```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  *
  * COPYRIGHT(c) 2017 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
#include "usb_device.h"

/* USER CODE BEGIN Includes */
#include "debug.h"
#include "config_drone.h"
#include "timer.h"
#include "rc.h"
#include "steval_fcu001_v1.h"
#include "steval_fcu001_v1_accelero.h"
#include "steval_fcu001_v1_gyro.h"
#include "steval_fcu001_v1_magneto.h"
#include "steval_fcu001_v1_pressure.h"
#include "sensor_data.h"
#include "quaternion.h"
#include "ahrs.h"
```

```
51 #include "flight_control.h"
52 #include "motor.h"
53 /* USER CODE END Includes */
54
55 /* Private variables ---------------------------------------------------------*/
56 ADC_HandleTypeDef hadc1;
57
58 SPI_HandleTypeDef hspi1;
59 SPI_HandleTypeDef hspi2;
60
61 TIM_HandleTypeDef htim2;
62 TIM_HandleTypeDef htim4;
63 TIM_HandleTypeDef htim9;
64
65 UART_HandleTypeDef huart1;
66
67 static void *LSM6DSL_X_0_handle = NULL;
68 static void *LSM6DSL_G_0_handle = NULL;
69 static void *LIS2MDL_M_0_handle = NULL;
70 static void *LPS22HB_P_0_handle = NULL;
71 //static void *LPS22HB_T_0_handle = NULL;       /* To be used in case Temperature reading
   is needed */
72
73 extern Queue_TypeDef que;
74 extern volatile tUserTimer tim;
75 extern char rc_connection_flag;
76 extern int16_t gAIL, gELE, gTHR, gRUD;
77 int32_t rc_cal_flag = 0;
78 int32_t rc_enable_motor = 0;
79 int32_t rc_cal_cnt = 0;
80 int32_t fly_ready = 0;
81 unsigned char ch, ch_flag;
82
83 uint32_t tim9_event_flag = 0, tim9_cnt = 0, tim9_cnt2 = 0;
84 float tmp_euler_z = 0;
85
86
87 /* Private function prototypes -----------------------------------------------*/
88 void SystemClock_Config(void);
89 static void MX_GPIO_Init(void);
90 static void MX_ADC1_Init(void);
91 static void MX_SPI1_Init(void);
92 static void MX_SPI2_Init(void);
93 static void MX_TIM2_Init(void);
94 static void MX_TIM4_Init(void);
95 static void MX_TIM9_Init(void);
96 static void MX_USART1_UART_Init(void);
97
98 static void initializeAllSensors( void );
99 void enableAllSensors( void );
```

```
100
101
102 /* USER CODE BEGIN 0 */
103 P_PI_PIDControlTypeDef pid;
104 EulerAngleTypeDef euler_rc, euler_ahrs, euler_rc_fil, euler_rc_y_pre[4], euler_rc_x_pre[4];
105 AxesRaw_TypeDef acc, gyro, mag, acc_fil_int, gyro_fil_int, mag_fil_int;
106 AxesRaw_TypeDef_Float acc_fil, acc_y_pre[4], acc_x_pre[4], acc_ahrs_FIFO[FIFO_Order],
    acc_FIFO[FIFO_Order], acc_ahrs;
107 AxesRaw_TypeDef_Float gyro_fil, gyro_y_pre[4], gyro_x_pre[4], gyro_ahrs_FIFO[FIFO_Order],
    gyro_FIFO[FIFO_Order], gyro_ahrs;
108 AxesRaw_TypeDef_Float mag_fil;
109 AxesRaw_TypeDef acc_off_calc, gyro_off_calc, acc_offset, gyro_offset;
110 EulerAngleTypeDef euler_ahrs_offset;
111 int sensor_init_cali = 0, sensor_init_cali_count = 0;
112 int gyro_cali_count = 0;
113
114 typedef struct
115 {
116   int16_t X_Degree;
117   int16_t Y_Degree;
118   int16_t Z_Degree;
119 }Attitude_Degree;
120
121 typedef struct
122 {
123   float a1, a2, b0, b1, b2;
124 }IIR_Coeff;
125
126 //sensor filter
127 //7hz, 800hz
128 //IIR_Coeff     gyro_fil_coeff     =     {1.922286512869545,          -0.92519529534950118,
    0.00072719561998898304, 0.0014543912399779661, 0.00072719561998898304};
129
130 //15hz, 800hz
131 //IIR_Coeff     gyro_fil_coeff     =     {1.8337326589246479,          -0.84653197479202391,
    0.003199828966843966, 0.006396579336879321, 0.003199828966843966};
132
133 //30hz, 800hz
134 //IIR_Coeff     gyro_fil_coeff     =     {1.66920314293119312,          -0.71663387350415764,
    0.011857682643241156, 0.023715365286482312, 0.011857682643241156};
135
136 //60hz, 800hz
137 //IIR_Coeff    gyro_fil_coeff    =    {1.3489677452527946    ,          -0.51398189421967566,
    0.041253537241720303, 0.082507074483440607, 0.041253537241720303};
138
139 //100hz, 800hz
140 IIR_Coeff     gyro_fil_coeff     =     {0.94280904158206336,          -0.33333333333333343,
    0.09763107293781749 , 0.19526214587563498 , 0.09763107293781749 };
141
142 Attitude_Degree  Fly, Fly_offset, Fly_origin;
```

```
143 Gyro_Rad gyro_rad, gyro_degree, gyro_cali_degree;
144 MotorControlTypeDef motor_pwm;
145 int count1 = 0, count2 = 0;
146 AHRS_State_TypeDef ahrs;
147 float pre;
148
149 uint32_t VBAT_Sense;
150 float VBAT = 0;
151
152 uint8_t tmp_lis2mdl;
153 SensorAxes_t tmp_mag;
154
155 /* start: add */
156 uint32_t cnt_usr_1 = 0; /* 800Hz */
157 /* end: add */
158
159 /* USER CODE END 0 */
160
