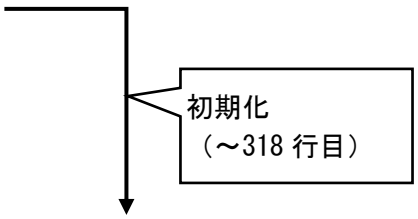
```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  *
  * COPYRIGHT(c) 2017 STMicroelectronics
  *
  * Redistribution 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 /* USER CODE END 0 */
156
157
158  int main(void)
159 {
160
161   /* USER CODE BEGIN 1 */
162   int16_t pid_interval, i;
163
164   int mytimcnt = 0;
165   acc_fil.AXIS_X = 0;
166   acc_fil.AXIS_Y = 0;
167   acc_fil.AXIS_Z = 0;
168   mag_fil.AXIS_X = 0;
169   mag_fil.AXIS_Y = 0;
170   mag_fil.AXIS_Z = 0;
171   gyro_fil.AXIS_X = 0;
172   gyro_fil.AXIS_Y = 0;
173   gyro_fil.AXIS_Z = 0;
174   euler_rc_fil.thx = 0;
175   euler_rc_fil.thy = 0;
176   euler_rc_fil.thz = 0;
177   acc_off_calc.AXIS_X = 0;
178   acc_off_calc.AXIS_Y = 0;
179   acc_off_calc.AXIS_Z = 0;
180   gyro_off_calc.AXIS_X = 0;
181   gyro_off_calc.AXIS_Y = 0;
182   gyro_off_calc.AXIS_Z = 0;
183   acc_offset.AXIS_X = 0;
184   acc_offset.AXIS_Y = 0;
185   acc_offset.AXIS_Z = 1000;
186   gyro_offset.AXIS_X = 0;
187   gyro_offset.AXIS_Y = 0;
188   gyro_offset.AXIS_Z = 0;
189   euler_rc.thz = euler_ahrs.thz;
190   euler_ahrs_offset.thx = 0;
191   euler_ahrs_offset.thy = 0;
192
```



```
193   for(i=0;i<4;i++)
194   {
195     acc_y_pre[i].AXIS_X = 0;
196     acc_y_pre[i].AXIS_Y = 0;
197     acc_y_pre[i].AXIS_Z = 0;
198     acc_x_pre[i].AXIS_X = 0;
199     acc_x_pre[i].AXIS_Y = 0;
200     acc_x_pre[i].AXIS_Z = 0;
201     gyro_y_pre[i].AXIS_X = 0;
202     gyro_y_pre[i].AXIS_Y = 0;
203     gyro_y_pre[i].AXIS_Z = 0;
204     gyro_x_pre[i].AXIS_X = 0;
205     gyro_x_pre[i].AXIS_Y = 0;
206     gyro_x_pre[i].AXIS_Z = 0;
207     euler_rc_y_pre[i].thx = 0;
208     euler_rc_y_pre[i].thy = 0;
209     euler_rc_y_pre[i].thz = 0;
210     euler_rc_x_pre[i].thx = 0;
211     euler_rc_x_pre[i].thy = 0;
212     euler_rc_x_pre[i].thz = 0;
213   }
214
215   /* USER CODE END 1 */
216
217   /* MCU Configuration--------------------------------------------------------*/
218
219   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
220   HAL_Init();
221
222   /* Configure the system clock */
223   SystemClock_Config();
224
225   /* Initialize all configured peripherals */
226   MX_GPIO_Init();
227   MX_ADC1_Init();
228   MX_TIM2_Init();
229   MX_TIM4_Init();
230   MX_TIM9_Init();
231   MX_USART1_UART_Init();
232   //MX_USB_DEVICE_Init();
233
234   /* USER CODE BEGIN 2 */
235
236   PRINTF("STEVAL-FCU001V1 FW rev.1.0 - Sep 2017¥n¥n");
237
238 //  Initialize Onboard LED
239   BSP_LED_Init(LED1);
240   BSP_LED_Init(LED2);
241   BSP_LED_Off(LED1);
242   BSP_LED_Off(LED2);
```

