**Madgwich4のプログラム**

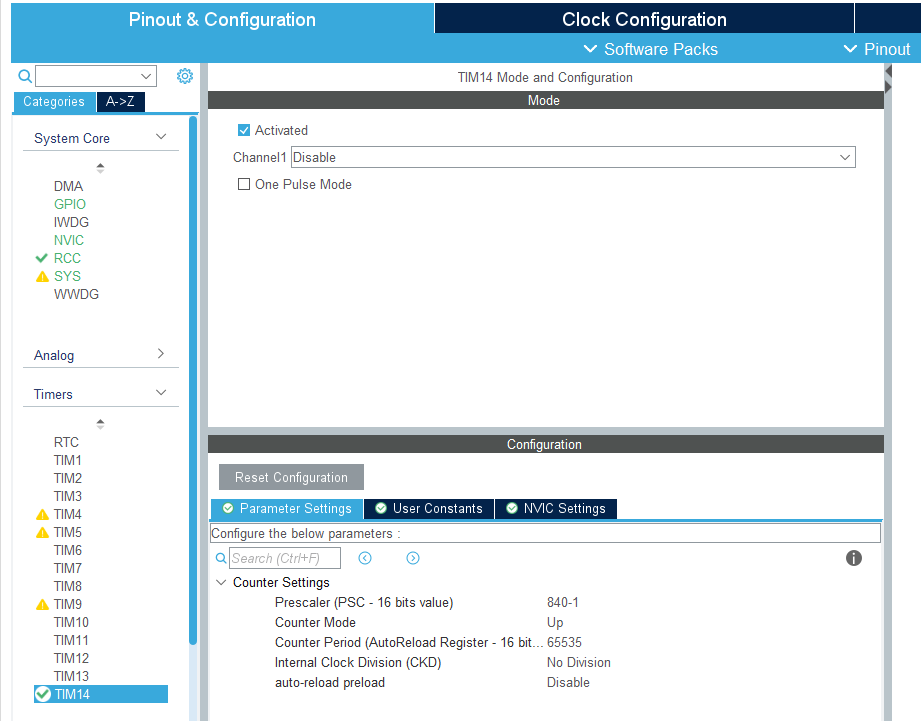
**2022.06.12 M. shimojo**

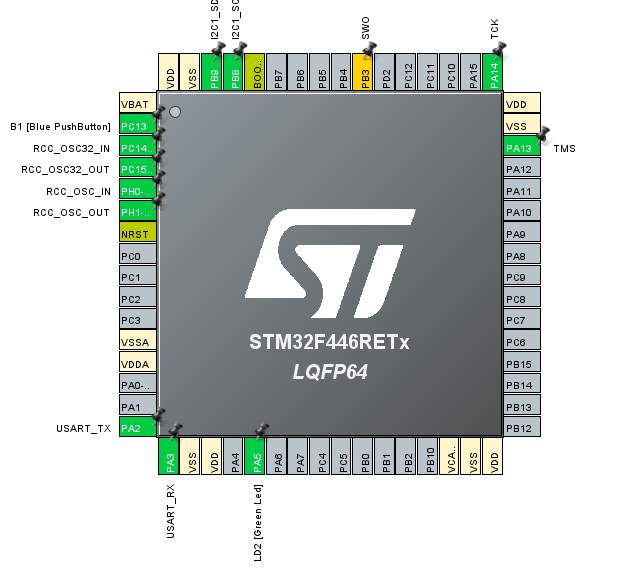
**MPU6050による姿勢の検出を行う。**

* **Madgwick Filterへのサンプリング周波数の受渡（Madgwick.cの一部改変含む）**
* **サンプリング周期の計測を入れた**
* **青ボタンによる割込み処理でGyroのオフセットの補正**
* **描画ソフトprocessingへのデータ出力**

