# Código del Arduino

Se adjunta el código de envío de datos de los sensores, "Ultrasonic Distance Sensor" y "Grove - IMU 9DOF", a través de cable USB-MicroUSB:

**sensor.ino**

```
 /**************************************************************************
 *
 * Interfacing Arduino Grove  ultrasonic ranger.
 * Distance value is printed in centimeters Arduino IDE serial monitor.
 * This is a free software with NO WARRANTY.
 * https://simple-circuit.com/
 *
 ************************************************************************/
#include "Ultrasonic.h"  // include Seeed Studio ultrasonic ranger library
#include "MPU6050.h"
#include "I2Cdev.h"
#include "Wire.h"

Ultrasonic ultrasonic(7);
MPU6050 accelgyro;
I2Cdev I2C_M;

uint8_t buffer_m[6];


int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t   mx, my, mz;



float heading;
float tiltheading;

float Axyz[3];
float Gxyz[3];
float Mxyz[3];


#define sample_num_mdate  5000

volatile float mx_sample[3];
volatile float my_sample[3];
volatile float mz_sample[3];

static float mx_centre = 0;
static float my_centre = 0;
static float mz_centre = 0;

volatile int mx_max = 0;
volatile int my_max = 0;
volatile int mz_max = 0;

volatile int mx_min = 0;
volatile int my_min = 0;
volatile int mz_min = 0;


void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  Wire.begin();
  // open serial communication
  Serial.begin(9600);

  // initialize device
    while(!Serial);
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();
    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

    delay(1000);
    Serial.println("     ");

    Mxyz_init_calibrated();
}

void loop() {
  long centimetros;

  /************************************/
  /********* OBTENCIÓN DATOS **********/
  /************************************/

  // Coger la distancia en centimetros
  centimetros = ultrasonic.MeasureInCentimeters();

  getAccel_Data();
  getGyro_Data();
  getCompassDate_calibrated(); // compass data has been calibrated here
  getHeading();                                //before we use this function we should run
'getCompassDate_calibrated()' frist, so that we can get calibrated data ,then we can get correct angle .
  getTiltHeading();

  /************************************/
  /********* IMPRESION DATOS **********/
  /************************************/

  // Imprimo distnacias al monitor serial
  Serial.println("Distancia: ");
  Serial.println(centimetros);

  Serial.println("Parámetro de calibrado: ");
  Serial.print(mx_centre);
  Serial.print("         ");
  Serial.print(my_centre);
  Serial.print("         ");
  Serial.println(mz_centre);
  Serial.println("     ");


  Serial.println("Aceleración (g) en X, Y, Z:");
  Serial.print(Axyz[0]);
  Serial.print(",");
  Serial.print(Axyz[1]);
  Serial.print(",");
  Serial.println(Axyz[2]);
  Serial.println("Giroscopio (grados/s) en X, Y, Z:");
  Serial.print(Gxyz[0]);
  Serial.print(",");
  Serial.print(Gxyz[1]);
  Serial.print(",");
  Serial.println(Gxyz[2]);
  Serial.println("Valor de la brújula en X, Y, Z:");
  Serial.print(Mxyz[0]);
  Serial.print(",");
  Serial.print(Mxyz[1]);
  Serial.print(",");
  Serial.println(Mxyz[2]);
  Serial.println("El ángulo en sentido horario entre el norte magnético y el eje X:");
  Serial.print(heading);
  Serial.println(" ");
  Serial.println("El ángulo en sentido horario entre el norte magnético y la proyección del eje X positivo en
el plano horizontal:");
  Serial.println(tiltheading);
  Serial.println("   ");
  Serial.println("   ");
  Serial.println("   ");


  delay(10000);  // Esperar 250 ms entre lecturas
}

void getHeading(void) {
    heading = 180 * atan2(Mxyz[1], Mxyz[0]) / PI;
    if (heading < 0) {
        heading += 360;
    }
}

void getTiltHeading(void) {
    float pitch = asin(-Axyz[0]);