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

```
282   /* Register CFG_REG_A 0x60 = 0x8c */
283   /* Register 0x61 = 0x02 */
284   BSP_MAGNETO_Write_Reg(LIS2MDL_M_0_handle, 0x60, 0x8c);
285   BSP_MAGNETO_Write_Reg(LIS2MDL_M_0_handle, 0x61, 0x02);
286
287   /* Initialize Remote control*/
288   init_remote_control();
289
290   /* Initialize TIM2 for External Remocon RF receiver PWM Input*/
291   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_1);
292   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_2);
293   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_3);
294   HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_4);
295
296   /* Initialize TIM4 for Motors PWM Output*/
297   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);
298   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
299   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_3);
300   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_4);
301
302   /* Initialize General purpose TIM9 50Hz*/
303   HAL_TIM_Base_Start_IT(&htim9);
304
305   /* Initialize PID and set Motor PWM to zero */
306   PIDControlInit(&pid);
307   set_motor_pwm_zero(&motor_pwm);
308
309   /* Setup a timer with 5ms interval */
310   pid_interval = PID_SAMPLING_TIME*1000;
311   SetupTimer(&tim, pid_interval);
312
313   /* Start timer */
314   StartTimer(&tim);
315   ch = 0;
316   ch_flag = 0;
317
318   /* USER CODE END 2 */
319
320
321
322   /* Infinite loop */
323   /* USER CODE BEGIN WHILE */
324   while (1)
325   {
326     /* USER CODE END WHILE */
327
328     /* USER CODE BEGIN 3 */
329
330     if (tim9_event_flag == 1)
331     {     // Timer9 event: frequency 50Hz
```

ここからループ処理

基本周期の 5 倍の周期（6.19 ms、約 160Hz）で処理を実行.（50Hz は誤り？）
（～403 行目）

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

過去5サンプル分の移動平均をとる

AHRS（機体姿勢）の計算

クォータニオンからオイラー角への変換

プロポ操縦かん操作量を機体姿勢角度の目標値へ変換

```
381       {
382         euler_ahrs_offset.thx = 0;
383         euler_ahrs_offset.thy = 0;
384       }
385
386       Fly_origin.X_Degree = (int16_t)(euler_ahrs.thx * 5730);
387       Fly_origin.Y_Degree = (int16_t)(euler_ahrs.thy * 5730);
388       Fly_origin.Z_Degree = (int16_t)(euler_ahrs.thz * 5730);
389
390
391       if(gTHR<MIN_THR)
392       {
393         euler_rc.thz = 0;
394         euler_ahrs.thz = 0;
395       }
396
397       euler_rc_fil.thx = euler_rc.thx;
398       euler_rc_fil.thy = euler_rc.thy;
399       euler_rc_fil.thz = euler_rc.thz;
400
401       FlightControlPID_OuterLoop(&euler_rc_fil, &euler_ahrs, &ahrs, &pid);
402
403     }
404
405   if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == GPIO_PIN_SET)
406   {
407     ch_flag = 1;
408   }
409
410   if (isTimerEventExist(&tim))     // Check if a timer event is present
411   {
412
413         ClearTimer(&tim);           // Clear current event;
414
415         count2++;
416
417         mytimcnt++;
418         if (rc_connection_flag && rc_enable_motor)
419         {
420           if (mytimcnt%50 == 0)
421           BSP_LED_On(LED2);
422         }
423         else
424         {
425           if (mytimcnt%50 == 0)
426             BSP_LED_Toggle(LED2);
427         }
428     }
429
430     /* Added for debug on UART*/
```