**ファイルの場所：D:\shimo\Documents\STM32workspace\IMU2\Madgwick4**

■STM32 NUCLEO F446REで動作、開発はSTM32 CUBE IDEを利用

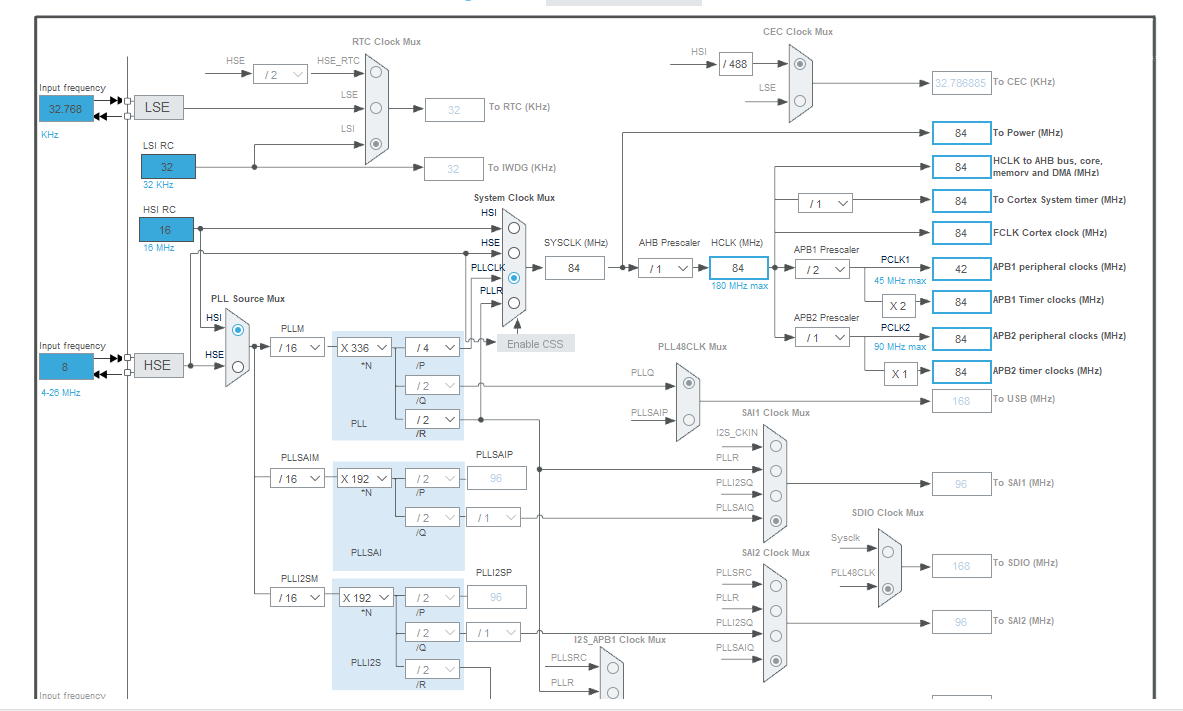
■.ioc



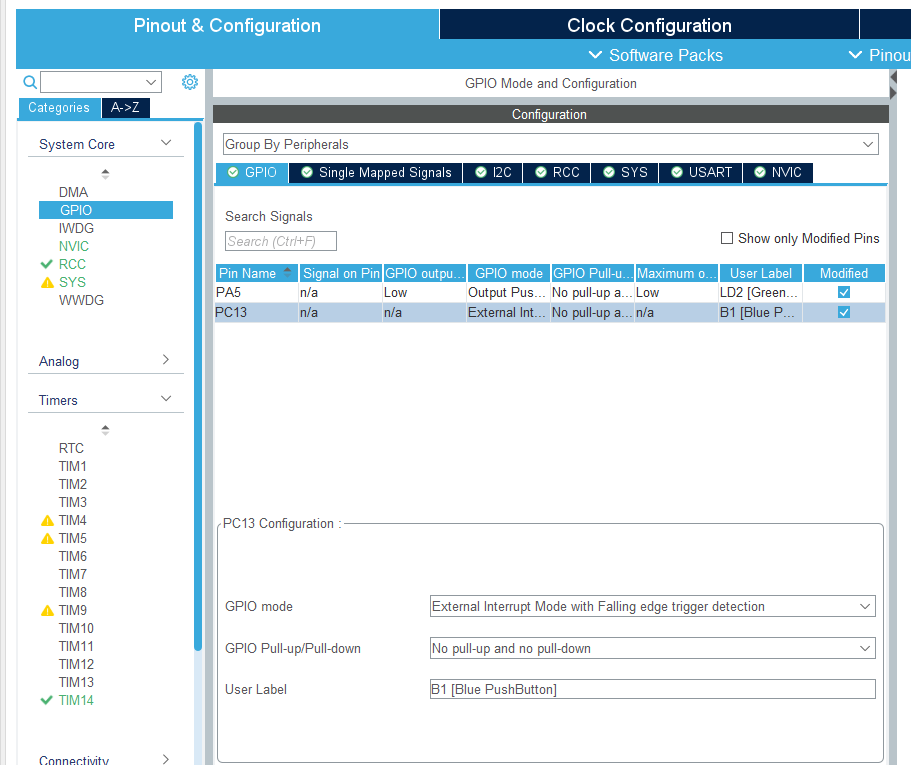
timer clockを1/840

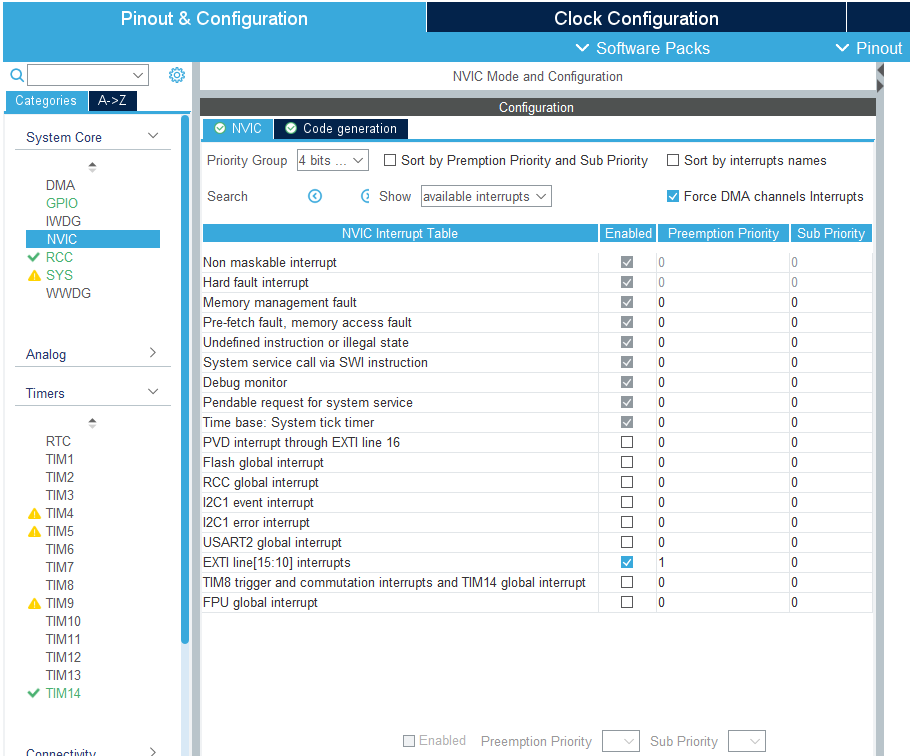
84MHz→1 MHz

1μs単位とする

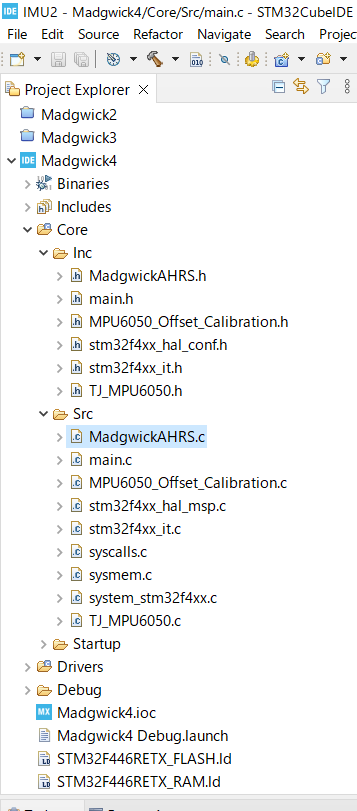


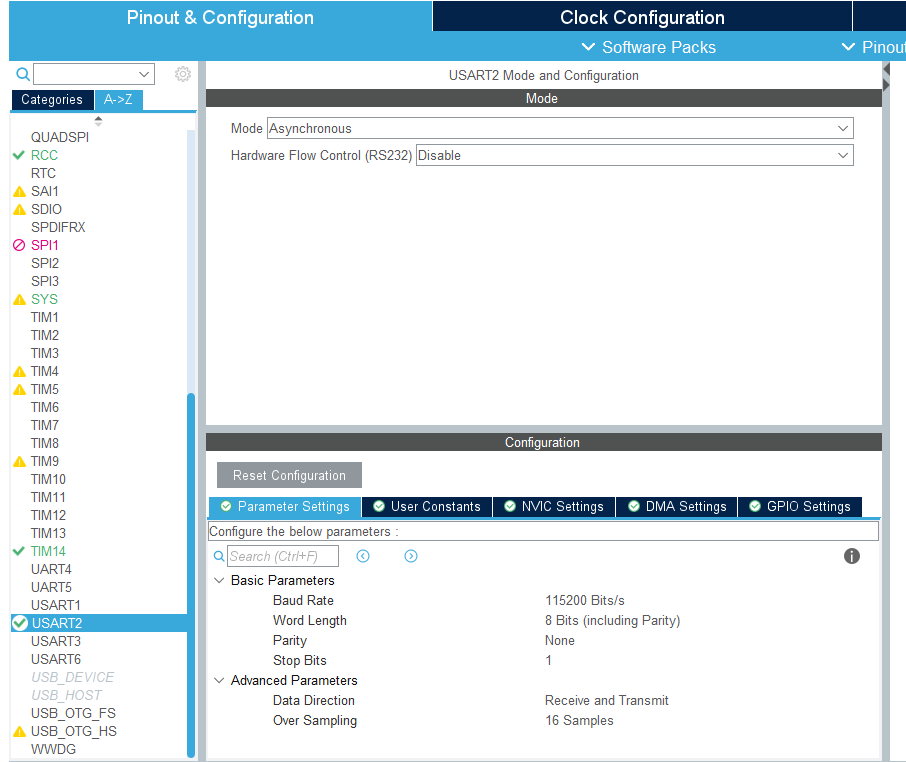
これの1/840





BlueButton割込み





プログラムの挿入方法。(**main.c**)

1. main.cを貼り付け(paste)で行うと、**コンパイラエラーとなる。**
2. このため、USER CODE BEGIN １，２，３，４，５で個別にプログラムコードをpasteしていく。

プログラムの挿入方法。(**main.c以外**)

1. (win10)exploreでプログラムが存在するフォルダを開き、MadgwickAHRS.hを右クリック→copy
2. (CubeIDE)ProjectExploreのフォルダ(Core->Inc)で、右クリック→paste

