第 1 章　序論

1.1　始めに

人工衛星とは，地球の周りをまわっている人工物体のことをいい，搭載しているセンサなどで地球の気象や地上の状況を調べる地球観測衛星，位置情報を正確に測る測位衛星，インターネットなどを構成する通信衛星などがある．内閣府の調査によれば，2023 年に打ち上げられた人工衛星等の機数は，過去最大の 2,901 機であり，10 年前と比べて約 14 倍に増加したという．数十〜数千の人工衛星を一体的に運用しネットワークを構築する，メガコンステレーションの構築への取り組みや，先進国による宇宙利用への期待の高まりから，宇宙輸送のニーズは一層拡大することが見込まれている．

1.2　超小型人工衛星とは

人工衛星は，大きいものでは例えば大きさが約 108.5 m × 72.8 m の ISS（国際宇宙ステーション）がある．それに対し小さいものでは，一辺 10cm の立方体サイズの超小型人工衛星が存在している．超小型人工衛星の定義は明確には決まっておらず，全質量が 100 kg 以下とするものや，50 kg 以下とするものもある．本研究では，超小型人工衛星の中でも，10 cm × 10 cm × 10 cm の立方体を 1U として規格化された小型衛星に絞って述べる．大型人工衛星，中型人工衛星と呼ばれる人工衛星は，多くが国家プロジェクトとして開発され，打ち上げられている．その特徴として，高機能で信頼性が高く，複雑な用途，複数のミッションに活用できるという利点がある．しかし，設計・製造に莫大な費用と時間が必要となる．反対に，超小型人工衛星は，機能が制限され，単一のミッションしか実行できないという欠点がある．その代わり，短期間での開発・低コストでの打ち上げが可能であるため，学生，大学院生が在学中に開発・打ち上げをすることが可能となっている．また，世界各国が宇宙開発研究でしのぎを削るなか，新しい技術を素早く試せることも，注目されている一つの理由である．
人工衛星は，多くの場合，姿勢の制御を必要とする．例えば地球との通信のためのアンテナや，発電のための太陽光パネルの向きを制御するために行われる．人工衛星の姿勢制御の方式には，主に以下のようなものがある．

- 重力傾度姿勢安定方式
  軌道を廻る衛星が地球の重力により衛星の軸のうち一つが常に地球の中心を向く性質を利用するもの．能動的な制御やエネルギーを必要としない．
- スピン安定方式
  慣性主軸の回りにスピンを与えて，コマのように安定させる方式．衛星全体を回転させるシングルスピン方式と，通信用のアンテナを回転させないため，アンテナ部と衛星本体をそれぞれ違う方向に回転させるデュアルスピン方式がある．

- 3 軸安定方式
  衛星の直交する 3 軸（ロール・ピッチ・ヨー角）の各軸について制御する方式である．リアクションホイールを用いるものには，ジャイロ効果を用いるバイアスモーメンタム方式と，外乱に応じて回転数を変化させるゼロモーメンタム方式がある．磁気トルカによる制御は，3 軸安定方式にあたる．

特に 3 軸安定方式に用いられるアクチュエータには，スラスター，リアクションホイール，コントロールモーメントジャイロ，磁気トルカといったものがある．スラスターは，推進剤を噴出し，その反力で推進力を得るものである．またリアクションホイールは，ホイールを回転させて生ずる反力によるトルクで回転を起こし，コントロールモーメントジャイロは，その回転の向きを変えることにより姿勢を制御する．磁気トルカは，電磁誘導を利用してコイルに発生させた磁場と地磁気を反応させることでトルクを発生させ，回転力を得るものである．超小型人工衛星は，その小ささから，搭載可能な機器のサイズや重量に制限がある．そのため，リアクションホイールやスラスタといった，性能の代わりに大型な機器は搭載が難しい．そこで，姿勢制御に磁気トルカと呼ばれる電磁石や CMG（コントロールモーメントジャイロ）が用いられることが多い．

1.3 本研究の目的

人工衛星の姿勢制御を地上で実験し，研究・検討するには，通常球面の空気軸受けを使った 3 軸テーブルが用いられる．しかし，3 軸で制御する衛星でも，3 軸を同時に制御する状態は必要とせず，1 軸のみでの制御実験を先に検証する場合が多い．また，3 軸テーブルでの実験の設計の難しさから，1 軸テーブルは有効な実験道具となる．