（380 行目～）
姿勢制御の準備

姿勢角度制御の計算を実行

タイマイベント発生時の
処理（1ms 毎）

受信機からの信号があり，モータが動作可能状態であれば，LED を点灯とする

受信機からの信号がないか，モータが動作可能状態でない場合，LED を点滅させる

```
431     /* Remocon ELE, AIL, RUD, THR, Motor1_pwm, AHRS Euler angle x and y axis */
432     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);
433                                                                    UART へのデータ出力
434     /* 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*/
435     //PRINTF("%d¥t%f¥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);
436     /* MEMS Accelerometer RAW data */
437     //PRINTF("%d¥t%d¥t%d¥t¥n", acc.AXIS_X, acc.AXIS_Y, acc.AXIS_Z);
438
439//     /* Pressure data on UART for debug*/
440//     PRINTF("Pressure [atm] = %f¥n¥n",pre);
441//     /* Magnetometer data on UART for debug*/
442//     PRINTF("Magnetometer X = %d¥tY = %d¥tZ = %d¥n¥n", mag.AXIS_X, mag.AXIS_Y, mag.AXIS_Z);
443//
444//     /* Added reading of Battery voltage and data on UART */
445//     //VBAT_Sense = HAL_ADC_GetValue(&hadc1);
446//     HAL_ADC_Start(&hadc1);
447//         if (HAL_ADC_PollForConversion(&hadc1, 1000000) == HAL_OK)
448//         {
449//             VBAT_Sense = HAL_ADC_GetValue(&hadc1);
450//             VBAT = (((VBAT_Sense*3.3)/4095)*(BAT_RUP+BAT_RDW))/BAT_RDW;
451//             PRINTF("Battery voltage = %fV¥n¥n", VBAT);
452//         }
453//     HAL_ADC_Stop(&hadc1);
454
455   }
456   /* USER CODE END 3 */
457
458}
459
460/** System Clock Configuration
461*/
462void SystemClock_Config(void)
463{
464   RCC_OscInitTypeDef RCC_OscInitStruct;
465   RCC_ClkInitTypeDef RCC_ClkInitStruct;
466
467     /**Configure the main internal regulator output voltage
468     */
469   __HAL_RCC_PWR_CLK_ENABLE();
470
471   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
472
473     /**Initializes the CPU, AHB and APB busses clocks
474     */
475   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
```

```
476   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
477   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
478   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
479   RCC_OscInitStruct.PLL.PLLM = 16;
480   RCC_OscInitStruct.PLL.PLLN = 336;
481   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
482   RCC_OscInitStruct.PLL.PLLQ = 7;
483
484   HAL_RCC_OscConfig(&RCC_OscInitStruct);
485
486     /**Initializes the CPU, AHB and APB busses clocks
487     */
488   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
489                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
490   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
491   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
492   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
493   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
494   HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2);
495
496     /**Configure the Systick interrupt time
497     */
498   HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
499
500     /**Configure the Systick
501     */
502   HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
503
504   /* SysTick_IRQn interrupt configuration */
505   HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
506
507
508 }
509
510 /* ADC1 init function */
511 void MX_ADC1_Init(void)
512 {
513
514   ADC_ChannelConfTypeDef sConfig;
515
516     /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and
    number of conversion)
517     */
518   hadc1.Instance = ADC1;
519   hadc1.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV4;
520   hadc1.Init.Resolution = ADC_RESOLUTION12b;
521   hadc1.Init.ScanConvMode = DISABLE;
522   hadc1.Init.ContinuousConvMode = DISABLE;
523   hadc1.Init.DiscontinuousConvMode = DISABLE;
524   hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
```