■プログラムを作る

1. **MPU6050\_offset\_Calibration.h**
2. **MPU6050\_offset\_Calibration.c**
3. **Madgwick.h 一部改変（①のため）**
4. **Madgwick.c 一部改変（①のため）**
5. **main.c**

maic.cからCall

ポインタ引数

1. MPU6050\_offset\_Calibration.h

// header file

void MPU6050\_Offset\_Calibration(int \*Ax\_bias, int \*Ay\_bias, int \*Az\_bias, int \*Gx\_bias, int \*Gy\_bias, int \*Gz\_bias );

//

1. MPU6050\_offset\_Calibration.c

/\*

\* MPU6050 offset calibration. shimojo

\* 拙いプログラムのため、ただの参考にしてください。間違いが多くあります。

\*/

printfを使う時に入れた

//Header files

#include <stdio.h>

#include "MPU6050\_Offset\_Calibration.h"

#include "TJ\_MPU6050.h"

10回程度計測しその平均をとる

#define Cal\_num  10 //オフセットを校正するための計測回数

//============================================================

// Functions

ポインタを使い値の入出力を行う

//------------------------------------------------------------

// Offset Caliburation

void MPU6050\_Offset\_Calibration(int \*Ax\_bias, int \*Ay\_bias, int \*Az\_bias, int \*Gx\_bias, int \*Gy\_bias, int \*Gz\_bias )

{

      RawData\_Def myAccelRaw, myGyroRaw; // add

必要、include "TJ\_MPU6050.h"も必要

これに気付くのに時間がかかった！

      //ScaledData\_Def myAccelScaled, myGyroScaled; // add

    int Ax,Ay,Az;//Raw data

    int Gx,Gy,Gz;//Raw data

    int Ax\_av, Ay\_av, Az\_av=0;

ローカル変数とした

大意無し

    int Gx\_av, Gy\_av, Gz\_av=0;

    int i=0;