本研究では，小型衛星の模型を用いて，磁気トルカで衛星の姿勢制御を行う様子を再現し，人工衛星によく用いられる，B-dot 制御則やクロスプロダクト則といった制御理論を検証する．実際の人工衛星の動きを完璧にシミュレーションするわけではなく，

第 2 章　実験システムの構築

2.1　構造

本研究では，図 2.1 に示すような超小型人工衛星の模型を用いて実験を行う．模型は，転がり軸受を利用してできるだけ滑らかに回転し，各制御理論の特性が摩擦の影響を受けることなくなるべく反映されるようになっている．人工衛星に搭載している物は，Arduino Uno，SD Card シールド，磁気・角度・角速度センサ，磁気トルカ駆動用の回路，磁気トルカ，9 V 角型乾電池である．Arduino への電力供給は角型乾電池を用い，磁気トルカへの電力供給は菊水電子工業株式会社製の PMC18-5A を用いる．

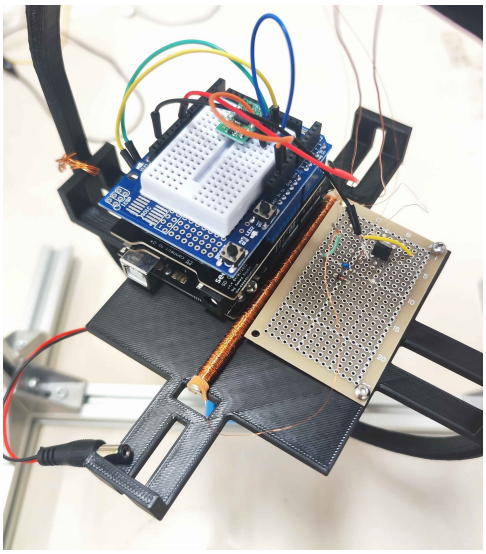

図 2.1　実験システムの外観

2.2　制御システムの設計

搭載する Arduino は，ELEGOO 社製の Arduino Uno R3 を，開発環境には Arduino IDE を用いた．姿勢角度，角速度および磁力の検出には，Bosch Sensortec 社の BNO055 と，Arduino IDE のライブラリ ”Adafruit_Sensor.h”，”Adafruit_BNO055.h” を用いる．また，姿勢角度，角速度および磁力の記録には SD カードを用いており，seeed studio 社製の SD Card shield V4.0 を Arduino に装着している．センサから取得した角度，角速度および磁気のデータを用いて，フィードバック制御を行う．記録した角度・角速度のデータを CSV ファイルに保存，それを Python でグラフに描画し，姿勢角度の遷移を確認する．

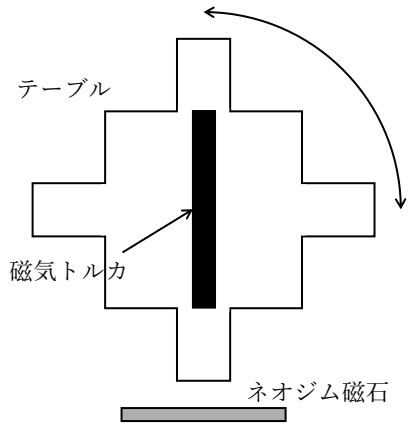

図 2.2 実験システム概略図

2.2.1 磁気トルカの制作

搭載した磁気トルカのパラメータを表 2.1 に示す．磁気トルカが発生させられるトルクは，磁気トルカの磁気モーメントを $\boldsymbol{M}$，磁気トルカがとらえる磁束密度を $\boldsymbol{B}$ とすると，

$$\boldsymbol{T} = \boldsymbol{M} \times \boldsymbol{B} \tag{2.1}$$

とあらわされる．

まず，芯材にステンレス鋼を用いて磁気トルカを作り，磁石を用いて反応性を確かめた．しかし，ステンレス鋼の保磁力の高さのため，磁気トルカに電流を流していないときも磁力を持っており，また比透磁率も低く，電流を流しているとき，いないときの差がなく磁力も低かったため，これを用いず PC パーマロイを用いて磁気トルカの制作を行った．

芯材である PC パーマロイの選定理由は，その透磁率の高さである．PC パーマロイは比透磁率が約 200,000 で，磁気トルカがとらえられる磁力が高くなる．また，保磁力が低く，消磁しやすいため，必要な時だけトルクを発生させられる．