システム・タイマ割り込みを 1 ms 毎に発生させる

```
  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;
  hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi2.Init.TIMode = SPI_TIMODE_DISABLED;
```

```
574   hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLED;
575   hspi2.Init.CRCPolynomial = 10;
576   HAL_SPI_Init(&hspi2);
577
578 }
579
580 /* TIM2 init function */
581 void MX_TIM2_Init(void)
582 {
583
584   TIM_ClockConfigTypeDef sClockSourceConfig;
585   TIM_MasterConfigTypeDef sMasterConfig;
586   TIM_IC_InitTypeDef sConfigIC;
587
588   htim2.Instance = TIM2;
589   htim2.Init.Prescaler = 20;
590   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
591   htim2.Init.Period = 32767;
592   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
593   HAL_TIM_Base_Init(&htim2);
594
595   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
596   HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig);
597
598   HAL_TIM_IC_Init(&htim2);
599
600   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
601   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
602   HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig);
603
604   sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
605   sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
606   sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
607   sConfigIC.ICFilter = 0;
608   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1);
609
610   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_2);
611
612   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_3);
613
614   HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_4);
615
616 }
617
618 /* TIM4 init function */
619 void MX_TIM4_Init(void)
620 {
621
622   TIM_ClockConfigTypeDef sClockSourceConfig;
623   TIM_MasterConfigTypeDef sMasterConfig;
```

```
624   TIM_OC_InitTypeDef sConfigOC;
625
626   htim4.Instance = TIM4;
627   #ifdef MOTOR_DC
628     htim4.Init.Prescaler = 84;                                          /* DC motor
    configuration - Freq 494Hz*/
629     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
630     htim4.Init.Period = 1999;
631   #endif
632   #ifdef MOTOR_ESC
633     htim4.Init.Prescaler = 100;                                         /* ESC motor
    configuration - Freq 400Hz*/
634     htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
635     htim4.Init.Period = 2075;
636   #endif
637
638   htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
639   HAL_TIM_Base_Init(&htim4);
640
641   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
642   HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig);
643
644   HAL_TIM_PWM_Init(&htim4);
645
646   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
647   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
648   HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig);
649
650   sConfigOC.OCMode = TIM_OCMODE_PWM1;
651   sConfigOC.Pulse = 0;
652   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
653   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
654   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1);
655
656   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2);
657
658   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_3);
659
660   HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_4);
661
662 }
663
664 /* TIM9 init function */
665 void MX_TIM9_Init(void)
666 {
667
668   TIM_ClockConfigTypeDef sClockSourceConfig;
669
670   htim9.Instance = TIM9;
671   htim9.Init.Prescaler = 51;
```

```
672   htim9.Init.CounterMode = TIM_COUNTERMODE_UP;
673   htim9.Init.Period = 1999;
674   htim9.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
675   HAL_TIM_Base_Init(&htim9);
676
677   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
678   HAL_TIM_ConfigClockSource(&htim9, &sClockSourceConfig);
679
680 }
681
682 /* USART1 init function */
683 void MX_USART1_UART_Init(void)
684 {
685
686   huart1.Instance = USART1;
687   huart1.Init.BaudRate = 115200;
688   huart1.Init.WordLength = UART_WORDLENGTH_8B;
689   huart1.Init.StopBits = UART_STOPBITS_1;
690   huart1.Init.Parity = UART_PARITY_NONE;
691   huart1.Init.Mode = UART_MODE_TX_RX;
692   huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
693   huart1.Init.OverSampling = UART_OVERSAMPLING_16;
694   HAL_UART_Init(&huart1);
695
696 }
697
698 /** Configure pins as
699         * Analog
700         * Input
701         * Output
702         * EVENT_OUT
703         * EXTI
704         * Free pins are configured automatically as Analog (this feature is enabled through
705         * the Code Generation settings)
706 */
707 void MX_GPIO_Init(void)
708 {
709
710   GPIO_InitTypeDef GPIO_InitStruct;
711
712   /* GPIO Ports Clock Enable */
713   __GPIOC_CLK_ENABLE();
714   __GPIOA_CLK_ENABLE();
715   __GPIOB_CLK_ENABLE();
716
717   /*Configure GPIO pins : PB4 PB5 */
718   GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5;
719   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
720   GPIO_InitStruct.Pull = GPIO_NOPULL;
721   GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
```

