UM2329

User manual

Getting started with the STSW-FCU001 reference design firmware for mini drones

Introduction

The STSW-FCU001 firmware enables the STEVAL-FCU001V1 and STEVAL-FCU001V2 evaluation board to support quadcopter drone design.

It comes with six main applications: a remote control interface to interpret data command from remote control; sensor management to retrieve sensor data; an attitude heading reference system (AHRS) to elaborate sensor data in roll pitch yaw angles (quaternions); flight control merging the AHRS output and information coming from the remote control to define the flight strategy; PID control to give commands to the four quadcopter motors.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
AHRS	Attitude and heading reference system
API	Application programming interface
BLE	Bluetooth low energy
ESC	Electronic speed controller
FCU	Flight controller unit
HAL	Hardware abstraction layer
IDE	Integrated development environment
PID	Proportional-integral-derivative controller
PPM	Pulse position modulation

2 STSW-FCU001 firmware structure

2.1 Overview

The STSW-FCU001 firmware is designed for mini sized quadcopters equipped with DC motors and larger quadcopters with external ESCs and DC brushless motors.

It features:

- Implementation of a flight controller unit based on the STEVAL-FCU001V1 and STEVAL-FCU001V2 evaluation boards
- Bluetooth connectivity protocol to dialog with ST_BLE_DRONE App.
- High performance stabilization for quadcopters in aerial movements
- Full support for the on-board gyroscope and accelerometer
- Barometer support for holding altitude available
- Magnetometer support for direction detection (available only on STEVAL-FCU001V1)
- Embedded routine to accept commands from the most common remote controls

It is built on STM32Cube to facilitate customization and integration of additional middleware algorithms.

You can send commands to the evaluation board via smartphone or tablet through BLE connectivity, or via an RF receiver module connected to the PWM input port.

2.2 Architecture

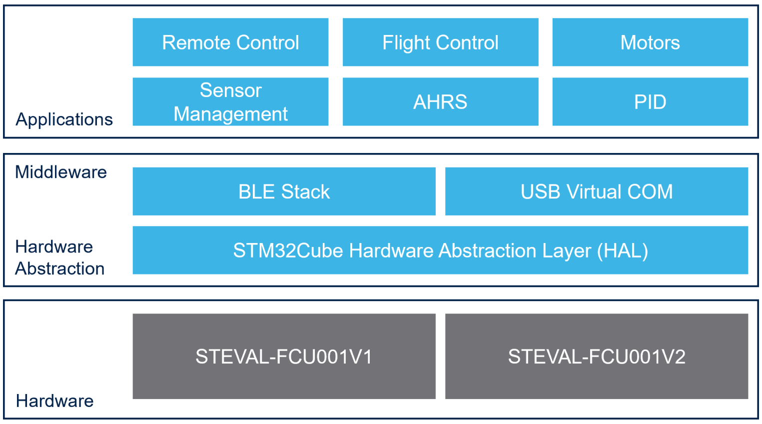
The firmware is based on STM32Cube technology and expands STM32Cube based packages.

The package provides a board support package (BSP) for the sensors and the middleware components for Bluetooth low energy communication with any external mobile device.

The firmware driver layers to access and use the hardware components are:

- STM32Cube HAL layer: simple, generic, multi-instance application programming interfaces (APIs) which interact with the upper layer applications, libraries and stacks. The APIs are based on the common STM32Cube framework so the other layers (e.g., middleware) can function without requiring any specific hardware information for a given microcontroller unit (MCU), thus improving library code reusability and guaranteeing easy portability across devices.
- Board support package (BSP) layer: provides firmware support for the evaluation board (excluding MCU) peripherals (LEDs, user buttons, etc.) but can also be used for the board serial and version information, and to support initializing, configuring and reading data from sensors.
  You can build the firmware using specific APIs for the following hardware subsystems:
  – Platform: to control and configure all the devices in the battery and power subsystem, button, LEDs and GPIOs
  – Sensors: to link, configure and control all the sensors involved

Figure 1. STSW-FCU001 firmware architecture



Important: *The middleware (USB virtual COM) in the figure above is not currently included in the firmware. It is due for inclusion in a future release of the firmware.*

2.3 Folder structure

The STSW-FCU001 Zip file is organized in two main folders, one containing the firmware for STEVAL-FCU001V1 and the other one containing the firmware for STEVAL-FCU001V2.

These are described in next two sections.

2.3.1 Firmware dedicated to STEVAL-FCU001V1

The first folder is structured in the following sub-folders:

1. User: is the firmware project core
2. Drivers: with device drivers and the board configuration
3. Middlewares: not used at application level in this firmware implementation

User

This folder contains the following files:

User application files

- **Ahrs**: algorithm based on the complementary filter to estimate the drone position
- **Basic_Math**: square root and inverse square root mathematical operations
- **Debug**: debug functions
- **Flight_Control**: functions to compute the PID output and control the motors
- **Motor**: configuration of motor driving signals for DC motor and external ESC configurations
- **PID**: PID controller
- **Quaternion**: quaternion computation
- **RC**: decodification of remote control PPM input signals
- **Sensor_Data**: sensor data conversion depending on the accelerometer and gyroscope mounting configuration

Standard STM32Cube application files

- Main
- Stm32f4xx_hal_msp
- Stm32f4xx_it
- Stm32f4xx_nucleo

USB device related files

- Timer
- Usb_device
- Usbd_cdc_if
- Usbd_conf
- Usbd_desc

Drivers

This folder contains the device drivers and board configuration used in ***main.c*** :

- **Board**: hardware configuration (GPIO and peripherals assignment, etc.).
- **CMSIS**: ARM® Cortex®-M4 device peripheral access layer system source file.
- **Components**: MEMS sensors (6-axis accelerometer/gyroscope, 3-axis magnetometer and pressure sensor) device drivers.
- **STM32F4xx_HAL_Driver**: STM32 HAL (hardware abstraction layer) peripheral drivers.

Middleware

This folder contains a USB library which is currently not used at the application level in this firmware implementation.

2.3.2 Firmware dedicated to STEVAL-FCU001V2

The second folder is structured in the following sub-folders:

1. Core: is the firmware project core
2. Drivers: with device drivers and the board configuration
3. Middlewares: Bluetooth Low Energy stack
4. X-CUBE-MEMS1: Sensor and motion algorithm software expansion for STM32Cube

Core

This folder contains the following files:

- **Ahrs**: algorithm based on the complementary filter to estimate the drone position
- **Basic_Math**: square root and inverse square root mathematical operations
- **Debug**: debug functions
- **Flight_Control**: functions to compute the PID output and control the motors
- **Motor**: configuration of motor driving signals for DC motor and external ESC configurations
- **PID**: PID controller
- **Quaternion**: quaternion computation
- **RC**: decodification of remote control PPM input signals
- **Main**

Drivers

This folder contains the device drivers and board configuration used in **main.c** :