    Ax\_av=0; Ay\_av=0; Az\_av=0;

    Gx\_av=0; Gy\_av=0; Gz\_av=0;

データ構造を構築する情報が必要ですよね

// Get initial measure data

    while (i< Cal\_num )

    {

        MPU6050\_Get\_Accel\_RawData(&myAccelRaw);//この命令で加速度とジャイロを取得

        MPU6050\_Get\_Gyro\_RawData(&myGyroRaw);//上の命令で取得したデータを渡すだけ

        Ax=myAccelRaw.x-\*Ax\_bias;  Ay=myAccelRaw.y-\*Ay\_bias; Az=myAccelRaw.z-\*Az\_bias;

        Gx=myGyroRaw.x-\*Gx\_bias;  Gy=myGyroRaw.y-\*Gy\_bias; Gz=myGyroRaw.z-\*Gz\_bias;

       Ax\_av+=Ax; Ay\_av+=Ay; Az\_av+=Az;

平均をとる

       Gx\_av+=Gx; Gy\_av+=Gy; Gz\_av+=Gz;

       i+=1;

       HAL\_Delay(10); //delay 10ms

    }

    Ax\_av=(float)Ax\_av/Cal\_num; Ay\_av=(float)Ay\_av/Cal\_num;Az\_av=(float)Az\_av/Cal\_num;

    Gx\_av=(float)Gx\_av/Cal\_num; Gy\_av=(float)Gy\_av/Cal\_num;Gz\_av=(float)Gz\_av/Cal\_num;

    // calculate average of offset value & Correct for GyroData only.

   // \*Ax\_bias+=(float)Ax\_av; \*Ay\_bias+=(float)Ay\_av;\*Az\_bias+=(float)Az\_av;

ジャイロ角速度のみ戻す

    \*Gx\_bias+=(float)Gx\_av; \*Gy\_bias+=(float)Gy\_av;\*Gz\_bias+=(float)Gz\_av;

}

なぜ、ジャイロ角速度のみオフセットの校正値を戻すのか？

1. MPU6050で姿勢検出を行ったところ、時間経過に従って姿勢がずれていく。例えば、z軸を上面としたとき、z軸周りの姿勢回転が増加する現象が観測された。これはz軸周りの角速度のオフセットがあると、z軸周りの回転があることになり、その積分としてz軸周りの回転が発生することになる。このため、ジャイロからの角速度出力オフセットは出来る限りゼロとすべきである。
2. 翻って、加速度成分の姿勢変動の寄与は少ない。加速度成分は、重力が方向を検出することで姿勢を検出する。この場合、軸方向成分では重力場方向の値が大きい。また姿勢推定アルゴリズムでは、加速度のX,Y,Zの相対的関係より姿勢を検出（正規化して利用する）しているため、温度などの一様なドリフトの影響は少なくなる。
3. Blue-Buttonを押し、Flagを割込みにより設定して、MPU6050\_offset\_Calibrationを行うようにしている。これはあくまでもジャイロ角速度のオフセット値の変更であり、ボタンを押した姿勢をオリジナルの姿勢とする分けではない。
4. Madgwick.h 一部改変
5. //extern volatile float sampling\_Freqency;      // sampling frequency shimojo
6. //---------------------------------------------------------------------------------------------------

ここで関数を宣言した。コンパイラに予め指示する必要がある

1. // Function declarations
2. void MadgwickSetSampling(float samFreq);  //shimojo
3. void MadgwickAHRSupdate(float gx, float gy, float gz, float ax, float ay, float az, float mx, float my, float mz);
4. void MadgwickAHRSupdateIMU(float gx, float gy, float gz, float ax, float ay, float az);

（４）　Madgwick.c 一部改変

// Variable definitions

volatile float beta = betaDef;                              // 2 \* proportional gain (Kp)

volatile float q0 = 1.0f, q1 = 0.0f, q2 = 0.0f, q3 = 0.0f;  // quaternion of sensor frame relative to auxiliary frame