161
162  int main(void)
163 {
164
165   /* USER CODE BEGIN 1 */
166   int16_t pid_interval, i;
167
168   int mytimcnt = 0;
169   acc_fil.AXIS_X = 0;
170   acc_fil.AXIS_Y = 0;
171   acc_fil.AXIS_Z = 0;
172   mag_fil.AXIS_X = 0;
173   mag_fil.AXIS_Y = 0;
174   mag_fil.AXIS_Z = 0;
175   gyro_fil.AXIS_X = 0;
176   gyro_fil.AXIS_Y = 0;
177   gyro_fil.AXIS_Z = 0;
178   euler_rc_fil.thx = 0;
179   euler_rc_fil.thy = 0;
180   euler_rc_fil.thz = 0;
181   acc_off_calc.AXIS_X = 0;
182   acc_off_calc.AXIS_Y = 0;
183   acc_off_calc.AXIS_Z = 0;
184   gyro_off_calc.AXIS_X = 0;
185   gyro_off_calc.AXIS_Y = 0;
186   gyro_off_calc.AXIS_Z = 0;
187   acc_offset.AXIS_X = 0;
188   acc_offset.AXIS_Y = 0;
189   acc_offset.AXIS_Z = 1000;
190   gyro_offset.AXIS_X = 0;
191   gyro_offset.AXIS_Y = 0;
192   gyro_offset.AXIS_Z = 0;
```

追加：
800Hz（0.00125s 毎）でカウントアップするカウンタ変数

```
193   euler_rc.thz = euler_ahrs.thz;
194   euler_ahrs_offset.thx = 0;
195   euler_ahrs_offset.thy = 0;
196
197   for(i=0;i<4;i++)
198   {
199     acc_y_pre[i].AXIS_X = 0;
200     acc_y_pre[i].AXIS_Y = 0;
201     acc_y_pre[i].AXIS_Z = 0;
202     acc_x_pre[i].AXIS_X = 0;
203     acc_x_pre[i].AXIS_Y = 0;
204     acc_x_pre[i].AXIS_Z = 0;
205     gyro_y_pre[i].AXIS_X = 0;
206     gyro_y_pre[i].AXIS_Y = 0;
207     gyro_y_pre[i].AXIS_Z = 0;
208     gyro_x_pre[i].AXIS_X = 0;
209     gyro_x_pre[i].AXIS_Y = 0;
210     gyro_x_pre[i].AXIS_Z = 0;
211     euler_rc_y_pre[i].thx = 0;
212     euler_rc_y_pre[i].thy = 0;
213     euler_rc_y_pre[i].thz = 0;
214     euler_rc_x_pre[i].thx = 0;
215     euler_rc_x_pre[i].thy = 0;
216     euler_rc_x_pre[i].thz = 0;
217   }
218
219   /* USER CODE END 1 */
220
221   /* MCU Configuration--------------------------------------------------------*/
222
223   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
224   HAL_Init();
225
226   /* Configure the system clock */
227   SystemClock_Config();
228
229   /* Initialize all configured peripherals */
230   MX_GPIO_Init();
231   MX_ADC1_Init();
232   MX_TIM2_Init();
233   MX_TIM4_Init();
234   MX_TIM9_Init();
235   MX_USART1_UART_Init();
236   //MX_USB_DEVICE_Init();
237
238   /* USER CODE BEGIN 2 */
239
240   PRINTF("STEVAL-FCU001V1 FW rev.1.0 - Sep 2017¥n¥n");
241
242 //  Initialize Onboard LED
```

```
243   BSP_LED_Init(LED1);
244   BSP_LED_Init(LED2);
245   BSP_LED_Off(LED1);
246   BSP_LED_Off(LED2);
247   BSP_LED_On(LED1);
248   BSP_LED_On(LED2);
249
250   /* Configure and disable all the Chip Select pins for sensors on SPI*/
251   Sensor_IO_SPI_CS_Init_All();
252
253   /* Initialize and Enable the available sensors on SPI*/
254   initializeAllSensors();
255   enableAllSensors();
256
257   /* Initialize settings for 6-axis MEMS Accelerometer */
258   /* ODR 6.6kHz */
259   /* FS 4g */
260   /* Analog Filter Bandwith @ 1500Hz */
261   /* ODR/2 low pass filtered sent to composite filter */
262   /* Low pass filter enabled @ ODR/400 */
263   //BSP_ACCELERO_Set_ODR_Value(LSM6DSL_X_0_handle, 1660.0);       /* ODR 1.6kHz */
264   BSP_ACCELERO_Set_ODR_Value(LSM6DSL_X_0_handle, 6660.0);       /* ODR 6.6kHz */
265   BSP_ACCELERO_Set_FS(LSM6DSL_X_0_handle, FS_MID);                    /* FS 4g */
266   //LSM6DSL_ACC_GYRO_W_InComposit(LSM6DSL_X_0_handle, LSM6DSL_ACC_GYRO_IN_ODR_DIV_4);   /*
    ODR/4 low pass filtered sent to composite filter */
267   LSM6DSL_ACC_GYRO_W_InComposit(LSM6DSL_X_0_handle, LSM6DSL_ACC_GYRO_IN_ODR_DIV_2);    /*
    ODR/2 low pass filtered sent to composite filter */
268   LSM6DSL_ACC_GYRO_W_LowPassFiltSel_XL(LSM6DSL_X_0_handle,
    LSM6DSL_ACC_GYRO_LPF2_XL_ENABLE); /* Enable LPF2 filter in composite filter block */
269   //LSM6DSL_ACC_GYRO_W_HPCF_XL(LSM6DSL_X_0_handle, LSM6DSL_ACC_GYRO_HPCF_XL_DIV4); /* Low
    pass filter @ ODR/50 */
270   //LSM6DSL_ACC_GYRO_W_HPCF_XL(LSM6DSL_X_0_handle,  LSM6DSL_ACC_GYRO_HPCF_XL_DIV100);  /*
    Low pass filter @ ODR/100 */
271   LSM6DSL_ACC_GYRO_W_HPCF_XL(LSM6DSL_X_0_handle, LSM6DSL_ACC_GYRO_HPCF_XL_DIV400); /* Low
    pass filter @ ODR/400 */
272   uint8_t tmp_6axis_reg_value;
273   BSP_ACCELERO_Read_Reg(LSM6DSL_X_0_handle, 0x10, &tmp_6axis_reg_value);
274   //tmp_6axis_reg_value = tmp_6axis_reg_value | 0x01;                              /* Set
    LSB to 1 >> Analog filter 400Hz*/
275   tmp_6axis_reg_value = tmp_6axis_reg_value & 0xFE;                           /* Set LSB
    to 0 >> Analog filter 1500Hz*/
276   BSP_ACCELERO_Write_Reg(LSM6DSL_X_0_handle, 0x10, tmp_6axis_reg_value);