表 2.1 磁気トルカのパラメータ

コイル長さ L [mm]	100
コイル直径 D [mm]	5
巻き数 n [-]	983
インダクタンス L [mH]	82.017
抵抗 [Ω]	18.3
芯材	PC パーマロイ
線材	ポリエステル被覆銅線

2.3　磁気トルカの駆動回路

2.3.1　駆動回路の設計

図 2.3 に，最初に構築した回路図を示す．この回路で，duty 比 50%の PWM 信号を入力した場合の，磁気トルカにかかる電圧を図 2.4 に示す．なお，電圧のレンジは 5 [V/div] である．図 2.4 のとおり，磁気トルカの逆起電力が大きい．そのため，FET が OFF のとき，磁気トルカに流れる電流が本来流すべき方向と逆方向に流れ，逆方向のトルクが発生することが予測された．

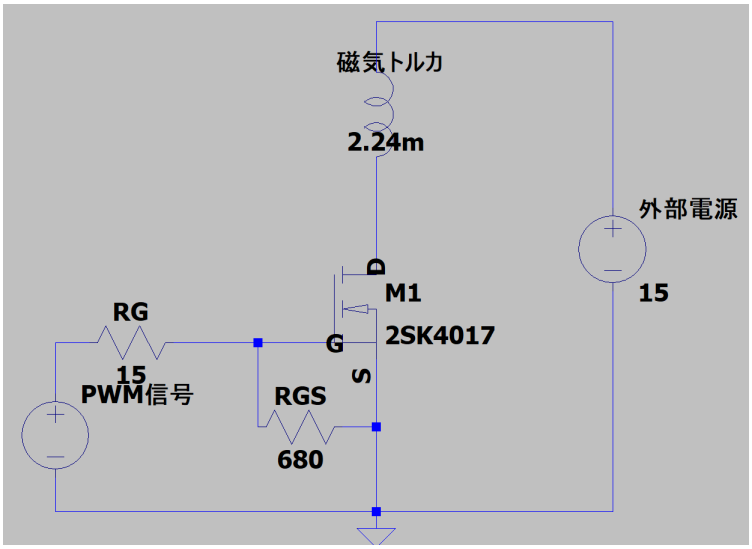

図 2.3　初期の磁気トルカの駆動回路

図 2.4　初期案

そこで次に，磁気トルカと並列にダイオードを接続することで逆起電力を防止した．制作した回路の回路図を図 2.6 に示す．具体的には，ON 時に磁気トルカに蓄えられたエネルギーを，ダイオードを通して閉回路となった磁気トルカの抵抗により消費することにより，逆起電力を防ぐ．

2.3.2　磁気トルカの電流計算など

電流の制御を PWM で行うため，コイルの過渡特性を考慮して電流値の計算を行う必要がある．一般的にコイルは，インダクタ成分のみでなく抵抗成分も含んでおり，低周波での等価回路は図 2.5 のようになる．よって，LR 直列回路の過渡現象を考えればよい．外部電源の電圧値 E [V]，抵抗成分 $R_L[\Omega]$，インダクタンス L [mH] から求められる FET が ON のときの電流の過渡特性は，

$$i(t) = \frac{E}{R_L}\left(1 - e^{-\frac{R_L}{L}t}\right) \tag{2.2}$$

で表される．また，FET が OFF となったときの時刻を t_1 [s] とすると，その後の電流の過渡特性は，

$$i(t) = \frac{E\left(1 - e^{-\frac{R_L}{L}t_1}\right)}{R_L} e^{-\frac{R_L}{L}(t-t_1)} \tag{2.3}$$

となる．さらに t_2 [s] 経過し，

$$i(t) = \frac{E\left(1 - e^{-\frac{R_L}{L}t_1}\right) e^{-\frac{R_L}{L}(t_2-t_1)}}{R_L}\left(1 - e^{-\frac{R_L}{L}(t-t_1-t_2)}\right) \tag{2.4}$$