必要か不明

volatile float sampleFreq = 100.0f;     // sampling frequency shimojo　？？？

//------------------------------------------------------------------------------

1. // AHRS algorithm update

新たに付け足した関数

不要；コンパイルオプション

1. #if 1
2. //The sampling frequency is set in the main program.(by shimojo)
3. void MadgwickSetSampling(float samFreq){
4. //float sampleFreq;

mainからの引数（値のみ引き渡す）

1. sampleFreq=samFreq;
2. }
3. //shimojo

不要；コンパイルオプション

1. #endif

Madgwick Filterではサンプリング周期に重要な意味がある。

その逆数のΔtがフィルタアルゴリズムで利用される。Δtは積分として扱われるので、その値の大きさが、結果の大きさに反映される。すなわちΔtが2倍になれは、フィルタ出力（quaternion）も２倍になる。

// Fast inverse square-root

// See: http://en.wikipedia.org/wiki/Fast\_inverse\_square\_root

float invSqrt(float x) {

    float halfx = 0.5f \* x;

    float y = x;

    long i = \*(long\*)&y;

    i = 0x5f3759df - (i>>1);

    y = \*(float\*)&i;

    y = y \* (1.5f - (halfx \* y \* y));//1st iteration shimojo

    y = y \* (1.5f - (halfx \* y \* y));//2nd iteration shimojo

    return y;

}

inverse square rootの近似式。2nd iterationまで記入した。これは無視可能とのこと（実行しなくてもよい）

**（５）main.c**

①CODE BEGIN Includes

/\* USER CODE BEGIN Includes \*/

#include <stdio.h> //add shimojo

不要

#include <stdint.h> //add shimojo

#include "TJ\_MPU6050.h"

#include "MadgwickAHRS.h"

新たに挿入

#include "MPU6050\_Offset\_Calibration.h"

#include "math.h"

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN 0 \*/

/\* Gloval values \*/

const float PI=3.14159265358979;

extern volatile float beta; // add extern

グローバル変数とした

extern volatile float q0, q1, q2, q3; // add extern

uint8\_t data[5];

float Yaw\_m,Pitch\_m,Roll\_m;

//It is made for sampling period measurement. Its value is observed by DEBUGGER(shimojo)

uint16\_t timer\_val; // timer start value

CPU処理経過時間計測用

uint16\_t timer\_val2; // sampling period

/\* USER CODE END 0 \*/

②USER CODE BEGIN 1

int main(void)

{

  /\* USER CODE BEGIN 1 \*/

  MPU\_ConfigTypeDef myMpuConfig; // ジャイロセンサの特性を決めるパラメータをset

  /\* USER CODE END 1 \*/

③USER CODE BEGIN ２

/\* USER CODE BEGIN 2 \*/

データ構造。今回生データを基に自前で処理

  RawData\_Def myAccelRaw, myGyroRaw; // add

  //ScaledData\_Def myAccelScaled, myGyroScaled; // add

スケーリングは、自前でおこなうので不要

   //printf("Gyro Start Madgwick3 \r\n");

    int Ax,Ay,Az;//Raw data

グローバル変数とした。ただ、MPU6050\_offset\_Calibration.cではローカル変数としてますね。不統一ですね。またexternが無い、上の記述と相違。externは必要/不要？？

    int Gx,Gy,Gz;//Raw data

    float Axf,Ayf,Azf;//Scaled data

    float Gxf,Gyf,Gzf;//Scaled data

    // Offset average

    int Ax\_bias,Ay\_bias,Az\_bias;//bias

キャリブレーションは事前に行った。これは重要

1. 加速度は、x,y,z軸を各々重力場方向と正逆配置して中間点をゼロとした
2. 角速度は、静止状態で測定、オフセットを求めた
3. LowPassフィルタ、感度設定で値が若干変わる

    int Gx\_bias,Gy\_bias,Gz\_bias;//bias

    Ax\_bias=671;Ay\_bias=-150;Az\_bias=1099; // measured data by experiment by shimojo

    Gx\_bias=-193;Gy\_bias=86;Gz\_bias=-50;// measured data by experiment by shimojo

    float AccelMetricScaling;

    //Here the gravity unit. Since acceleration is normalized. (shimojo)

    AccelMetricScaling=(2000.0f/32768.0f);  //2000.0f for AFS\_SEL\_2g

    float GyroMetricScaling;

  // Gyro's Angular velocity uses radians. (shimojo)