277
278   /* Initialize settings for 6-axis MEMS Gyroscope */
279   /* FS 2000dps */
280   /* ODR 416Hz */
281   /* LPF1 FTYPE set to 10b */
282   LSM6DSL_ACC_GYRO_W_LP_BW_G(LSM6DSL_G_0_handle,  LSM6DSL_ACC_GYRO_LP_G_NARROW);  /*  LPF1
    FTYPE set to 10b */
283   BSP_GYRO_Write_Reg(LSM6DSL_G_0_handle, 0x11, 0x6C);                             /*
```

```
    Gyroscope settings: full scale 2000dps, ODR 416Hz */
284
285   /* Initialize settings for Magnetometer settings (By default after reset is in in idle
    mode) */
286   /* Register CFG_REG_A 0x60 = 0x8c */
287   /* Register 0x61 = 0x02 */
288   BSP_MAGNETO_Write_Reg(LIS2MDL_M_0_handle, 0x60, 0x8c);
289   BSP_MAGNETO_Write_Reg(LIS2MDL_M_0_handle, 0x61, 0x02);
290
291   /* Initialize Remote control*/
292   init_remote_control();
293
294   /* Initialize TIM2 for External Remocon RF receiver PWM Input*/
295   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_1);
296   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_2);
297   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_3);
298   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_4);
299
300   /* Initialize TIM4 for Motors PWM Output*/
301   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);
302   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
303   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_3);
304   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_4);
305
306   /* Initialize General purpose TIM9 50Hz*/
307   HAL_TIM_Base_Start_IT(&htim9);
308
309   /* Initialize PID and set Motor PWM to zero */
310   PIDControlInit(&pid);
311   set_motor_pwm_zero(&motor_pwm);
312
313   /* Setup a timer with 5ms interval */
314   pid_interval = PID_SAMPLING_TIME*1000;
315   SetupTimer(&tim, pid_interval);
316
317   /* Start timer */
318   StartTimer(&tim);
319   ch = 0;
320   ch_flag = 0;
321
322   /* USER CODE END 2 */
323
324
325
326   /* Infinite loop */
327   /* USER CODE BEGIN WHILE */
328   while (1)
329   {
330     /* USER CODE END WHILE */
331
```

```
332     /* USER CODE BEGIN 3 */
333
334     if (tim9_event_flag == 1)
335     {     // Timer9 event: frequency 50Hz
336       tim9_event_flag = 0;
337
338       count1++;
339
340       acc_ahrs.AXIS_X = 0;
341       acc_ahrs.AXIS_Y = 0;
342       acc_ahrs.AXIS_Z = 0;
343       gyro_ahrs.AXIS_X = 0;
344       gyro_ahrs.AXIS_Y = 0;
345       gyro_ahrs.AXIS_Z = 0;
346
347       for(i=0;i<FIFO_Order;i++)
348       {
349         acc_ahrs.AXIS_X += acc_ahrs_FIFO[i].AXIS_X;
350         acc_ahrs.AXIS_Y += acc_ahrs_FIFO[i].AXIS_Y;
351         acc_ahrs.AXIS_Z += acc_ahrs_FIFO[i].AXIS_Z;
352         gyro_ahrs.AXIS_X += gyro_ahrs_FIFO[i].AXIS_X;
353         gyro_ahrs.AXIS_Y += gyro_ahrs_FIFO[i].AXIS_Y;
354         gyro_ahrs.AXIS_Z += gyro_ahrs_FIFO[i].AXIS_Z;
355       }
356
357       acc_ahrs.AXIS_X *=FIFO_Order_Recip;
358       acc_ahrs.AXIS_Y *=FIFO_Order_Recip;
359       acc_ahrs.AXIS_Z *=FIFO_Order_Recip;
360       gyro_ahrs.AXIS_X *=FIFO_Order_Recip;
361       gyro_ahrs.AXIS_Y *=FIFO_Order_Recip;
362       gyro_ahrs.AXIS_Z *=FIFO_Order_Recip;
363
364       acc_fil_int.AXIS_X = acc_ahrs.AXIS_X;
365       acc_fil_int.AXIS_Y = acc_ahrs.AXIS_Y;
366       acc_fil_int.AXIS_Z = acc_ahrs.AXIS_Z;
367       gyro_fil_int.AXIS_X = gyro_ahrs.AXIS_X;
368       gyro_fil_int.AXIS_Y = gyro_ahrs.AXIS_Y;
369       gyro_fil_int.AXIS_Z = gyro_ahrs.AXIS_Z;
370
371
372       //PRINTF("%f  %f  %f  %f¥n",  acc_ahrs.AXIS_X,  acc_ahrs.AXIS_Y,  gyro_ahrs.AXIS_X,
    gyro_ahrs.AXIS_Y);
373
374       // AHRS update, quaternion & true gyro data are stored in ahrs
375       ahrs_fusion_ag(&acc_ahrs, &gyro_ahrs, &ahrs);
376
377       // Calculate euler angle drone
378       QuaternionToEuler(&ahrs.q, &euler_ahrs);
379
380       // Get target euler angle from remote control
```

```
381       GetTargetEulerAngle(&euler_rc, &euler_ahrs);
382
383
384       if(gTHR<MIN_THR)
385       {
386         euler_ahrs_offset.thx = 0;
387         euler_ahrs_offset.thy = 0;
388       }
389
390       Fly_origin.X_Degree = (int16_t)(euler_ahrs.thx * 5730);
391       Fly_origin.Y_Degree = (int16_t)(euler_ahrs.thy * 5730);
392       Fly_origin.Z_Degree = (int16_t)(euler_ahrs.thz * 5730);
393
394
395       if(gTHR<MIN_THR)
396       {
397         euler_rc.thz = 0;
398         euler_ahrs.thz = 0;
399       }
400
401       euler_rc_fil.thx = euler_rc.thx;
402       euler_rc_fil.thy = euler_rc.thy;
403       euler_rc_fil.thz = euler_rc.thz;