というような電流の推移をする．

表 2.2　磁気トルカの等価回路

インダクタンス L [mH]	82.0
抵抗 R_L [Ω]	18.3

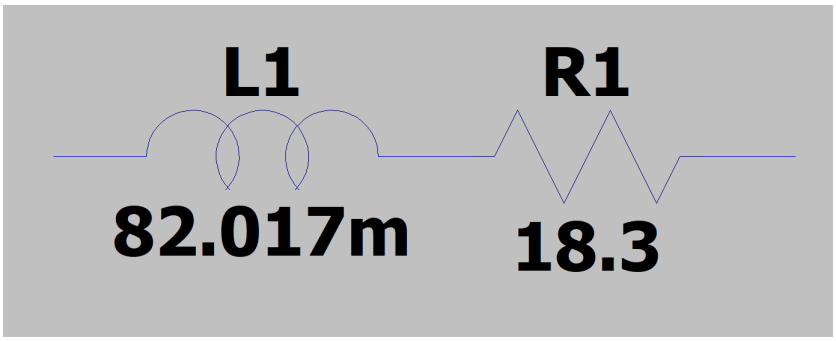

図 2.5　磁気トルカの等価回路

この電流の推移を，図 2.6 の回路を用いて duty 比 50%でシミュレーションしたグラフを図 2.8 に示す．このように，磁気トルカに電圧 E [V] を印加したときの電流を $I_{\max}[A]$ とすると，duty 比 D の PWM 信号で磁気トルカを駆動させたとき，およその電流を $I(\mathrm{D}) = I_{\max}D[A]$ で近似できる．