    GyroMetricScaling= (250.0f/32768.0f)\*(PI/180.0f);    //250.0f for FS\_SEL\_250

角速度はradian/sとする

   // GyroMetricScaling= (500.0f/32768.0f);    //500.0f for FS\_SEL\_500

サンプリング周期のMadgwickFilterへの通知は本質的に必要。オリジナルではMadgwick.c内の#defineで指定している。これは面倒くさいので、プログラムを下条が改変した。

不要、コンパイラオプション

#if 1

    /\* The sampling frequency is calculated by the following formula

     \* sampling frequency=1000/(5+delay(ms))

     \* exsample 50=1000/(5+15)

     \* CPU processing time=5ms, delay=15ms

     \*/

    MadgwickSetSampling(66.7f); // Setsampling Frequency (Added by shimojo)

サンプリング周期をMagdwickへ渡す

#endif

    //timer is created for sampling period measurement. Its value is observed by DEBUGGER(shimojo)

    // Start timer

     HAL\_TIM\_Base\_Start(&htim14); //

CPU処理計測用タイマをenable

    //1. initialize the MPU6050 module and i2c

      MPU6050\_Init(&hi2c1);

    //2.  configure accel and gyro parameters

MPU6050の各種設定を行う。ローパスフィルタは、ノイズと遅延時間のトレードオフ。感度も同様。

      myMpuConfig.Accel\_Full\_Scale=AFS\_SEL\_2g  ;

      myMpuConfig.ClockSource = Internal\_8MHz;

      //myMpuConfig.CONFIG\_DLPF =  DLPF\_21A\_20G\_Hz;// Delay 8.5ms

      myMpuConfig.CONFIG\_DLPF = DLPF\_184A\_188G\_Hz;// Delay 2.0ms

     // myMpuConfig.CONFIG\_DLPF = DLPF\_260A\_256G\_Hz;//

      myMpuConfig.Gyro\_Full\_Scale =FS\_SEL\_250;

      //myMpuConfig.Gyro\_Full\_Scale = FS\_SEL\_500;

      myMpuConfig.Sleep\_Mode\_Bit = 0;  //1: sleep mode, 0: normal mode

オフセット値は事前に計測した。特に加速度センサは、x､y､z各軸に対して重力場を反転させ、計測し、中立点を設定した。あと温度変動は行っていない。必要かも？？

      MPU6050\_Config(&myMpuConfig);

      Ax\_bias=671;Ay\_bias=-150;Az\_bias=1099; // measured data by experiment by shimojo

      Gx\_bias=-193;Gy\_bias=86;Gz\_bias=-50;// measured data by experiment by shimojo

      //Here the gravity unit. Since acceleration is normalized. (shimojo)

      AccelMetricScaling=(2000.0f/32768.0f);  //2000.0f for AFS\_SEL\_2g

    // Gyro's Angular velocity uses radians. (shimojo)

ジャイロの角速度はラジアン/秒とする。

加速度は、正規化し相対的な値として利用するので単位は問わない。

      GyroMetricScaling= (250.0f/32768.0f)\*(PI/180.0f);    //250.0f for FS\_SEL\_500

   // GyroMetricScaling= (500.0f/32768.0f);    //500.0f for FS\_SEL\_500

  // Calculate offset value. Compensate only for Gyro offset data.

    //Because,Attitude drift is due to gyro angular velocity offset. (shimojo)

      MPU6050\_Offset\_Calibration(&Ax\_bias, &Ay\_bias, &Az\_bias,  &Gx\_bias, &Gy\_bias, &Gz\_bias );

“＆”　アドレスを渡す

  /\* USER CODE END 2 \*/

④　USER CODE BEGIN ３

  /\* USER CODE BEGIN WHILE \*/

  while (1)

  {

    /\* USER CODE END WHILE \*/

    /\* USER CODE BEGIN 3 \*/

CPU処理計測　Start!