404
405       FlightControlPID_OuterLoop(&euler_rc_fil, &euler_ahrs, &ahrs, &pid);
406
407 //PRINTF("%d  %d  %d  %d  %d  %d¥r¥n", (int)(euler_ahrs.thx * 57.3), (int)(euler_ahrs.thy
    * 57.3), (int)(euler_ahrs.thz * 57.3), (int)(euler_rc.thx * 57.3), (int)(euler_rc.thy *
    57.3), (int)(euler_rc.thz * 57.3));
408 /*if ((count1 % 2) == 0)        // 約80Hz
409 PRINTF("%d %d %d %d %d %d %d %d¥r¥n",
410         (int)(acc_fil_int.AXIS_Z / 10),
411         (int)(gyro_fil_int.AXIS_X   /   1000),   (int)(gyro_fil_int.AXIS_Y   /   1000),
    (int)(gyro_fil_int.AXIS_Z / 1000),
412         gELE / 4, gAIL / 4, gRUD / 4, gTHR / 4);*/
413 /*if ((count1 % 2) == 0)        // 約80Hz ・・・システム同定用（nZ）
414 PRINTF("%d %d %d %d %d %d %d %d %d¥n",
415         (int)(cnt_usr_1 / 10),
416         (int)(acc_fil_int.AXIS_Z / 10), // 分解能10mG。
417         (int)(gyro_fil_int.AXIS_X   /   1000),   (int)(gyro_fil_int.AXIS_Y   /   1000),
    (int)(gyro_fil_int.AXIS_Z / 1000),      // 分解能1deg/s
418         ((int)motor_pwm.motor1_pwm)   /   2,   ((int)motor_pwm.motor2_pwm)   /   2,
    ((int)motor_pwm.motor3_pwm) / 2, ((int)motor_pwm.motor4_pwm) / 2);      // 分解能2LSB
419         */
420 /*if ((count1 % 2) == 0)        // 約80Hz ・・・システム同定用（p, q, r）
421 PRINTF("%d %d %d %d %d %d %d %d¥n",
422         (int)(cnt_usr_1 / 10),
423         (int)(acc_fil_int.AXIS_Z / 10), // 分解能10mG。
424         (int)(gyro_fil_int.AXIS_X   /   1000),   (int)(gyro_fil_int.AXIS_Y   /   1000),
    (int)(gyro_fil_int.AXIS_Z / 1000),      // 分解能1deg/s
```

追加：UART 送信…XBeeによるフライトデータのダウンリンク（～435 行目）

【上下加速度のシステム同定時】にコメントアウトを解除して使う

【角速度モデルのシステム同定時】にコメントアウトを解除して使う（～426 行目）

```
425         (int)(pid.x_s1 * 57.29578F), (int)(pid.y_s1 * 57.29578F), (int)(pid.z_s1 *
    57.29578F));    // 分解能 1deg/s
426         */
427 if ((count1 % 8) == 0)  // 約 20Hz ・・・制御モニタ用
428 PRINTF("%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d¥n",
429         (int)(cnt_usr_1 / 10),
430         (int)(acc_fil_int.AXIS_Z / 10), // 分解能 10mG。
431         (int)(gyro_fil_int.AXIS_X / 1000), (int)(gyro_fil_int.AXIS_Y / 1000),
    (int)(gyro_fil_int.AXIS_Z / 1000),      // 分解能 1deg/s
432         (int)(pid.x_s1 * 57.29578F), (int)(pid.y_s1 * 57.29578F), (int)(pid.z_s1 *
    57.29578F),     // 分解能 1deg/s
433         (int)(euler_ahrs.thx * 57.29578F * 5.F), (int)(euler_ahrs.thy * 57.29578F * 5.F),
    (int)(euler_ahrs.thz * 57.29578F * 5.F),        // 分解能 0.2deg
434         (int)(euler_rc.thx * 57.29578F * 5.F), (int)(euler_rc.thy * 57.29578F * 5.F),
    (int)(euler_rc.thz * 57.29578F * 5.F),  // 分解能 0.2deg
435         ((int)motor_pwm.motor1_pwm) / 2, ((int)motor_pwm.motor2_pwm) / 2,
    ((int)motor_pwm.motor3_pwm) / 2, ((int)motor_pwm.motor4_pwm) / 2);      // 分解能 2LSB
436     }
437
438   if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == GPIO_PIN_SET)
439   {
440     ch_flag = 1;
441   }
442
443   if (isTimerEventExist(&tim))    // Check if a timer event is present
444   {
445
446         ClearTimer(&tim);           // Clear current event;
447
448         count2++;
449
450         mytimcnt++;
451         if (rc_connection_flag && rc_enable_motor)
452         {
453           if (mytimcnt%50 == 0)
454           BSP_LED_On(LED2);
455         }
456         else
457         {
458           if (mytimcnt%50 == 0)
459             BSP_LED_Toggle(LED2);
460         }
461     }
462
463     /* Added for debug on UART*/
464     /* Remocon ELE, AIL, RUD, THR [LSB](4LSB/us) */
465 //    PRINTF("%4d  %4d  %4d  %4d¥r¥n", gELE / 4, gAIL / 4, gRUD / 4, gTHR / 4);
466     /* AHRS Euler angle x, y, z axis, Remocon Euler angle x, y, z axis */
467 //        PRINTF("%d    %d    %d    %d    %d    %d¥r¥n", (int)(euler_ahrs.thx * 57.3),
    (int)(euler_ahrs.thy * 57.3), (int)(euler_ahrs.thz * 57.3), (int)(euler_rc.thx * 57.3),
```

制御実験時のモニタリング用に使う．システム同定時はコメントアウトする．

```
(int)(euler_rc.thy * 57.3), (int)(euler_rc.thz * 57.3));
468
469     /* Remocon ELE, AIL, RUD, THR, Motor1_pwm, AHRS Euler angle x and y axis */
470 //          PRINTF("%d¥t%d¥t%d¥t%d¥t%f¥t%f¥t%f¥t%f¥t%f¥n",  gELE,  gAIL,  gRUD,  gTHR,
motor_pwm.motor1_pwm, euler_ahrs.thx * 57.3, euler_ahrs.thy * 57.3, euler_rc.thx * 57.3,
euler_rc.thy * 57.3);
471
472     /* Remocon THR, Acc and Gyro FIFO data x and y axis, AHRS Euler angle x and y axis,
Remocon Euler angle x and y axis*/
473     //PRINTF("%d¥t%f¥t%f¥t%f¥t%f¥t%f¥t%f¥t%f¥t%f¥n",         gTHR,         acc_ahrs.AXIS_X,