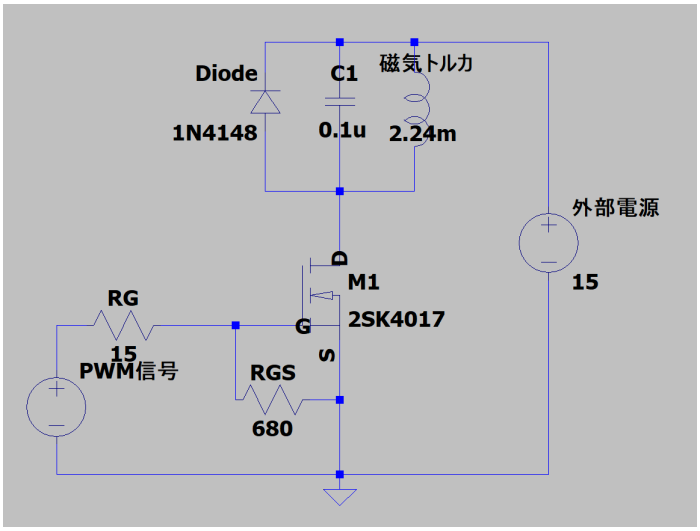

図 2.6　磁気トルカの駆動回路

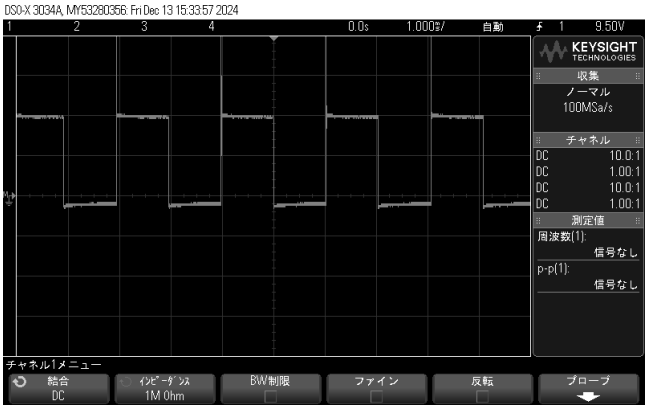

図 2.7　磁気トルカに加わる電圧

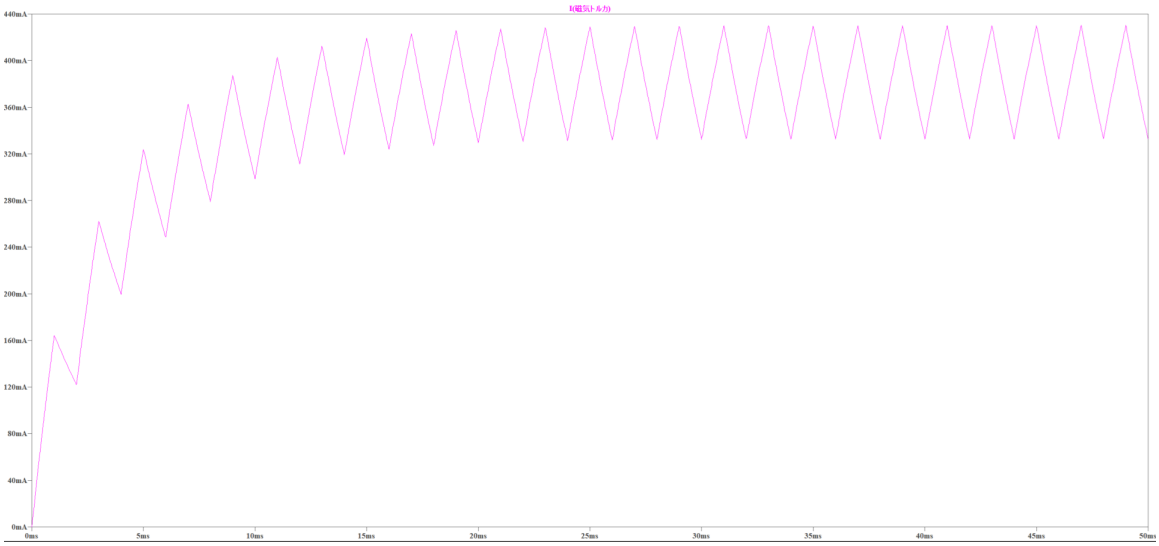

図 2.8　磁気トルカに流れる電流

参 考 文 献

1) 佐藤太郎：高等専門学校における一般科目と専門科目，京都出版 (2018) ····· 本の場合

2) 鈴木次郎, 高橋三郎：高等専門学校と大学の違い, 電子制御学会論文誌, Vol. 25, No. 13, 123/130 (2017) ····· 学会誌論文の場合

3) 田中史朗，伊藤五郎，渡辺花子：高専における就職活動，電気電子工学講演会資料，543/546 (2016) ····· 講演会等資料（ページが記載されているもの）の場合

4) 山本一二三，中村五十六：高専における就職活動，メカトロニクス講演会資料，全 4 頁 (2016) ····· 講演会等資料（ページが記載されていないもの）の場合

5) 中村十三子：機械加工と実習工場，平成 29 年度舞鶴工業高等専門学校機械工学科卒業論文 (2018) ····· 卒業論文の場合

6) T. Sato: General Subjects and Special subjects at National Institute of Technology, Kyoto Publishing (2018)

7) J. Suzuki and S. Takahashi: Difference between National Institute of Technology and Universities, Journal of the Electronic Control Society, Vol. 25, No. 13, 123/130 (2017)

8) S. Tanaka, G. Ito and H. Watanabe: Job Hunting in NIT, Proceedings of Conference on Electrical and Electronics Engineering, 543/546 (2016)

9) H. Yamamoto and I. Nakamura: Job Hunting in NIT, Proceedings of Mechatoronics Conference, 4 pages (2016)