（670 行目～）
TIM9 の基本周波数・周期の設定

```
722   HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
723
724 }
725
726 /* USER CODE BEGIN 4 */
727 /*
728  *  Handle Timer9 interrupt @ 800Hz
729  *  Set the event flag and increase time index
730  */
731 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
732 {
733   if(sensor_init_cali == 0)
734   {
735     sensor_init_cali_count++;
736
737     if(sensor_init_cali_count > 800)
738     {
739       // Read sensor data and prepare for specific coodinate system
740       ReadSensorRawData(LSM6DSL_X_0_handle,     LSM6DSL_G_0_handle,     LIS2MDL_M_0_handle,
    LPS22HB_P_0_handle, &acc, &gyro, &mag, &pre);
741
742       acc_off_calc.AXIS_X += acc.AXIS_X;
743       acc_off_calc.AXIS_Y += acc.AXIS_Y;
744       acc_off_calc.AXIS_Z += acc.AXIS_Z;
745
746       gyro_off_calc.AXIS_X += gyro.AXIS_X;
747       gyro_off_calc.AXIS_Y += gyro.AXIS_Y;
748       gyro_off_calc.AXIS_Z += gyro.AXIS_Z;
749
750       if (sensor_init_cali_count >= 1600)
751       {
752         acc_offset.AXIS_X = acc_off_calc.AXIS_X * 0.00125;
753         acc_offset.AXIS_Y = acc_off_calc.AXIS_Y * 0.00125;
754         acc_offset.AXIS_Z = acc_off_calc.AXIS_Z * 0.00125;
755
756         gyro_offset.AXIS_X = gyro_off_calc.AXIS_X * 0.00125;
757         gyro_offset.AXIS_Y = gyro_off_calc.AXIS_Y * 0.00125;
758         gyro_offset.AXIS_Z = gyro_off_calc.AXIS_Z * 0.00125;
759
760         acc_off_calc.AXIS_X = 0;
761         acc_off_calc.AXIS_Y = 0;
762         acc_off_calc.AXIS_Z = 0;
763         gyro_off_calc.AXIS_X = 0;
764         gyro_off_calc.AXIS_Y = 0;
765         gyro_off_calc.AXIS_Z = 0;
766
767         sensor_init_cali_count = 0;
768         sensor_init_cali = 1;
769       }
770     }
```

TIM9 の周期で呼び出される

センサ初期化が終わっていない場合に実行

マイコンを起動して約 1 s 後から実行

センサ・データの読み込み

800 サンプル（約 1 s 間）にわたる各軸の加速度と角速度のデータを平均し，オフセット（0 点ずれ）を算出する

```
771   }
772
773   if(sensor_init_cali == 1)
774   {
775     tim9_cnt++;
776     tim9_cnt2++;
777
778     // Read sensor data and prepare for specific coodinate system
779     ReadSensorRawData(LSM6DSL_X_0_handle,      LSM6DSL_G_0_handle,      LIS2MDL_M_0_handle,
    LPS22HB_P_0_handle, &acc, &gyro, &mag, &pre);
780
781     if (rc_cal_flag == 1)
782     {
783       acc_off_calc.AXIS_X += acc.AXIS_X;
784       acc_off_calc.AXIS_Y += acc.AXIS_Y;
785       acc_off_calc.AXIS_Z += acc.AXIS_Z;
786
787       gyro_off_calc.AXIS_X += gyro.AXIS_X;
788       gyro_off_calc.AXIS_Y += gyro.AXIS_Y;
789       gyro_off_calc.AXIS_Z += gyro.AXIS_Z;
790
791       rc_cal_cnt++;
792
793       if (rc_cal_cnt >= 800)
794       {
795         acc_offset.AXIS_X = acc_off_calc.AXIS_X * 0.00125;
796         acc_offset.AXIS_Y = acc_off_calc.AXIS_Y * 0.00125;
797         acc_offset.AXIS_Z = acc_off_calc.AXIS_Z * 0.00125;
798
799         gyro_offset.AXIS_X = gyro_off_calc.AXIS_X * 0.00125;
800         gyro_offset.AXIS_Y = gyro_off_calc.AXIS_Y * 0.00125;
801         gyro_offset.AXIS_Z = gyro_off_calc.AXIS_Z * 0.00125;
802
803         acc_off_calc.AXIS_X = 0;
804         acc_off_calc.AXIS_Y = 0;
805         acc_off_calc.AXIS_Z = 0;
806         gyro_off_calc.AXIS_X = 0;
807         gyro_off_calc.AXIS_Y = 0;
808         gyro_off_calc.AXIS_Z = 0;
809
810         rc_cal_cnt = 0;
811         rc_cal_flag = 0;
812       }
813     }
814
815     acc.AXIS_X -= acc_offset.AXIS_X;
816     acc.AXIS_Y -= acc_offset.AXIS_Y;
817     acc.AXIS_Z -= (acc_offset.AXIS_Z - 1000);
818     gyro.AXIS_X -= gyro_offset.AXIS_X;
819     gyro.AXIS_Y -= gyro_offset.AXIS_Y;
```