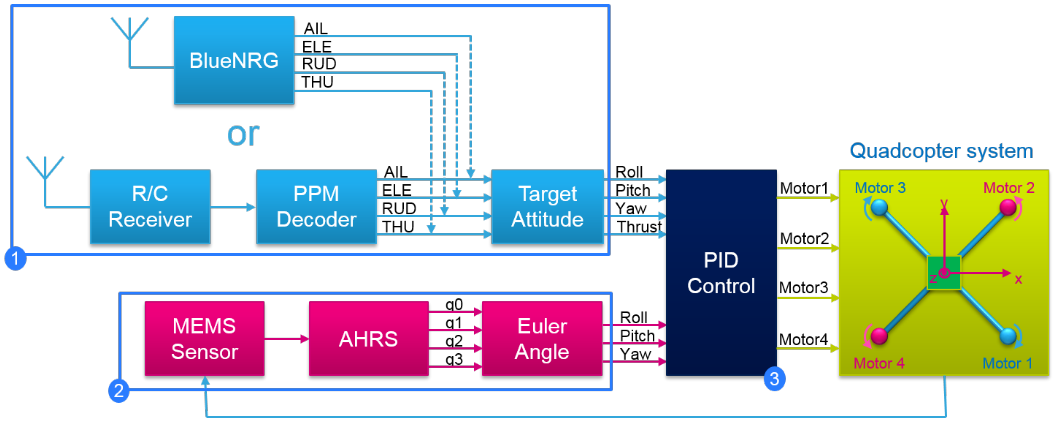
- **Board**: hardware configuration (GPIO and peripherals assignment, etc.)
- **CMSIS**: ARM® Cortex®-M4 device peripheral access layer system source file
- **STM32F4xx_HAL_Driver**: STM32 HAL (hardware abstraction layer) peripheral drivers.

Middleware

This folder contains the Bluetooth Low Energy stack.

3 STSW-FCU001 firmware project modules

Figure 2. **STSW-FCU001 firmware project architecture**



As shown in the figure above, the firmware project architecture is based on the following blocks:

1. Modules to decode the commands from the external Remocon (remote controller) or the smartphone app (i.e. ST_BLE_DRONE) and translate them in the attitude (Euler angle) target input for the PID control, plus the thrust (or throttle) common to all motors:
   – R/C receiver
   – PPM decoder
   – Target attitude
   – BlueNRG command translator
2. Modules to estimate the real position in the 3 axes based on:
   – MEMS sensor setup of accelerometer and gyroscope key parameters
   – AHRS algorithm estimation through quaternion and complementary filter
   – Euler angle translation of quaternions into position
3. PID control to compute the differentiation between target attitude and estimated position and control the motors to adjust the position, plus the thrust/throttle contribution.

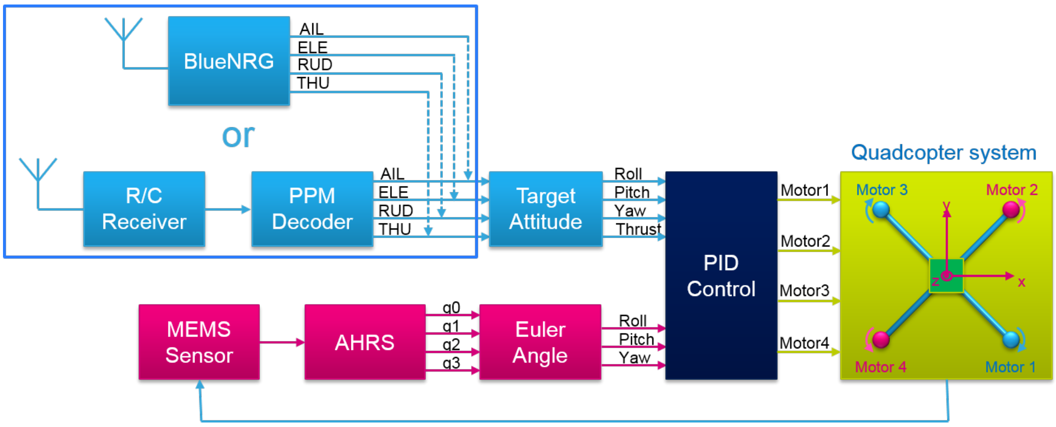
The firmware implementation enables the quadcopter to receive standard PPM input commands from the Remocon RF receiver board or by a smartphone or tablet via the board BLE module.

*Note: The current firmware project implementation supports only the **angle** or **stabilize** modes (where the Remocon commands fix a target angle for the quadcopter).*

Attention: *The firmware project is intended as a reference design for the customer and must be handled by professional users. It has not been conceived and tested for commercial use.*

3.1 R/C receiver and PPM decoder

Figure 3. **STSW-FCU001 firmware project architecture: R/C receiver and PPM decoder modules**



The modules highlighted in the figure above are implemented in **rc.c** and provide an interface to work with remote control receiver signals, according to the procedures and modes listed below.

When using BLE to control the drone, the BLE stack manages the communication and the PPM decoder is absent, since the values are given directly by the BLE module with the variables:

`int16_t gAIL, gELE, gTHR, gRUD`

To switch from the RF R/C to the BLE app configuration, simply change the define in the file **rc.h**:

```
//#define REMOCON_PWM // Remocon RX 4 channel PWM
#define REMOCON_BLE // BLE Remocon App
```

Input capture mode

This mode calculates the pulse width of each PPM signal via TIM2 (4 channels).

For details about TIM2 PPM decoding and data filtering, call the function `void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)`, which processes the input capture channels interrupt to calculate the pulse of each RC channel and save the pulse width value in the global variable rc t[4]. The default time step is 0.25 µs/LSB.

By calling the function `update_rc_data(idx)`, you can convert the pulse width signal into the real control signal stored in global variables: gAIL, gELE, gTHR, gRUD.

gAIL, gELE and gRUD are 0 centered (±0.5 ms with 0.25 µs/LSB) whereas gTHE is 0~1 ms with 0.25 µs/LSB.

Other variables to check the Remocon connection are:

`volatile int rc_timeout; >> R/C timeout counter` and `char rc_connection_flag; >> R/C connection status.`

R/C channels and R/C calibration setup

The RC channels are mapped in the file **config_drone.h**, where you can also define the timeout interval (if no signal from the Remocon motor):

```
#define RC_TIMEOUT_VALUE 30
// Define the GPIO port for R/C input capture channels
#define RC_CHANNEL1_PORT GPIOA
#define RC_CHANNEL1_PIN GPIO_PIN_0
```

After connecting a new Remocon receiver (and pairing it to the relative transmitter), you must perform calibration by modifying the parameters (center and full scale positions, arming threshold) in **rc.h**:

```
// Definition for AIL(Roll) Channel
#define AIL_LEFT 7661
#define AIL_MIDDLE 6044
#define AIL_RIGHT 4434
…
#define RC_FULLSCALE 1500
#define RC_CAL_THRESHOLD 1200
```

Figure 4. **STSW-FCU001 firmware user project: Remocon and R/C channels assigned**

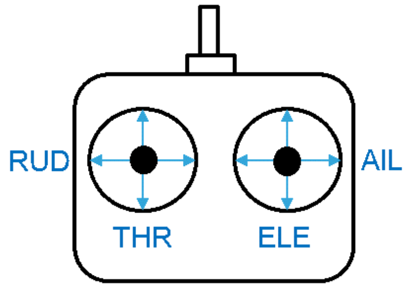
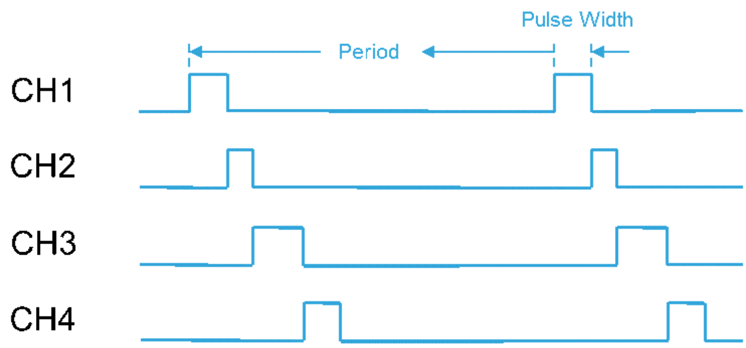