acc_ahrs.AXIS_Y, gyro_ahrs.AXIS_X, gyro_ahrs.AXIS_Y, euler_ahrs.thx * 57.3, euler_ahrs.thy
* 57.3, euler_rc.thx * 57.3, euler_rc.thy * 57.3);
474     /* MEMS Accelerometer RAW data */
475     //PRINTF("%d¥t%d¥t%d¥t¥n", acc.AXIS_X, acc.AXIS_Y, acc.AXIS_Z);
476
477 //     /* Pressure data on UART for debug*/
478 //     PRINTF("Pressure [atm] = %f¥n¥n",pre);
479 //     /* Magnetometer data on UART for debug*/
480 //     PRINTF("Magnetometer X = %d¥tY = %d¥tZ = %d¥n¥n", mag.AXIS_X, mag.AXIS_Y, mag.AXIS_Z);
481 //
482 //     /* Added reading of Battery voltage and data on UART */
483 //     //VBAT_Sense = HAL_ADC_GetValue(&hadc1);
484 //     HAL_ADC_Start(&hadc1);
485 //         if (HAL_ADC_PollForConversion(&hadc1, 1000000) == HAL_OK)
486 //         {
487 //             VBAT_Sense = HAL_ADC_GetValue(&hadc1);
488 //             VBAT = (((VBAT_Sense*3.3)/4095)*(BAT_RUP+BAT_RDW))/BAT_RDW;
489 //             PRINTF("Battery voltage = %fV¥n¥n", VBAT);
490 //         }
491 //     HAL_ADC_Stop(&hadc1);
492
493   }
494   /* USER CODE END 3 */
495
496 }
497
498 /** System Clock Configuration
499 */
500 void SystemClock_Config(void)
501 {
502   RCC_OscInitTypeDef RCC_OscInitStruct;
503   RCC_ClkInitTypeDef RCC_ClkInitStruct;
504
505     /**Configure the main internal regulator output voltage
506     */
507   __HAL_RCC_PWR_CLK_ENABLE();
508
509   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
510
511     /**Initializes the CPU, AHB and APB busses clocks
```

変更：コメントアウト

```
512     */
513   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
514   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
515   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
516   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
517   RCC_OscInitStruct.PLL.PLLM = 16;
518   RCC_OscInitStruct.PLL.PLLN = 336;
519   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
520   RCC_OscInitStruct.PLL.PLLQ = 7;
521
522   HAL_RCC_OscConfig(&RCC_OscInitStruct);
523
524     /**Initializes the CPU, AHB and APB busses clocks
525     */
526   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
527                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
528   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
529   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
530   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
531   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
532   HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2);
533
534     /**Configure the Systick interrupt time
535     */
536   HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
537
538     /**Configure the Systick
539     */
540   HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
541
542   /* SysTick_IRQn interrupt configuration */
543   HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
544
545
546 }
547
548 /* ADC1 init function */
549 void MX_ADC1_Init(void)
550 {
551
552   ADC_ChannelConfTypeDef sConfig;
553
554     /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and
    number of conversion)
555     */
556   hadc1.Instance = ADC1;
557   hadc1.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV4;
558   hadc1.Init.Resolution = ADC_RESOLUTION12b;
559   hadc1.Init.ScanConvMode = DISABLE;
560   hadc1.Init.ContinuousConvMode = DISABLE;
```

```
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = EOC_SINGLE_CONV;
  HAL_ADC_Init(&hadc1);

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_9;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

}

/* SPI1 init function */
void MX_SPI1_Init(void)
{

  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLED;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLED;
  hspi1.Init.CRCPolynomial = 10;
  HAL_SPI_Init(&hspi1);

}

/* SPI2 init function */
void MX_SPI2_Init(void)
{

  hspi2.Instance = SPI2;
  hspi2.Init.Mode = SPI_MODE_MASTER;
  hspi2.Init.Direction = SPI_DIRECTION_2LINES;
  hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi2.Init.NSS = SPI_NSS_SOFT;
  hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
```

```
  610   hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  611   hspi2.Init.TIMode = SPI_TIMODE_DISABLED;
  612   hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLED;
  613   hspi2.Init.CRCPolynomial = 10;
  614   HAL_SPI_Init(&hspi2);
  615
  616 }
  617
  618 /* TIM2 init function */
  619 void MX_TIM2_Init(void)
  620 {
  621
  622   TIM_ClockConfigTypeDef sClockSourceConfig;