センサ初期化完了後に実行

センサ・データの 0 点ずれを補正
（～820 行目）

```
820     gyro.AXIS_Z -= gyro_offset.AXIS_Z;
821
822     // Save filtered data to acc_FIFO
823     acc_FIFO[tim9_cnt2-1].AXIS_X = acc.AXIS_X;
824     acc_FIFO[tim9_cnt2-1].AXIS_Y = acc.AXIS_Y;
825     acc_FIFO[tim9_cnt2-1].AXIS_Z = acc.AXIS_Z;
826
827     // IIR Filtering on gyro
828     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
829                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_X + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_X;
830     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
831                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_Y + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_Y;
832     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
833                                                 +
    gyro_fil_coeff.a1*gyro_y_pre[0].AXIS_Z + gyro_fil_coeff.a2*gyro_y_pre[1].AXIS_Z;
834     // Shift IIR filter state
835     for(int i=1;i>0;i--)
836     {
837       gyro_x_pre[i].AXIS_X = gyro_x_pre[i-1].AXIS_X;
838       gyro_x_pre[i].AXIS_Y = gyro_x_pre[i-1].AXIS_Y;
839       gyro_x_pre[i].AXIS_Z = gyro_x_pre[i-1].AXIS_Z;
840       gyro_y_pre[i].AXIS_X = gyro_y_pre[i-1].AXIS_X;
841       gyro_y_pre[i].AXIS_Y = gyro_y_pre[i-1].AXIS_Y;
842       gyro_y_pre[i].AXIS_Z = gyro_y_pre[i-1].AXIS_Z;
843     }
844     gyro_x_pre[0].AXIS_X = gyro.AXIS_X;
845     gyro_x_pre[0].AXIS_Y = gyro.AXIS_Y;
846     gyro_x_pre[0].AXIS_Z = gyro.AXIS_Z;
847     gyro_y_pre[0].AXIS_X = gyro_fil.AXIS_X;
848     gyro_y_pre[0].AXIS_Y = gyro_fil.AXIS_Y;
849     gyro_y_pre[0].AXIS_Z = gyro_fil.AXIS_Z;
850
851     //  Save filtered data to gyro_FIFO
852     gyro_FIFO[tim9_cnt2-1].AXIS_X = gyro_fil.AXIS_X;
853     gyro_FIFO[tim9_cnt2-1].AXIS_Y = gyro_fil.AXIS_Y;
854     gyro_FIFO[tim9_cnt2-1].AXIS_Z = gyro_fil.AXIS_Z;
855
856
857     if(tim9_cnt2 == FIFO_Order)
858     {
859       tim9_cnt2 = 0;
860       tim9_event_flag = 1;
861       for(int i=0;i<FIFO_Order;i++)
862       {
863         acc_ahrs_FIFO[i].AXIS_X = acc_FIFO[i].AXIS_X;
```

加速度データをFIFOバッファに格納

角速度をノイズ・フィルタ（IIRフィルタ）に通す

角速度データをFIFOバッファに格納

```
864         acc_ahrs_FIFO[i].AXIS_Y = acc_FIFO[i].AXIS_Y;
865         acc_ahrs_FIFO[i].AXIS_Z = acc_FIFO[i].AXIS_Z;
866         gyro_ahrs_FIFO[i].AXIS_X = gyro_FIFO[i].AXIS_X;
867         gyro_ahrs_FIFO[i].AXIS_Y = gyro_FIFO[i].AXIS_Y;
868         gyro_ahrs_FIFO[i].AXIS_Z = gyro_FIFO[i].AXIS_Z;
869       }
870     }
871
872
873       gyro_rad.gx = gyro_fil.AXIS_X*COE_MDPS_TO_RADPS;
874       gyro_rad.gy = gyro_fil.AXIS_Y*COE_MDPS_TO_RADPS;
875       gyro_rad.gz = gyro_fil.AXIS_Z*COE_MDPS_TO_RADPS;
876
877       euler_ahrs.thz += gyro_rad.gz*PID_SAMPLING_TIME;
878
879       if(gTHR<MIN_THR)
880       {
881         euler_rc.thz = 0;
882         euler_ahrs.thz = 0;
883       }
884
885
886       if (rc_connection_flag && rc_enable_motor)
887       {   // Do PID Control
888         FlightControlPID_innerLoop(&euler_rc_fil, &gyro_rad, &ahrs, &pid, &motor_pwm);
889       }
890       else
891       {
892         // set motor output zero
893         set_motor_pwm_zero(&motor_pwm);
894       }
895
896       if(gTHR<MIN_THR)
897       {
898         set_motor_pwm_zero(&motor_pwm);
899       }
900
901       set_motor_pwm(&motor_pwm);       /* To comment if want to debug remocon calibration
    switching off the motors */
902   }
903 }
904
905
906 /**
907 * @brief  Initialize all sensors
908 * @param  None
909 * @retval None
910 */
911 static void initializeAllSensors( void )
912 {
```