Table 2. **R/C channels assigned and pulse period modulation (L → Left; R → Right; U → Up; D → Down)**

Channel	Control	Function	Position	L/D	Modulation	R/U
CH1	AIL	Roll	R: LR	2 ms	1.5 ms	1 ms
CH2	ELE	Pitch	R: UD	1 ms	1.5 ms	2 ms
CH3	THR	Thrust	L: UD	1 ms	-	2 ms
CH4	RUD	Yaw	L: LR	2 ms	1.5 ms	1 ms

Figure 5. **STSW-FCU001 firmware user project: PPM modulation with period = ~5..20 ms (50..200 Hz) and pulse width = 1~2 ms**



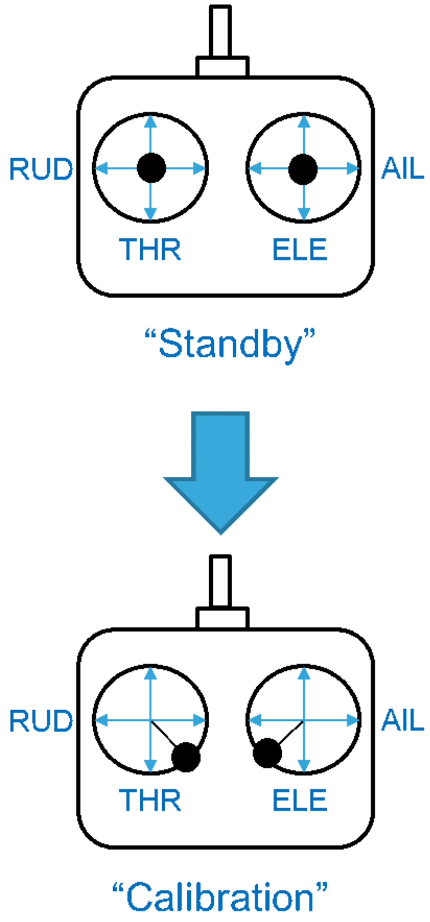
Sensor calibration procedure

If the battery is applied to the FCU (or the Reset button is pushed) when the quadcopter is not on a flat surface (or the FCU is not mounted flat on the frame), the AHRS Euler angle will have an offset, which can be eliminated by running a calibration procedure via the Remocon after startup.

The **calibration** procedure can be customized via the function `update_rc_data`:

```
if ( (gTHR == 0) && (gELE < - RC_CAL_THRESHOLD) && (gAIL > RC_CAL_THRESHOLD)
&& (gRUD < - RC_CAL_THRESHOLD))
{
rc_cal_flag = 1;
}
```

Figure 6. **STSW-FCU001 firmware user project: calibration procedure**



Motor arming procedure

To avoid any damage or injury when the quadcopter battery is inserted and the Remocon is switched ON, an *Arming* procedure is always activated: motors are switched OFF until a certain sequence of commands on the Remocon is applied.

The procedure can be customized via the function `update_rc_data`:

```
if ( (gTHR == 0) && (gELE < - RC_CAL_THRESHOLD) && (gAIL < - RC_CAL_THRESHOLD)
&& (gRUD > RC_CAL_THRESHOLD))
{
rc_enable_motor = 1;
fly_ready = 1;
}
```

Figure 7. STSW-FCU001 firmware user project: arming procedure

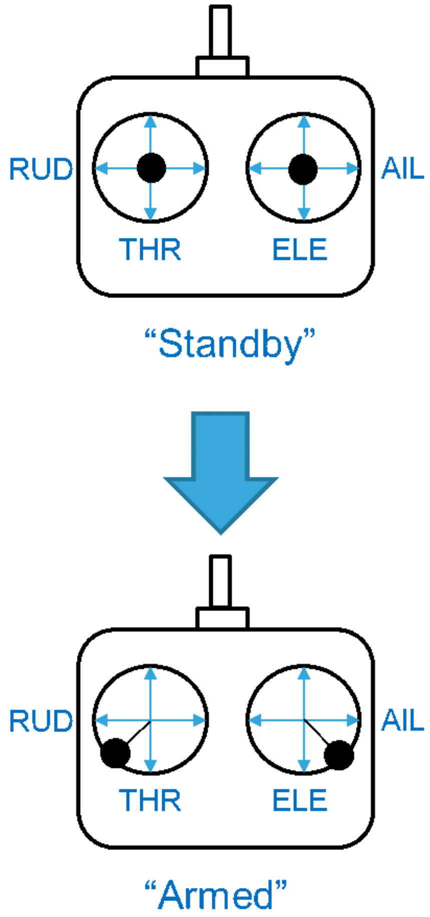
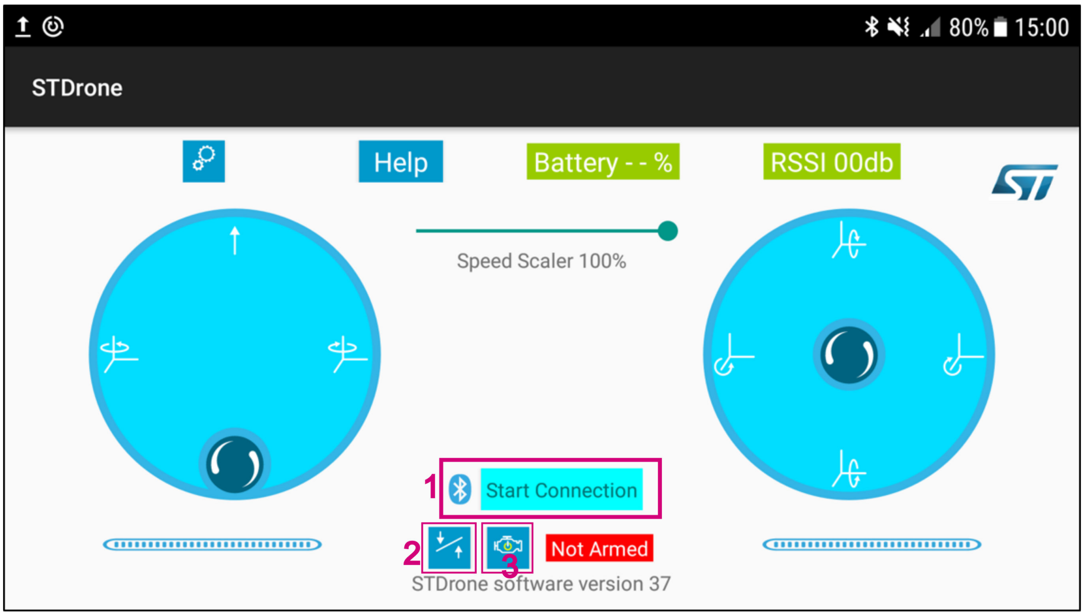


