAddress, char value){  
  //Set Chip Select pin low to signal the beginning of an SPI packet.  
  digitalWrite(CS, LOW);  
  //Transfer the register address over SPI.  
  SPI.transfer(registerAddress);  
  //Transfer the desired register value over SPI.  
  SPI.transfer(value);  
  //Set the Chip Select pin high to signal the end of an SPI packet.  
  digitalWrite(CS, HIGH);  
}  
  
//This function will read a certain number of registers starting from a specified address and store their values in a buffer.  
//Parameters:  
//  char registerAddress - The register addresse to start the read sequence from.  
//  int numBytes - The number of registers that should be read.  
//  char \* values - A pointer to a buffer where the results of the operation should be stored.  
void readRegister(char registerAddress, int numBytes, char \* values){  
  //Since we're performing a read operation, the most significant bit of the register address should be set.  
  char address = 0x80 | registerAddress;  
  //If we're doing a multi-byte read, bit 6 needs to be set as well.  
  if(numBytes > 1)address = address | 0x40;  
    
  //Set the Chip select pin low to start an SPI packet.  
  digitalWrite(CS, LOW);  
  //Transfer the starting register address that needs to be read.  
  SPI.transfer(address);  
  //Continue to read registers until we've read the number specified, storing the results to the input buffer.  
  for(int i=0; i<numBytes; i++){  
    values *= SPI.transfer(0x00);  
  }  
  //Set the Chips Select pin high to end the SPI packet.  
  digitalWrite(CS, HIGH);  
}  
THIS IS FOR ARDUINO UNO AND SIM900  
[/#include "SIM900.h"  
#include <SoftwareSerial.h>  
//If not used, is better to exclude the HTTP library,  
//for RAM saving.  
//If your sketch reboots itself proprably you have finished,  
//your memory available.  
//#include "inetGSM.h"  
  
//If you want to use the Arduino functions to manage SMS, uncomment the lines below.  
#include "sms.h"  
SMSGSM sms;  
  
//To change pins for Software Serial, use the two lines in GSM.cpp.  
  
//GSM Shield for Arduino  
//www.open-electronics.org  
//this code is based on the example of Arduino Labs.  
  
//Simple sketch to send and receive SMS.  
  
int numdata;  
boolean started=false;  
char smsbuffer[160];  
char n[20];  
  
void setup()   
{  
  //Serial connection.  
  Serial.begin(9600);  
  Serial.println("GSM Shield testing.");  
  //Start configuration of shield with baudrate.  
  //For http uses is raccomanded to use 4800 or slower.  
  if (gsm.begin(2400)){  
    Serial.println("\nstatus=READY");  
    started=true;    
  }  
  else Serial.println("\nstatus=IDLE");  
    
  if(started){  
    //Enable this two lines if you want to send an SMS.  
    if (sms.SendSMS("+971505393365", "Arduino SMS"))  
      Serial.println("\nSMS sent OK");  
  }  
  
};  
  
void loop()   
{  
  if(started){  
    //Read if there are messages on SIM card and print them.  
    if(gsm.readSMS(smsbuffer, 160, n, 20))  
    {  
      Serial.println(n);  
      Serial.println(smsbuffer);  
    }  
    delay(1000);  
  }  
};*