  623   TIM_MasterConfigTypeDef sMasterConfig;
  624   TIM_IC_InitTypeDef sConfigIC;
  625
  626   htim2.Instance = TIM2;
  627   htim2.Init.Prescaler = 20;
  628   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  629   htim2.Init.Period = 32767;
  630   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  631   HAL_TIM_Base_Init(&htim2);
  632
  633   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  634   HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig);
  635
  636   HAL_TIM_IC_Init(&htim2);
  637
  638   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  639   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  640   HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig);
  641
  642   sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
  643   sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  644   sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  645   sConfigIC.ICFilter = 0;
  646   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1);
  647
  648   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_2);
  649
  650   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_3);
  651
  652   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_4);
  653
  654 }
  655
  656 /* TIM4 init function */
  657 void MX_TIM4_Init(void)
  658 {
  659
```

```
660   TIM_ClockConfigTypeDef sClockSourceConfig;
661   TIM_MasterConfigTypeDef sMasterConfig;
662   TIM_OC_InitTypeDef sConfigOC;
663
664   htim4.Instance = TIM4;
665   #ifdef MOTOR_DC
666     htim4.Init.Prescaler = 84;                                          /* DC motor
    configuration - Freq 494Hz*/
667     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
668     htim4.Init.Period = 1999;
669   #endif
670   #ifdef MOTOR_ESC
671     htim4.Init.Prescaler = 100;                                         /* ESC motor
    configuration - Freq 400Hz*/
672     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
673     htim4.Init.Period = 2075;
674   #endif
675
676   htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
677   HAL_TIM_Base_Init(&htim4);
678
679   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
680   HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig);
681
682   HAL_TIM_PWM_Init(&htim4);
683
684   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
685   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
686   HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig);
687
688   sConfigOC.OCMode = TIM_OCMODE_PWM1;
689   sConfigOC.Pulse = 0;
690   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
691   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
692   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1);
693
694   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2);
695
696   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3);
697
698   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_4);
699
700 }
701
702 /* TIM9 init function */
703 void MX_TIM9_Init(void)
704 {
705
706   TIM_ClockConfigTypeDef sClockSourceConfig;
707
```

```
  htim9.Instance = TIM9;
  htim9.Init.Prescaler = 51;
  htim9.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim9.Init.Period = 1999;
  htim9.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  HAL_TIM_Base_Init(&htim9);

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  HAL_TIM_ConfigClockSource(&htim9, &sClockSourceConfig);

}

/* USART1 init function */
void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  HAL_UART_Init(&huart1);

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
        * Free pins are configured automatically as Analog (this feature is enabled through
        * the Code Generation settings)
*/
void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __GPIOC_CLK_ENABLE();
  __GPIOA_CLK_ENABLE();
  __GPIOB_CLK_ENABLE();

  /*Configure GPIO pins : PB4 PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
```

```
758   GPIO_InitStruct.Pull = GPIO_NOPULL;
759   GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
760   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
761
762 }
763
764 /* USER CODE BEGIN 4 */
765 /*
766  *  Handle Timer9 interrupt @ 800Hz
767  *  Set the event flag and increase time index
768  */
769 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
770 {
771   if(sensor_init_cali == 0)
772   {
773     sensor_init_cali_count++;
774
775     if(sensor_init_cali_count > 800)
776     {