Figure 8. Equivalence of joystick movements in the STDrone app

1. Button to connect to the drone
2. Button to calibrate sensors
3. Button to arm the motors

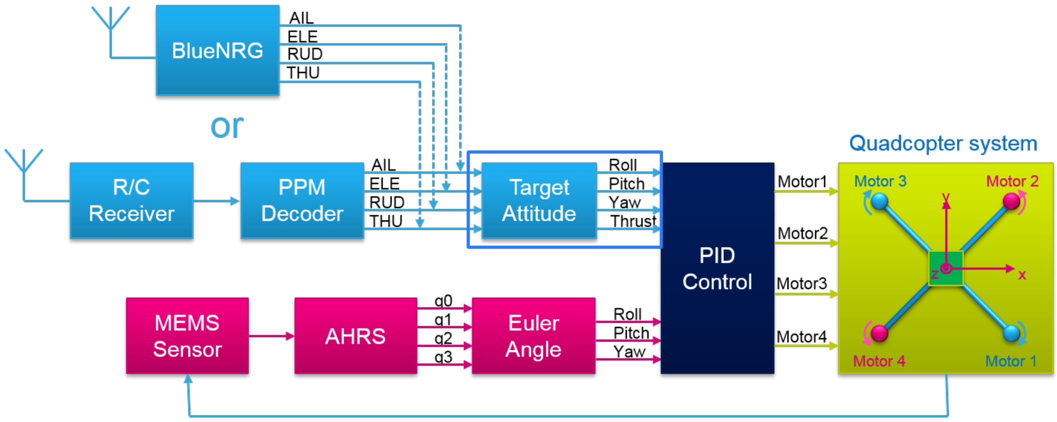


Both the sensor calibration and the motor arming procedures can be activated by the ST_BLE_DRONE app: tap the related button for few seconds and, when the calibration is complete, the button turns green.

In the same way, to arm the motor, tap the related button and the "Armed" message appears.

3.2 Target attitude

Figure 9. STSW-FCU001 firmware project architecture: target attitude module



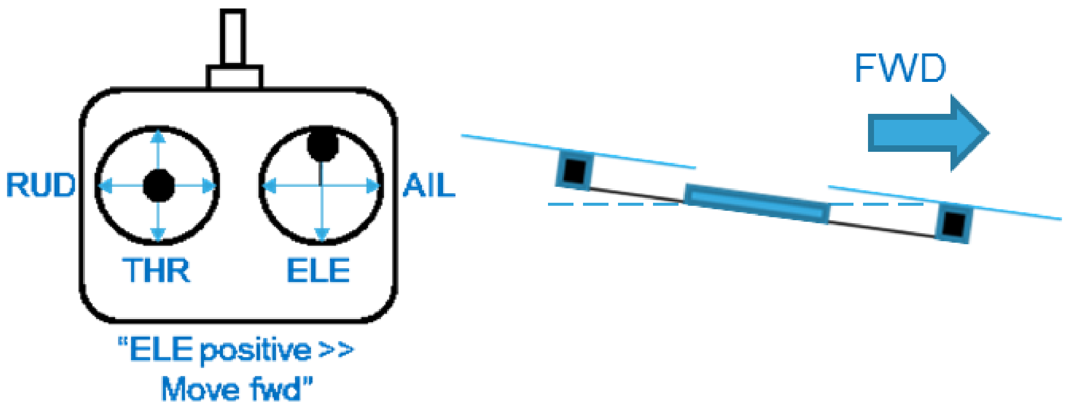
The conversion from Remocon Euler angle to AHRS Euler angle (needed to set the PID control loop target) is implemented in the **rc.c** file via the following function `void GetTargetEulerAngle(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs).`

The data from the Remocon (i.e. gELE) is normalized to the full scale value and converted into angles:

```
t1 = gELE;
if (t1 > RC_FULLSCALE)
t1 = RC_FULLSCALE;
else if (t1 < -RC_FULLSCALE)
t1 = - RC_FULLSCALE;
euler_rc->thx = -t1 * max_pitch_rad / RC_FULLSCALE;
const float max_pitch_rad = I*PITCH_MAX_DEG/180.0;
```

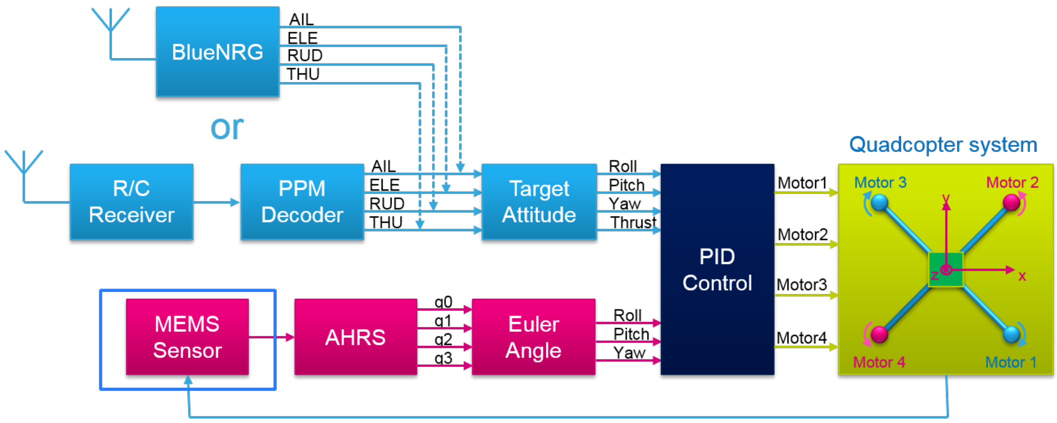
The negative value (-t1) is related to the increasing ELE (compared to the middle level) which, moving forward, corresponds to a negative target angle for the pitch.

Figure 10. STSW-FCU001: Remocon Euler angle converted to AHRS Euler angle



3.3 MEMS sensors

Figure 11. STSW-FCU001 firmware project architecture: MEMs sensor module



The main sensors used for flight stabilization are the accelerometer and gyroscope.

The firmware project contains also drivers for magnetometer (only for STEVAL-FCU001V1) and pressure sensors but they are not used in the current firmware implementation.

The MEMS sensor module converts sensor data to the related coordinate; X and Y axes are oriented in line with the on-board accelerometer and gyroscope sensors:

1. Sensors are initialized in the main.c file via the function CUSTOM_MOTION_SENSOR_Init.
2. Raw data from 6-axis accelerometer and gyroscope sensors are converted in [mg] and [mdps] and moved to the coordinate system used (FCU board orientation vs. motor configuration).

The default coordinate system of the above configuration is Option 3.

The translation of coordinates for AHRS can be performed using the options below.

Option 1

```
Drone FWD direction > (ACC -X)
                  (x)<--O(z)
                        |
                       V(y)
```

Option 2

```
Drone FWD direction > (ACC -Y)
                  ^(x)
                    |
                (y)<--O(z)
```

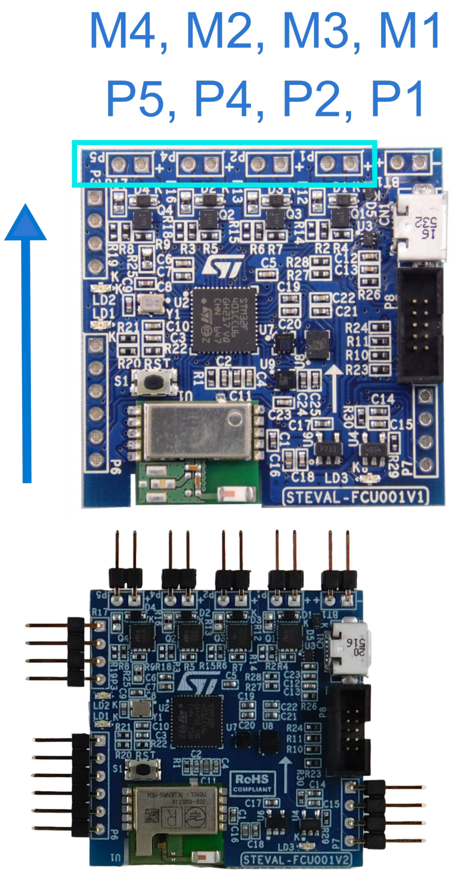
Option 3

```
Drone FWD direction ^ (ACC X)
                      ^(y)
                        |
                    (z)O-->(x)
```

Option 4

```
Drone FWD direction < (ACC Y)
                     (z)O-->(y)
                          |
                         V(x)
```

Figure 12. **STEVAL-FCU001V1 and STEVAL-FCU001V2 evaluation boards: drone front direction**



3.3.1 Accelerometer and gyroscope sensor setup

The flight control algorithm is based on data generated by accelerometer and gyroscope sensors.