    float roll = asin(Axyz[1] / cos(pitch));

    float xh = Mxyz[0] * cos(pitch) + Mxyz[2] * sin(pitch);
    float yh = Mxyz[0] * sin(roll) * sin(pitch) + Mxyz[1] * cos(roll) - Mxyz[2] * sin(roll) * cos(pitch);
    float zh = -Mxyz[0] * cos(roll) * sin(pitch) + Mxyz[1] * sin(roll) + Mxyz[2] * cos(roll) * cos(pitch);
    tiltheading = 180 * atan2(yh, xh) / PI;
    if (yh < 0) {
        tiltheading += 360;
    }
}



void Mxyz_init_calibrated() {

    Serial.println(F("Before using 9DOF,we need to calibrate the compass frist,It will takes about 2 
minutes."));
    Serial.print("  ");
    Serial.println(F("During  calibratting ,you should rotate and turn the 9DOF all the time within 2 
minutes."));
    Serial.print("  ");
    Serial.println(F("If you are ready ,please sent a command data 'ready' to start sample and calibrate."));
    //while (!Serial.find("ready"));
    Serial.println("  ");
    Serial.println("ready");
    Serial.println("Sample starting......");
    Serial.println("waiting ......");

    get_calibration_Data();

    Serial.println("     ");
    Serial.println("compass calibration parameter ");
    Serial.print(mx_centre);
    Serial.print("     ");
    Serial.print(my_centre);
    Serial.print("     ");
    Serial.println(mz_centre);
    Serial.println("    ");
}


void get_calibration_Data() {
    for (int i = 0; i < sample_num_mdate; i++) {
        get_one_sample_date_mxyz();
        /*
            Serial.print(mx_sample[2]);
            Serial.print(" ");
            Serial.print(my_sample[2]);                            //you can see the sample data here .
            Serial.print(" ");
            Serial.println(mz_sample[2]);
        */



        if (mx_sample[2] >= mx_sample[1]) {
            mx_sample[1] = mx_sample[2];
        }
        if (my_sample[2] >= my_sample[1]) {
            my_sample[1] = my_sample[2];    //find max value
        }
        if (mz_sample[2] >= mz_sample[1]) {
            mz_sample[1] = mz_sample[2];
        }

        if (mx_sample[2] <= mx_sample[0]) {
            mx_sample[0] = mx_sample[2];
        }
        if (my_sample[2] <= my_sample[0]) {
            my_sample[0] = my_sample[2];    //find min value
        }
        if (mz_sample[2] <= mz_sample[0]) {
            mz_sample[0] = mz_sample[2];
        }

    }

    mx_max = mx_sample[1];
    my_max = my_sample[1];
    mz_max = mz_sample[1];

    mx_min = mx_sample[0];
    my_min = my_sample[0];
    mz_min = mz_sample[0];



    mx_centre = (mx_max + mx_min) / 2;
    my_centre = (my_max + my_min) / 2;
    mz_centre = (mz_max + mz_min) / 2;

}






void get_one_sample_date_mxyz() {
    getCompass_Data();
    mx_sample[2] = Mxyz[0];
    my_sample[2] = Mxyz[1];
    mz_sample[2] = Mxyz[2];
}


void getAccel_Data(void) {
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    Axyz[0] = (double) ax / 16384;
    Axyz[1] = (double) ay / 16384;
    Axyz[2] = (double) az / 16384;
}

void getGyro_Data(void) {
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);
    Gxyz[0] = (double) gx * 250 / 32768;
    Gxyz[1] = (double) gy * 250 / 32768;
    Gxyz[2] = (double) gz * 250 / 32768;
}

void getCompass_Data(void) {
    I2C_M.writeByte(MPU9150_RA_MAG_ADDRESS, 0x0A, 0x01); //enable the magnetometer
    delay(10);
    I2C_M.readBytes(MPU9150_RA_MAG_ADDRESS, MPU9150_RA_MAG_XOUT_L, 6, buffer_m);

    mx = ((int16_t)(buffer_m[1]) << 8) | buffer_m[0] ;
    my = ((int16_t)(buffer_m[3]) << 8) | buffer_m[2] ;
    mz = ((int16_t)(buffer_m[5]) << 8) | buffer_m[4] ;
    Mxyz[0] = (double) mx * 1200 / 4096;
    Mxyz[1] = (double) my * 1200 / 4096;
    Mxyz[2] = (double) mz * 1200 / 4096;
}

void getCompassDate_calibrated() {
    getCompass_Data();
    Mxyz[0] = Mxyz[0] - mx_centre;
    Mxyz[1] = Mxyz[1] - my_centre;
    Mxyz[2] = Mxyz[2] - mz_centre;
}
```