777       // Read sensor data and prepare for specific coodinate system
778       ReadSensorRawData(LSM6DSL_X_0_handle,     LSM6DSL_G_0_handle,     LIS2MDL_M_0_handle,
    LPS22HB_P_0_handle, &acc, &gyro, &mag, &pre);
779
780       acc_off_calc.AXIS_X += acc.AXIS_X;
781       acc_off_calc.AXIS_Y += acc.AXIS_Y;
782       acc_off_calc.AXIS_Z += acc.AXIS_Z;
783
784       gyro_off_calc.AXIS_X += gyro.AXIS_X;
785       gyro_off_calc.AXIS_Y += gyro.AXIS_Y;
786       gyro_off_calc.AXIS_Z += gyro.AXIS_Z;
787
788       if (sensor_init_cali_count >= 1600)
789       {
790         acc_offset.AXIS_X = acc_off_calc.AXIS_X * 0.00125;
791         acc_offset.AXIS_Y = acc_off_calc.AXIS_Y * 0.00125;
792         acc_offset.AXIS_Z = acc_off_calc.AXIS_Z * 0.00125;
793
794         gyro_offset.AXIS_X = gyro_off_calc.AXIS_X * 0.00125;
795         gyro_offset.AXIS_Y = gyro_off_calc.AXIS_Y * 0.00125;
796         gyro_offset.AXIS_Z = gyro_off_calc.AXIS_Z * 0.00125;
797
798         acc_off_calc.AXIS_X = 0;
799         acc_off_calc.AXIS_Y = 0;
800         acc_off_calc.AXIS_Z = 0;
801         gyro_off_calc.AXIS_X = 0;
802         gyro_off_calc.AXIS_Y = 0;
803         gyro_off_calc.AXIS_Z = 0;
804
805         sensor_init_cali_count = 0;
806         sensor_init_cali = 1;
```

```
807       }
808     }
809   }
810 
811   if(sensor_init_cali == 1)
812   {
813     tim9_cnt++;
814     tim9_cnt2++;
815 
816     // Read sensor data and prepare for specific coodinate system
817     ReadSensorRawData(LSM6DSL_X_0_handle,      LSM6DSL_G_0_handle,      LIS2MDL_M_0_handle,
    LPS22HB_P_0_handle, &acc, &gyro, &mag, &pre);
818 
819     if (rc_cal_flag == 1)
820     {
821       acc_off_calc.AXIS_X += acc.AXIS_X;
822       acc_off_calc.AXIS_Y += acc.AXIS_Y;
823       acc_off_calc.AXIS_Z += acc.AXIS_Z;
824 
825       gyro_off_calc.AXIS_X += gyro.AXIS_X;
826       gyro_off_calc.AXIS_Y += gyro.AXIS_Y;
827       gyro_off_calc.AXIS_Z += gyro.AXIS_Z;
828 
829       rc_cal_cnt++;
830 
831       if (rc_cal_cnt >= 800)
832       {
833         acc_offset.AXIS_X = acc_off_calc.AXIS_X * 0.00125;
834         acc_offset.AXIS_Y = acc_off_calc.AXIS_Y * 0.00125;
835         acc_offset.AXIS_Z = acc_off_calc.AXIS_Z * 0.00125;
836 
837         gyro_offset.AXIS_X = gyro_off_calc.AXIS_X * 0.00125;
838         gyro_offset.AXIS_Y = gyro_off_calc.AXIS_Y * 0.00125;
839         gyro_offset.AXIS_Z = gyro_off_calc.AXIS_Z * 0.00125;
840 
841         acc_off_calc.AXIS_X = 0;
842         acc_off_calc.AXIS_Y = 0;
843         acc_off_calc.AXIS_Z = 0;
844         gyro_off_calc.AXIS_X = 0;
845         gyro_off_calc.AXIS_Y = 0;
846         gyro_off_calc.AXIS_Z = 0;
847 
848         rc_cal_cnt = 0;
849         rc_cal_flag = 0;
850       }
851     }
852 
853     acc.AXIS_X -= acc_offset.AXIS_X;
854     acc.AXIS_Y -= acc_offset.AXIS_Y;
855     acc.AXIS_Z -= (acc_offset.AXIS_Z - 1000);
```

```
856     gyro.AXIS_X -= gyro_offset.AXIS_X;
857     gyro.AXIS_Y -= gyro_offset.AXIS_Y;
858     gyro.AXIS_Z -= gyro_offset.AXIS_Z;
859
860     // Save filtered data to acc_FIFO
861     acc_FIFO[tim9_cnt2-1].AXIS_X = acc.AXIS_X;
862     acc_FIFO[tim9_cnt2-1].AXIS_Y = acc.AXIS_Y;
863     acc_FIFO[tim9_cnt2-1].AXIS_Z = acc.AXIS_Z;
864
865     // IIR Filtering on gyro
866     gyro_fil.AXIS_X             =             gyro_fil_coeff.b0*gyro.AXIS_X             +
    gyro_fil_coeff.b1*gyro_x_pre[0].AXIS_X + gyro_fil_coeff.b2*gyro_x_pre[1].AXIS_X
867                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_X + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_X;
868     gyro_fil.AXIS_Y             =             gyro_fil_coeff.b0*gyro.AXIS_Y             +
    gyro_fil_coeff.b1*gyro_x_pre[0].AXIS_Y + gyro_fil_coeff.b2*gyro_x_pre[1].AXIS_Y
869                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_Y + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_Y;
870     gyro_fil.AXIS_Z             =             gyro_fil_coeff.b0*gyro.AXIS_Z             +
    gyro_fil_coeff.b1*gyro_x_pre[0].AXIS_Z + gyro_fil_coeff.b2*gyro_x_pre[1].AXIS_Z
871                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_Z + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_Z;
872     // Shift IIR filter state
873     for(int i=1;i>0;i--)
874     {
875       gyro_x_pre[i].AXIS_X = gyro_x_pre[i-1].AXIS_X;
876       gyro_x_pre[i].AXIS_Y = gyro_x_pre[i-1].AXIS_Y;
877       gyro_x_pre[i].AXIS_Z = gyro_x_pre[i-1].AXIS_Z;
878       gyro_y_pre[i].AXIS_X = gyro_y_pre[i-1].AXIS_X;