You should minimize motor vibrations propagated to the FCU through the frame (by using, for example, a mechanical dumper under the FCU).

The effect of this vibration on the system can be modulated by setting few key parameters (in the **main.c** file of the current firmware project implementation).

Table 3. **Accelerometer setup**

Parameter	Value
Analog filter bandwidth	1.5 kHz (1)
Full scale selection(2)	±4 g
Output data rate and power mode selection (3)	1.6 kHz(4)
Composite filter input selection default(5)	Default
Low-pass filter(6)	Default(7)

1. *Default.*
2. *FS.*
3. *ODR.*
4. *For default power down mode.*
5. *ODR/2.*
6. *LPF2.*
7. *Set to **Low pass filter enabled** at ODR/50 or ODR/100 depending on the mechanical vibration noise.*

```
BSP_ACCELERO_Set_ODR_Value(LSM6DSL_X_0_handle, 1660.0); /* ODR 1.6kHz */
BSP_ACCELERO_Set_FS(LSM6DSL_X_0_handle, FS_MID); /* FS 4g */
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 */
LSM6DSL_ACC_GYRO_W_LowPassFiltSel_XL(LSM6DSL_X_0_handle,
LSM6DSL_ACC_GYRO_LPF2_XL_ENABLE); /* Enable LPF2 filter in composite filter block */
LSM6DSL_ACC_GYRO_W_HPCF_XL(LSM6DSL_X_0_handle,
LSM6DSL_ACC_GYRO_HPCF_XL_DIV100); /* Low pass filter @ ODR/100 */
uint8_t tmp_6axis_reg_value;
BSP_ACCELERO_Read_Reg(LSM6DSL_X_0_handle, 0x10, &tmp_6axis_reg_value);
tmp_6axis_reg_value = tmp_6axis_reg_value & 0xFE /* Set LSB to 0 >> Analog filter 1500Hz*/
BSP_ACCELERO_Write_Reg(LSM6DSL_X_0_handle, 0x10, tmp_6axis_reg_value);
```

Figure 13. LSM6DSL block diagram: accelerometer chain, composite filter and key parameters

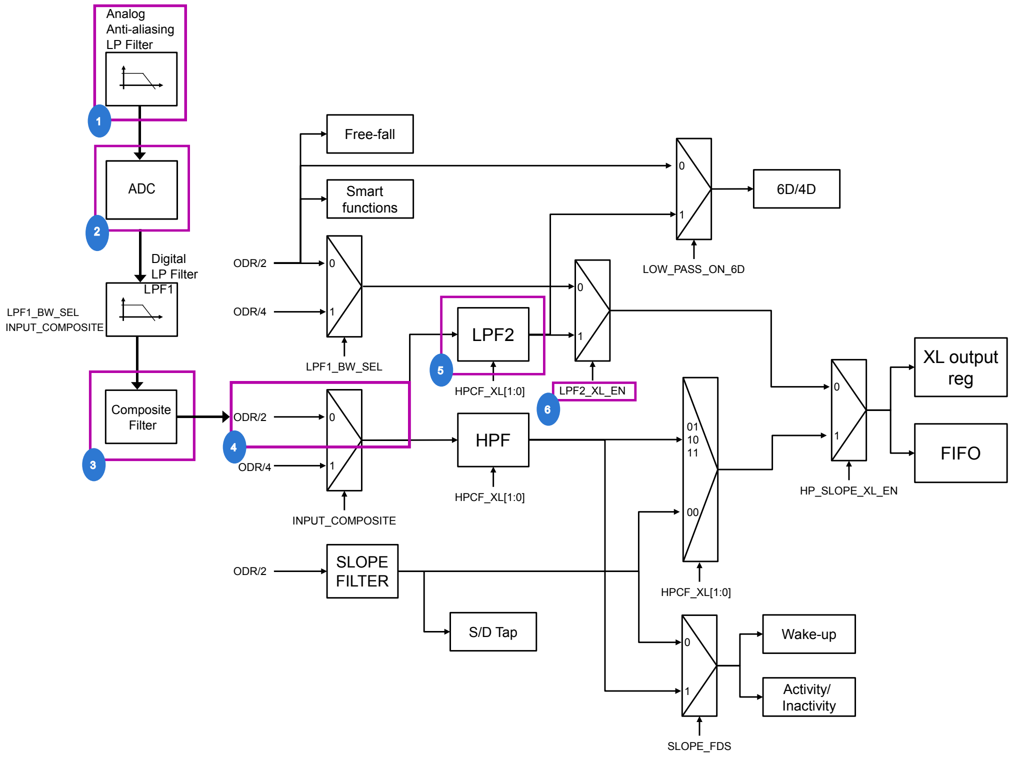


Table 4. Gyroscope setup

Parameter	Value
Full scale	2000 dps(1)
Output data rate(2)	104 Hz (3)
Low-pass filter bandwidth(4)	Narrow(5)

1. *The default value is 245 dps.*
2. *ODR.*
3. *The default value is power down.*
4. *LPF1 FTYPE.*
5. *10 b.*

```
BSP_GYRO_Set_FS(LSM6DSL_G_0_handle, FS_HIGH); /* Set FS to 2000dps */
BSP_GYRO_Set_ODR(LSM6DSL_G_0_handle, ODR_HIGH);   /* Set ODR to 104Hz */
LSM6DSL_ACC_GYRO_W_LP_BW_G(LSM6DSL_G_0_handle,
LSM6DSL_ACC_GYRO_LP_G_NARROW); /* LPF1 FTYPE set to 10b */
```

3.3.2 Pressure sensor and magnetometer

To use the pressure sensor (for altitude estimation) and the magnetometer (for e-Compass), you must enable the sensors in the **config_drone.h** file as follows:

```
// Use magnetic sensor or not 1 = use
#define USE_MAG_SENSOR       0
// Use presure sensor or not 1 = use
#define USE_PRESSURE_SENSOR  0
```

3.3.3 Sensor calibration after startup

After inserting the battery, put the quadcopter on a flat surface and press the Reset button; if the quadcopter is set on an inclined plane, an automatic calibration process will adjust the **euler_ahrs** coordinates accordingly.