10) 舞鶴高専ホームページ：`http://www.maizuru-ct.ac.jp`

謝 辞

ここに謝辞を記入する（必須ではない）．

付 録

A.1 ちちち

以下に，本研究で使用した Arduino 用の C++プログラムを示す.

```
    #include <Wire.h>
#include <SD.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#define PWM 9
#define VMAX 15.0f
#define RESISTER 18.37

double T=0.0327438; //実行時間
double e_pre = 0.0; // 微分の近似計算のための初期値
double de = 0.0;
double ie = 0.0; // 積分の近似計算のための初期値
const int analogWriteStep = 255;
int dutyRatio = 80; //duty比[%]
double angle[3]={-1000,-1000,-1000};
double magne[3]={-1000,-1000,-1000};
double gyro[3]={-1000,-1000,-1000};
double bdot[3]={0,0,0};
unsigned long int time;
const int chipSelect = 10;
double kb = 0.00005;
double kp = 0.007;
double kd = 0.004;
double ki = 0.001;
double current = 0;
double voltage = 0;
//string型宣言
String datastr = "";
// Check I2C device address and correct line below (by default address is 0x29
    or 0x28)
// id, address
Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28, &Wire);

void setup(void)
{
  pinMode(10, OUTPUT);
  pinMode(9,OUTPUT);
  Serial.begin(115200);
  Serial.println("Orientation Sensor Test"); Serial.println("");
  //センサの初期化
  if (!bno.begin())
```

```
  {
    /* There was a problem detecting the BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR
        !");
    while (1);
  }
  //SD カードの初期化
  if (!SD.begin(chipSelect)) {
    Serial.println("SD card initialization failed!");
    while (1); // 初期化に失敗したら停止
  }
  Serial.println("initialization done.");
}

void loop(void){
  //could add VECTOR_ACCELEROMETER, VECTOR_MAGNETOMETER,VECTOR_GRAVITY...
  sensors_event_t orientationData , angVelocityData , linearAccelData,
      magnetometerData, accelerometerData, gravityData;
  bno.getEvent(&orientationData, Adafruit_BNO055::VECTOR_EULER);
  bno.getEvent(&angVelocityData, Adafruit_BNO055::VECTOR_GYROSCOPE);
  bno.getEvent(&linearAccelData, Adafruit_BNO055::VECTOR_LINEARACCEL);
  bno.getEvent(&magnetometerData, Adafruit_BNO055::VECTOR_MAGNETOMETER);
  bno.getEvent(&accelerometerData, Adafruit_BNO055::VECTOR_ACCELEROMETER);
  bno.getEvent(&gravityData, Adafruit_BNO055::VECTOR_GRAVITY);

  //センサの値を取得
  returnEvent(&orientationData,angle);
  returnEvent(&magnetometerData,magne);
  returnEvent(&angVelocityData,gyro);

  //Bdot 制御則の計算
  bdot[0]=magne[1]*gyro[2]-magne[2]*gyro[1];
  bdot[1]=magne[0]*gyro[2]-magne[2]*gyro[0];
  bdot[2]=magne[0]*gyro[1]-magne[1]*gyro[0];

  //目標角度を 0[deg]とする
  double e = angle[0];
  if(e > 270){
    e = abs(e - 360);
  }
  de = (e - e_pre)/T;
  ie = ie + (e + e_pre)*T/2;

  //B-dot ならコメントアウト
  current = kp*e + ki*ie + kd*de;
  //PID ならコメントアウト
  // current = kb*bdot[0]/0.04935;

  //算出した電流値を電圧値に変換
  voltage = current * RESISTER;
```

```
90    if(abs(voltage) > VMAX) {
91      voltage = VMAX;
92    }
93    //電圧値をduty 比に変換
94    int duty = int(abs((voltage / 15.0) * 100.0));
95    Serial.print(duty);
96
97    int ledBrightness = analogWriteStep * duty / 100.0f;
98    analogWrite( 9, ledBrightness );
99
100   //値をstring 型に
101   datastr = String(angle[0]);
102   for (int i = 0; i < 3; i++){
103     datastr += ","+String(magne[i]);
104   }
105   datastr += ","+String(gyro[2]);
106   //csv ファイルの open
107   File angFile = SD.open("LOG.CSV", FILE_WRITE);
108   //書き込み
109   if (angFile) {
110     angFile.println(datastr);
111     angFile.close(); // ファイルを閉じる
112     Serial.println("Write successful.");
113   } else {
114     Serial.println("Error opening test.txt for writing.");
115   }
116   e_pre = e;
117 }
118
119 //センサの値を返す
120 void returnEvent(sensors_event_t* event,double tmp[3]) {
121   double x = -1000000, y = -1000000 , z = -1000000; //dumb values, easy to
          spot problem
122   if (event->type == SENSOR_TYPE_ACCELEROMETER) {
123     Serial.print("Accl:");
124     x = event->acceleration.x;
125     y = event->acceleration.y;
126     z = event->acceleration.z;
127   }
128   else if (event->type == SENSOR_TYPE_ORIENTATION) {
129     // Serial.print("orient:");
130     x = event->orientation.x;
131     y = event->orientation.y;
132     z = event->orientation.z;
133   }
134   else if (event->type == SENSOR_TYPE_MAGNETIC_FIELD) {
135     // Serial.print("Mag:");
136     x = event->magnetic.x;
137     y = event->magnetic.y;
138     z = event->magnetic.z;
```

```
  }
  else if (event->type == SENSOR_TYPE_GYROSCOPE) {
    // Serial.print("Gyro:");
    x = event->gyro.x;
    y = event->gyro.y;
    z = event->gyro.z;
  }
  else if (event->type == SENSOR_TYPE_ROTATION_VECTOR) {
    Serial.print("Rot:");
    x = event->gyro.x;
    y = event->gyro.y;
    z = event->gyro.z;
  }
  else if (event->type == SENSOR_TYPE_LINEAR_ACCELERATION) {
    Serial.print("Linear:");
    x = event->acceleration.x;
    y = event->acceleration.y;
    z = event->acceleration.z;
  }
  else if (event->type == SENSOR_TYPE_GRAVITY) {
    Serial.print("Gravity:");
    x = event->acceleration.x;
    y = event->acceleration.y;
    z = event->acceleration.z;
  }
  else {
    Serial.print("Unk:");
  }

  tmp[0]=x;
  tmp[1]=y;
  tmp[2]=z;
}
```