879       gyro_y_pre[i].AXIS_Y = gyro_y_pre[i-1].AXIS_Y;
880       gyro_y_pre[i].AXIS_Z = gyro_y_pre[i-1].AXIS_Z;
881     }
882     gyro_x_pre[0].AXIS_X = gyro.AXIS_X;
883     gyro_x_pre[0].AXIS_Y = gyro.AXIS_Y;
884     gyro_x_pre[0].AXIS_Z = gyro.AXIS_Z;
885     gyro_y_pre[0].AXIS_X = gyro_fil.AXIS_X;
886     gyro_y_pre[0].AXIS_Y = gyro_fil.AXIS_Y;
887     gyro_y_pre[0].AXIS_Z = gyro_fil.AXIS_Z;
888
889     //  Save filtered data to gyro_FIFO
890     gyro_FIFO[tim9_cnt2-1].AXIS_X = gyro_fil.AXIS_X;
891     gyro_FIFO[tim9_cnt2-1].AXIS_Y = gyro_fil.AXIS_Y;
892     gyro_FIFO[tim9_cnt2-1].AXIS_Z = gyro_fil.AXIS_Z;
893
894
895     if(tim9_cnt2 == FIFO_Order)
896     {
897       tim9_cnt2 = 0;
898       tim9_event_flag = 1;
899       for(int i=0;i<FIFO_Order;i++)
```

```
900       {
901         acc_ahrs_FIFO[i].AXIS_X = acc_FIFO[i].AXIS_X;
902         acc_ahrs_FIFO[i].AXIS_Y = acc_FIFO[i].AXIS_Y;
903         acc_ahrs_FIFO[i].AXIS_Z = acc_FIFO[i].AXIS_Z;
904         gyro_ahrs_FIFO[i].AXIS_X = gyro_FIFO[i].AXIS_X;
905         gyro_ahrs_FIFO[i].AXIS_Y = gyro_FIFO[i].AXIS_Y;
906         gyro_ahrs_FIFO[i].AXIS_Z = gyro_FIFO[i].AXIS_Z;
907       }
908     }
909
910
911       gyro_rad.gx = gyro_fil.AXIS_X*COE_MDPS_TO_RADPS;
912       gyro_rad.gy = gyro_fil.AXIS_Y*COE_MDPS_TO_RADPS;
913       gyro_rad.gz = gyro_fil.AXIS_Z*COE_MDPS_TO_RADPS;
914
915       euler_ahrs.thz += gyro_rad.gz*PID_SAMPLING_TIME;
916
917       if(gTHR<MIN_THR)
918       {
919         euler_rc.thz = 0;
920         euler_ahrs.thz = 0;
921       }
922
923
924       if (rc_connection_flag && rc_enable_motor)
925       {   // Do PID Control
926         FlightControlPID_innerLoop(&euler_rc_fil, &gyro_rad, &ahrs, &pid, &motor_pwm);
927       }
928       else
929       {
930         // set motor output zero
931         set_motor_pwm_zero(&motor_pwm);
932       }
933
934       if(gTHR<MIN_THR)
935       {
936         set_motor_pwm_zero(&motor_pwm);
937       }
938
939       set_motor_pwm(&motor_pwm);       /* To comment if want to debug remocon calibration
    switching off the motors */
940   }
941
942   ++cnt_usr_1;  /* 800Hz */
943 }
944
945
946 /**
947 * @brief  Initialize all sensors
948 * @param  None
```

追加：カウンタのインクリメント

```
949 * @retval None
950 */
951 static void initializeAllSensors( void )
952 {
953   if (BSP_ACCELERO_Init( LSM6DSL_X_0, &LSM6DSL_X_0_handle ) != COMPONENT_OK)
954   {
955     while(1);
956   }
957
958   if (BSP_GYRO_Init( LSM6DSL_G_0, &LSM6DSL_G_0_handle ) != COMPONENT_OK)
959   {
960     while(1);
961   }
962
963   if (BSP_MAGNETO_Init( LIS2MDL_M_0, &LIS2MDL_M_0_handle ) != COMPONENT_OK)
964   {
965     while(1);
966   }
967
968
969   if (BSP_PRESSURE_Init( LPS22HB_P_0, &LPS22HB_P_0_handle ) != COMPONENT_OK)
970   {
971     while(1);
972   }
973
974 //  if (BSP_TEMPERATURE_Init( LPS22HB_T_0, &LPS22HB_T_0_handle ) != COMPONENT_OK)
975 //  {
976 //    while(1);
977 //  }
978 //
979
980 }
981
982 /**
983 * @brief  Enable all sensors
984 * @param  None
985 * @retval None
986 */
987 void enableAllSensors( void )
988 {
989   BSP_ACCELERO_Sensor_Enable( LSM6DSL_X_0_handle );
990   PRINTF("LSM6DSL MEMS Accelerometer initialized and enabled¥n");
991   BSP_GYRO_Sensor_Enable( LSM6DSL_G_0_handle );
992   PRINTF("LSM6DSL MEMS Gyroscope initialized and enabled¥n");
993   BSP_MAGNETO_Sensor_Enable( LIS2MDL_M_0_handle );
994   PRINTF("LIS2MDL Magnetometer initialized and enabled¥n");
995   BSP_PRESSURE_Sensor_Enable( LPS22HB_P_0_handle );
996   PRINTF("LPS22HB Pressure sensor initialized and enabled¥n");
997 //  BSP_TEMPERATURE_Sensor_Enable( LPS22HB_T_0_handle );
998 }
```

```
 999
1000
1001
1002/* USER CODE END 4 */
1003
1004#ifdef USE_FULL_ASSERT
1005
1006/**
1007   * @brief Reports the name of the source file and the source line number
1008   * where the assert_param error has occurred.
1009   * @param file: pointer to the source file name
1010   * @param line: assert_param error line source number
1011   * @retval None
1012   */
1013void assert_failed(uint8_t* file, uint32_t line)
1014{
1015  /* USER CODE BEGIN 6 */
1016  /* User can add his own implementation to report the file name and line number,
1017     ex: printf("Wrong parameters value: file %s on line %d¥r¥n", file, line) */
1018  /* USER CODE END 6 */
1019
1020}
1021
1022#endif
1023
1024/**
1025  * @}
1026  */
1027
1028/**
1029  * @}
1030*/
1031
1032/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
1033
```