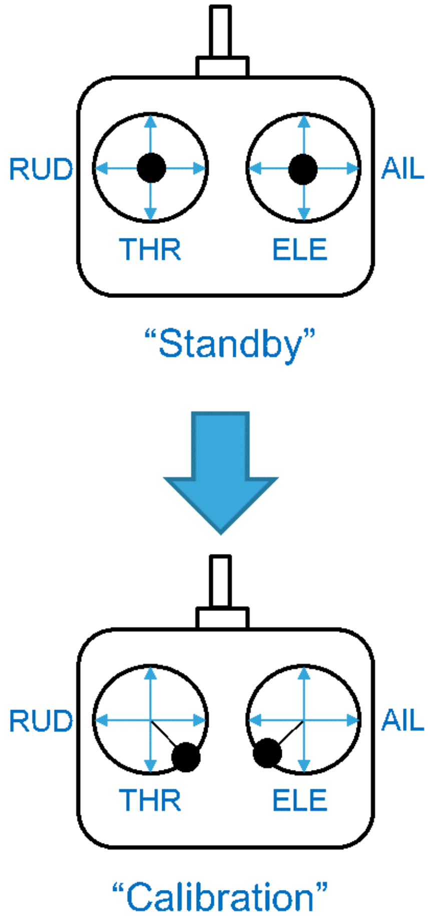
The `HAL_TIM_PeriodElapsedCallback` function (in the **main.c** file) calibrates the accelerometer and the gyroscope:

1. when the **sensor_init_cali variable** is 0, just after system startup (battery inserted or Reset button pushed)
2. when the **rc_cal_flag variable** is 1, manual calibration is launched before arming the motors if the sticks are in a certain position (`check update_rc_data` function in the **rc.c** file):

```
if ( (gTHR == 0) && (gELE < - RC_CAL_THRESHOLD) &&
(gAIL > RC_CAL_THRESHOLD) && (gRUD < -
RC_CAL_THRESHOLD)) {
rc_cal_flag = 1;
}

if(sensor_init_cali == 0)
 {
 sensor_init_cali_count++;
 if(sensor_init_cali_count > 800)
 {
 // Read sensor data and prepare for specific coordinate system
ReadSensorRawData(&acc, &gyro, &mag, &pre);
 acc_off_calc.AXIS_X += acc.AXIS_X;
 acc_off_calc.AXIS_Y += acc.AXIS_Y;
 acc_off_calc.AXIS_Z += acc.AXIS_Z;
 gyro_off_calc.AXIS_X += gyro.AXIS_X;
 gyro_off_calc.AXIS_Y += gyro.AXIS_Y;
 gyro_off_calc.AXIS_Z += gyro.AXIS_Z;
 if (sensor_init_cali_count >= 1600)
 {
 acc_offset.AXIS_X = acc_off_calc.AXIS_X * 0.00125;
 acc_offset.AXIS_Y = acc_off_calc.AXIS_Y * 0.00125;
 ...
 sensor_init_cali_count = 0;
 sensor_init_cali = 1;
 }
 }
```

Figure 14. **STSW-FCU001: sensor calibration**

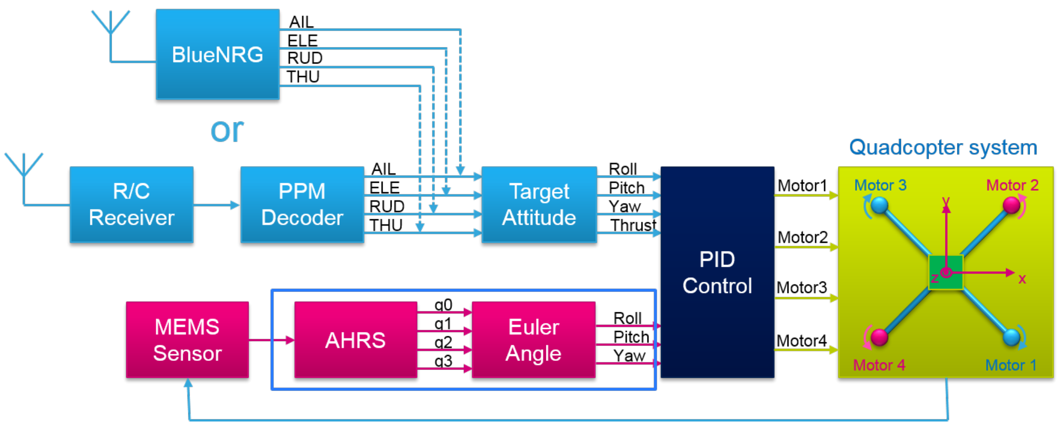


3.4 Battery voltage monitoring

The ADC1 channel 9 (PB1) is connected to V_{BAT} through a resistor partition to monitor the battery level for telemetry or any other action to be taken (stop the motor, sound alarm buzzer, etc.).

3.5 AHRS

Figure 15. **STSW-FCU001 firmware project architecture: AHRS module**



Attitude and heading reference system (AHRS) is the key algorithm for a UAV system. The algorithm chosen for the STSW-FCU001 firmware is the complementary filter shown below.

Figure 16. STSW-FCU001: complementary filter

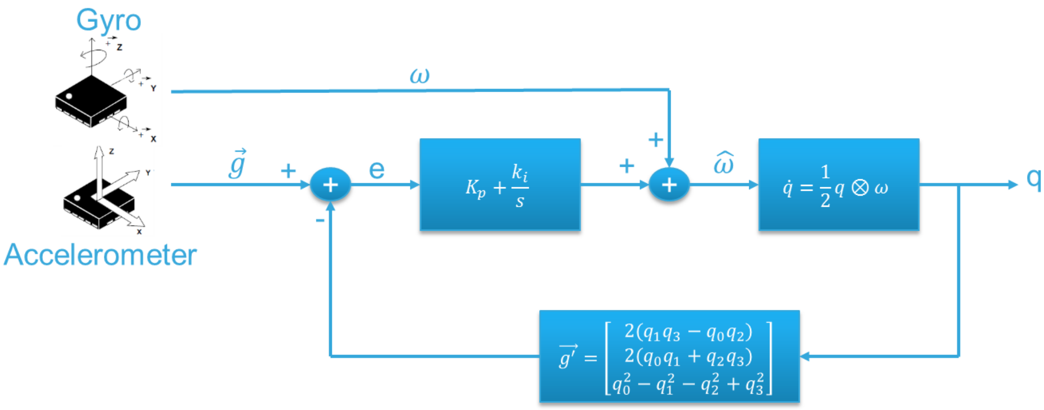


Table 5. Gyroscope and accelerometer sensor functions

Accelerometer	Gyroscope
Independent altitude calculation	
Influenced by high-frequency noise	Low-frequency offset
Inaccurate short-term results vs. long-term accurate results	Accurate short-term results vs. long-term offset
Calibrates the altitude and the gravity measurement is used to estimate the gyroscope drift on pitch and roll (1)	Gets the altitude (1)

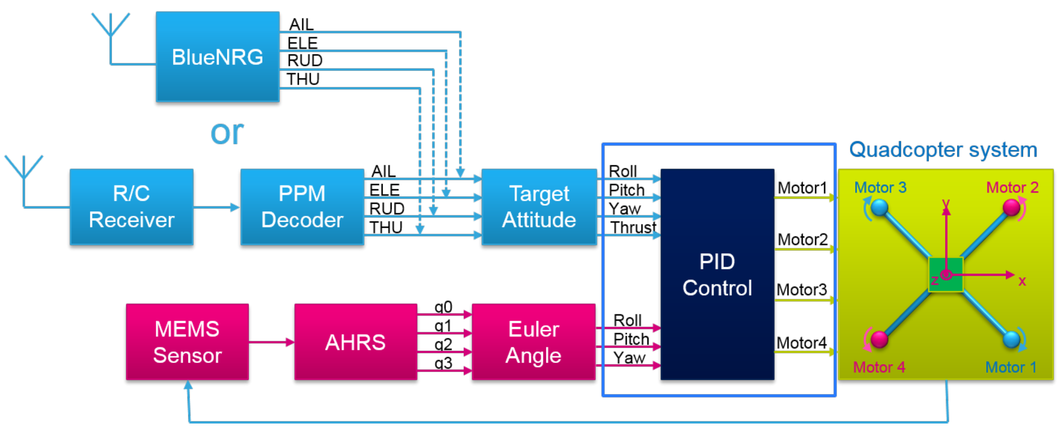
1. *Based on Mahoney filter.*

The accelerometer and gyroscope raw data are transferred to the AHRS algorithm in the **ahrs.c** file by the function `ahrs_fusion_ag(&acc_ahrs, &gyro_ahrs, &ahrs);`

To perform the drone stabilization algorithm, AHRS quaternion data must be translated to the Euler angle in the **quaternion.c** file by the function `QuaternionToEuler(&ahrs.q, &euler_ahrs);`.