（859 行目～）
AHRS 計算（main()関数のループから呼び出し）に渡すためのデータを作成

角速度の単位を[rad/s]へ変換

機体方向のオイラー角を算出（Z 軸角速度を積分）

プロポ・受信機が正常動作し，かつモータが動作可能状態であれば，インナ・ループ角速度制御の計算を実行

モータへ PWM 信号を出力

```
913   if (BSP_ACCELERO_Init( LSM6DSL_X_0, &LSM6DSL_X_0_handle ) != COMPONENT_OK)
914   {
915     while(1);
916   }
917
918   if (BSP_GYRO_Init( LSM6DSL_G_0, &LSM6DSL_G_0_handle ) != COMPONENT_OK)
919   {
920     while(1);
921   }
922
923   if (BSP_MAGNETO_Init( LIS2MDL_M_0, &LIS2MDL_M_0_handle ) != COMPONENT_OK)
924   {
925     while(1);
926   }
927
928
929   if (BSP_PRESSURE_Init( LPS22HB_P_0, &LPS22HB_P_0_handle ) != COMPONENT_OK)
930   {
931     while(1);
932   }
933
934 //  if (BSP_TEMPERATURE_Init( LPS22HB_T_0, &LPS22HB_T_0_handle ) != COMPONENT_OK)
935 //  {
936 //    while(1);
937 //  }
938 //
939
940 }
941
942 /**
943 * @brief  Enable all sensors
944 * @param  None
945 * @retval None
946 */
947 void enableAllSensors( void )
948 {
949   BSP_ACCELERO_Sensor_Enable( LSM6DSL_X_0_handle );
950   PRINTF("LSM6DSL MEMS Accelerometer initialized and enabled¥n");
951   BSP_GYRO_Sensor_Enable( LSM6DSL_G_0_handle );
952   PRINTF("LSM6DSL MEMS Gyroscope initialized and enabled¥n");
953   BSP_MAGNETO_Sensor_Enable( LIS2MDL_M_0_handle );
954   PRINTF("LIS2MDL Magnetometer initialized and enabled¥n");
955   BSP_PRESSURE_Sensor_Enable( LPS22HB_P_0_handle );
956   PRINTF("LPS22HB Pressure sensor initialized and enabled¥n");
957 //  BSP_TEMPERATURE_Sensor_Enable( LPS22HB_T_0_handle );
958 }
959
960
961
962 /* USER CODE END 4 */
```

```
963
964#ifdef USE_FULL_ASSERT
965
966/**
967   * @brief Reports the name of the source file and the source line number
968   * where the assert_param error has occurred.
969   * @param file: pointer to the source file name
970   * @param line: assert_param error line source number
971   * @retval None
972   */
973void assert_failed(uint8_t* file, uint32_t line)
974{
975  /* USER CODE BEGIN 6 */
976  /* User can add his own implementation to report the file name and line number,
977     ex: printf("Wrong parameters value: file %s on line %d¥r¥n", file, line) */
978  /* USER CODE END 6 */
979
980}
981
982#endif
983
984/**
985  * @}
986  */
987
988/**
989  * @}
990*/
991
992/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
993
```