  //if (Cal\_num==1)   {  printf("%u us\r\n",timer\_val2);Cal\_num=0;  }

   timer\_val = \_\_HAL\_TIM\_GET\_COUNTER(&htim14);//Start timer

    //Get Raw data

        MPU6050\_Get\_Accel\_RawData(&myAccelRaw);//この命令で加速度とジャイロを取得

        MPU6050\_Get\_Gyro\_RawData(&myGyroRaw);//上の命令で取得したデータを渡すだけ

        Ax=myAccelRaw.x-Ax\_bias;  Ay=myAccelRaw.y-Ay\_bias; Az=myAccelRaw.z-Az\_bias;

        Gx=myGyroRaw.x-Gx\_bias;  Gy=myGyroRaw.y-Gy\_bias; Gz=myGyroRaw.z-Gz\_bias;

        Axf=(float)Ax\*(AccelMetricScaling);Ayf=(float)Ay\*(AccelMetricScaling);Azf=(float)Az\*(AccelMetricScaling);// gravity unit

        Gxf=(float)Gx\*(GyroMetricScaling);Gyf=(float)Gy\*(GyroMetricScaling);Gzf=(float)Gz\*(GyroMetricScaling);// radian/s

        //Gxf=0.0f;Gyf=0.0f;Gzf=0.0f;// radian/s

STM32CubeIDEのprintfで%fは不可！

1. プロジェクトを右click, properties,
2. C/C++ Build→Settings→MCU Settings,
3. Use float with printf from newlib-nano (-u \_printf\_float)にcheck

        /\*STM32CubeIDEのprintfでfloatが使えない、対処方法　\*/

        /\*

プロジェクトを右click, properties,

          C/C++ Build→Settings→MCU Settings,

          Use float with printf from newlib-nano (-u \_printf\_float)にcheck

         \*/

Madgwick Filter

        //Madgwick Filter. Sampling frequency must be set in MadgwickAHRS.c (shimojo))

            MadgwickAHRSupdateIMU(Gxf,Gyf,Gzf,Axf,Ayf,Azf);

ジャイロの角速度はラジアン/秒とする

      HAL\_Delay(10); //delay 5ms

サンプリング周期を決める遅延（重要）

遅延0だとMadgwick Filterの値が？？

      //Roll(X-axis),  Pitch(Y-axis), Yaw(Z-axis)

      Roll\_m=(180.0f / PI)\*atan2f(q0\*q1 + q2\*q3, 0.5f - q1\*q1 - q2\*q2);

      Pitch\_m =(180.0f / PI)\*asinf(-2.0f \* (q1\*q3 - q0\*q2));

      Yaw\_m=(180.0f / PI)\*atan2f(q1\*q2 + q0\*q3, 0.5f - q2\*q2 - q3\*q3);

#if 1

      //Drawing graph by Processing

QuaternionからRoll,Pitch,Yawへの変換

特異姿勢が存在するようだ。

Quaternionをそのまま利用する方が良いか？

      printf ("%f",Roll\_m); printf (",");//

    printf ("%f",Pitch\_m); printf (",");//

      printf ("%f",Yaw\_m); printf ("  \r\n");//

       //get time elaasped

       timer\_val2 = \_\_HAL\_TIM\_GET\_COUNTER(&htim14) - timer\_val;

#endif

CPU処理計測　ここまでの経過時間

#if 0

        //MPU6050 measurd data quaternion

        printf ("%f",q0); printf (",");//

Quaternion出力。processingのためのformat

        printf ("%f",q1); printf (",");//

        printf ("%f",q2); printf (",");//

        printf ("%f",q3);  printf ("  \r\n");//

#endif

コンパイラする/しないを指示

#if 0

        //MPU6050 measurd data

測定値出力

        printf ("%f",Axf); printf (",");//

        printf ("%f",Ayf); printf (",");//

        printf ("%f",Azf); printf (",");//

        printf ("%f",Gxf); printf (",");//

        printf ("%f",Gyf); printf (",");//

        printf ("%f",Gzf);  printf ("  \r\n");//

#endif

  }

  /\* USER CODE END 3 \*/

⑤　USER CODE BEGIN ４

/\* USER CODE BEGIN 4 \*/

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin) //blueBottonの割込みルーチン

{

BlueButton用割込み処理

  if (GPIO\_Pin == GPIO\_PIN\_13)

  {

    Pushed = 1;

  }

printf処理

}

// printf用コー�?.ST-LinkよりTeratermに出力す�?,COM4経由かな

int \_write(int file, char \*ptr, int len)

printf用処理.　ST-LinkよりTeratermに出力する

今回の例では、Wnidows10のCOM4経由となっていた。

{

  HAL\_UART\_Transmit(&huart2,(uint8\_t \*)ptr,len,10);

  return len;

}

/\* USER CODE END 4 \*/