3.6 Flight PID control

Figure 17. STSW-FCU001 firmware project architecture: PID control module



The PID control stabilizes the algorithm via a function in the **flight_control.c** file.

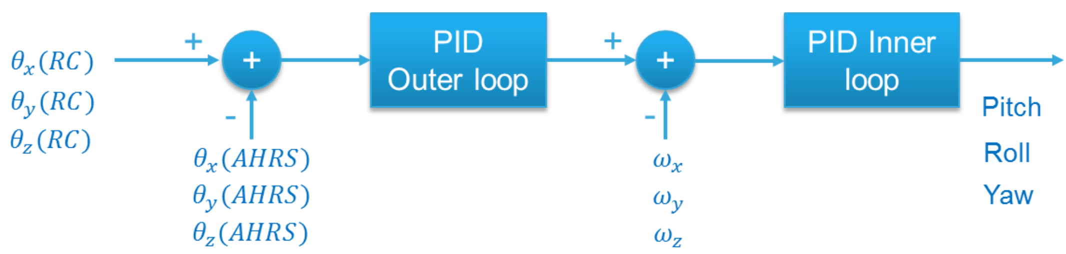
The control is performed in two different stages:

1. *PID Outer loop*: by comparing the target Euler angles given by the AHRS Remocon and the Euler angles, it controls the inclination angle via the function:
   `FlightControlPID_OuterLoop(&euler_rc_fil, &euler_ahrs, &ahrs, &pid);`
2. *PID Inner loop*: tracks the angular rate via the function:
   `FlightControlPID_innerLoop(&euler_rc_fil, &gyro_rad, &ahrs, &pid, &motor_pwm);`

To make the system stable when creating a mathematical model for a drone, it is necessary to add an inner loop to the outer loop.

Both **Inner** and **Outer Flight control PID loop** functions are located in the **flight_control.c** file and called in the **main.c** loop.

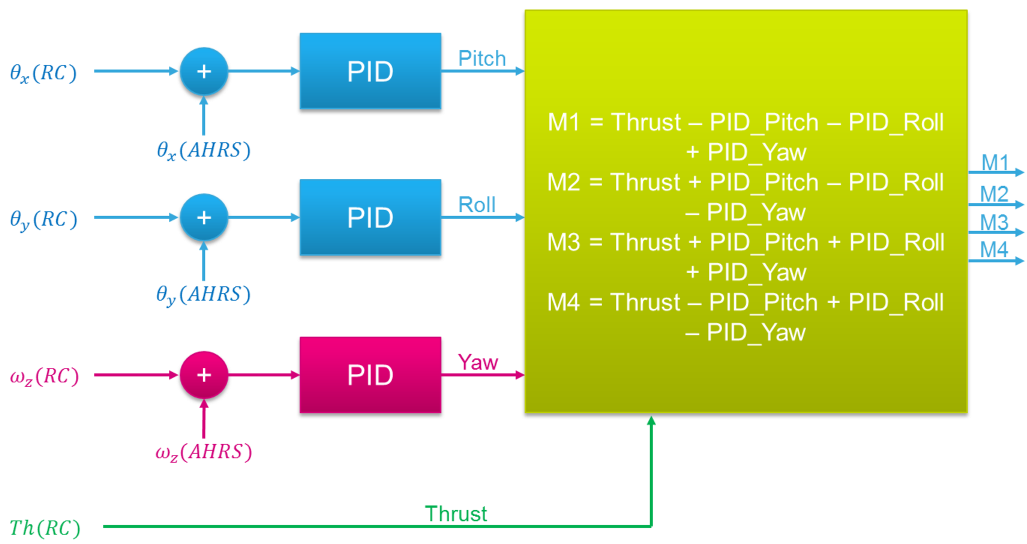
Figure 18. **PID control stages: outer and inner loops**



The PID output is given by the speed values of the 4 quadcopter motors (PWM hardware signal for ESC or MOSFET driving):

```
motor_pwm->motor1_pwm = motor_thr - pid->x_s2 - pid->y_s2 + pid->z_s2 + MOTOR_OFF1;
motor_pwm->motor2_pwm = motor_thr + pid->x_s2 - pid->y_s2 - pid->z_s2 + MOTOR_OFF2;
motor_pwm->motor3_pwm = motor_thr + pid->x_s2 + pid->y_s2 + pid->z_s2 + MOTOR_OFF3;
motor_pwm->motor4_pwm = motor_thr - pid->x_s2 + pid->y_s2 - pid->z_s2 + MOTOR_OFF4;
```

Figure 19. **PID control output converted to PWM signals for the ESC**



With an external ESC, a few modifications may be necessary to ensure that the PWM output signal for each motor is compatible with the ESC input signal.

Most ESCs accept an input signal with frequency varying from 50 to 400 Hz, and the Ton time of the PWM is proportional to the motor speed.

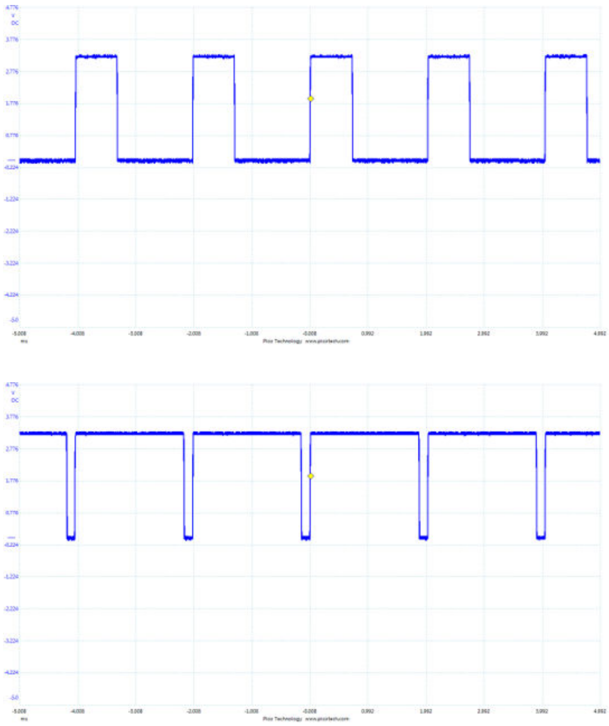
You must also check the minimum Ton time to arm the ESC (when not armed usually the ESC emits a particular alarm beep).

It might be necessary to adjust the following variables:

- `MIN_THR` (**flight_control.h**)
- `MOTOR_MIN_PWM_VALUE` (**motor.h**)
- `MOTOR_MAX_PWM_VALUE` (**motor.h**)
- `htim4.Init.Prescaler` (**main.c**)

Note: Refer to UM2311, freely available at www.st.com, for details on the hardware modification to connect the STEVAL-FCU001V1 to external ESCs (i.e. STEVAL-ESC001V1 or STEVAL-ESC002V1).

Figure 20. Motor control connector PWM signal on pin 2 (or P1, P2, P4, P5) with THR to min. value (on the left) and to max. value (on the right)



3.7 Main.c file

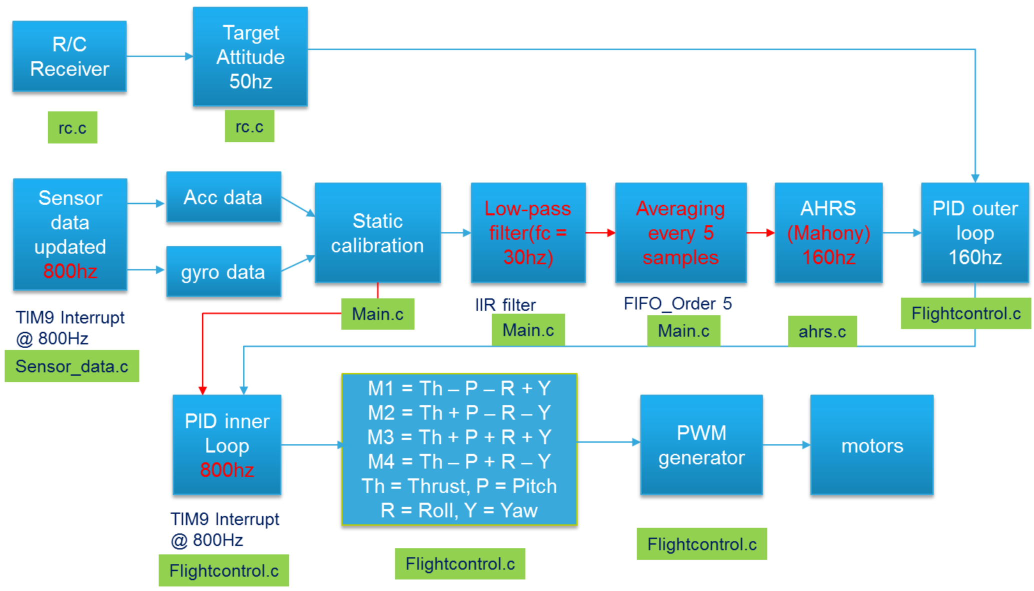
The **main.c** file performs the following operations:

- MCU configuration
- Reset of all peripherals
- Initialization of the Flash interface and the Systick
- Configuration of the system clock
- Initialization all configured peripherals (TIM2, TIM4, TIM9, ADC1, SPI1, SPI2, UART, GPIO)
- Initialization of the on-board LED
- Configuration and disabling of all the chip select pins for SPI sensors
- Initialization and enabling the available sensors on SPI
- Initialization of:
  - Settings for the 6-axis MEMS accelerometer
  - Settings for the 6-axis MEMS gyroscope
  - Remote control
  - Timers
    - TIM2 for external Remocon RF receiver PWM input (PPM signal in)
    - TIM4 for motors PWM output
    - General purpose TIM9 50 Hz
  - PID
- Setting of motor PWM to zero
- Setting timer to 5 ms interval
- AHRS update, quaternion and real gyroscope data stored in the *ahrs* variable
- Calculation of quadcopter Euler angle
- Target Euler angle from remote control
- Execution of flight control PID inner loop
- Execution of flight control PID outer loop and update of motor PWM output signals

The figure below shows the main.c file global flow with the relevant functions and key parameters defined to guarantee stabilization and anti-drifting (settings have been identified during test flight characterization).

Note: Blocks and lines in red are key points for stabilization and anti-drifting.

Figure 21. STSW-FCU001 firmware project architecture: main.c file global flow and related functions

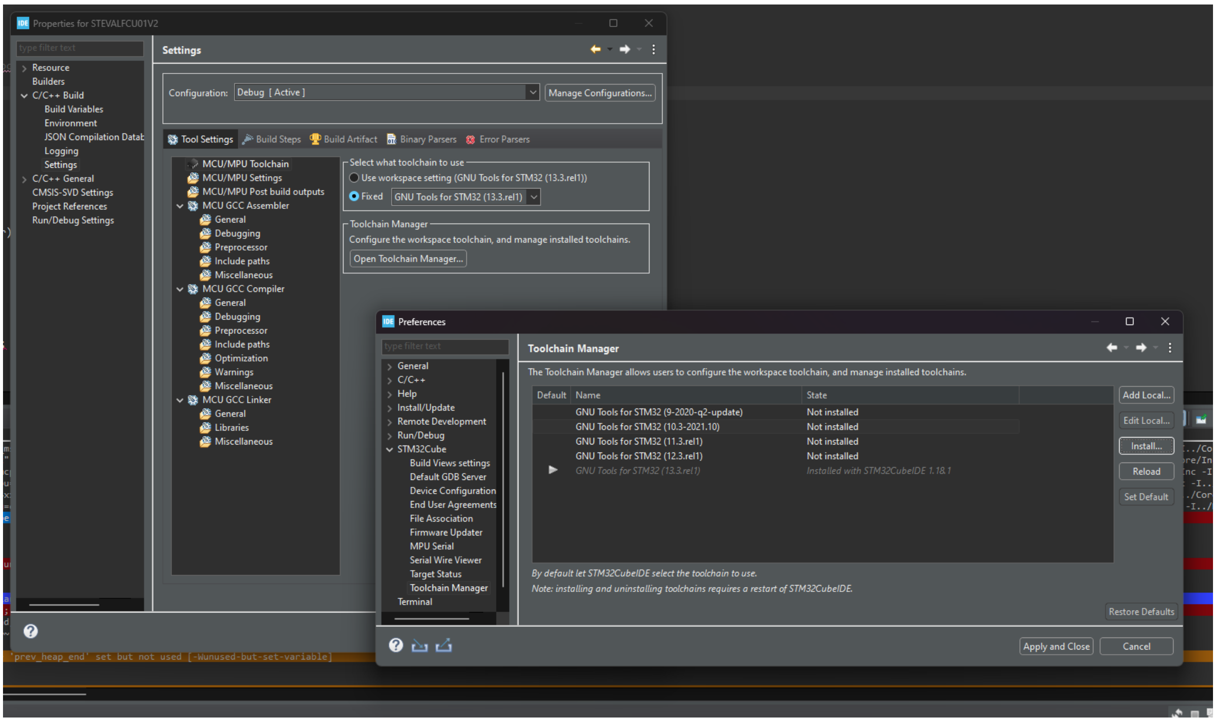


4 Tool chains

The STM32Cube expansion framework supports IAR Embedded Workbench for ARM® (IAR-EWARM), KEIL RealView microcontroller development Kit (MDK-ARM-STM32) and System Workbench for STM32 (SW4STM32) to build applications with the STEVAL-FCU001V1 evaluation board.

Meanwhile, the firmware for STEVAL-FCU001V2 supports only STM32CubeIDE and Tool Chain 10.3-2021. If you have installed a most recent version you have to fix it as in Figure 22.

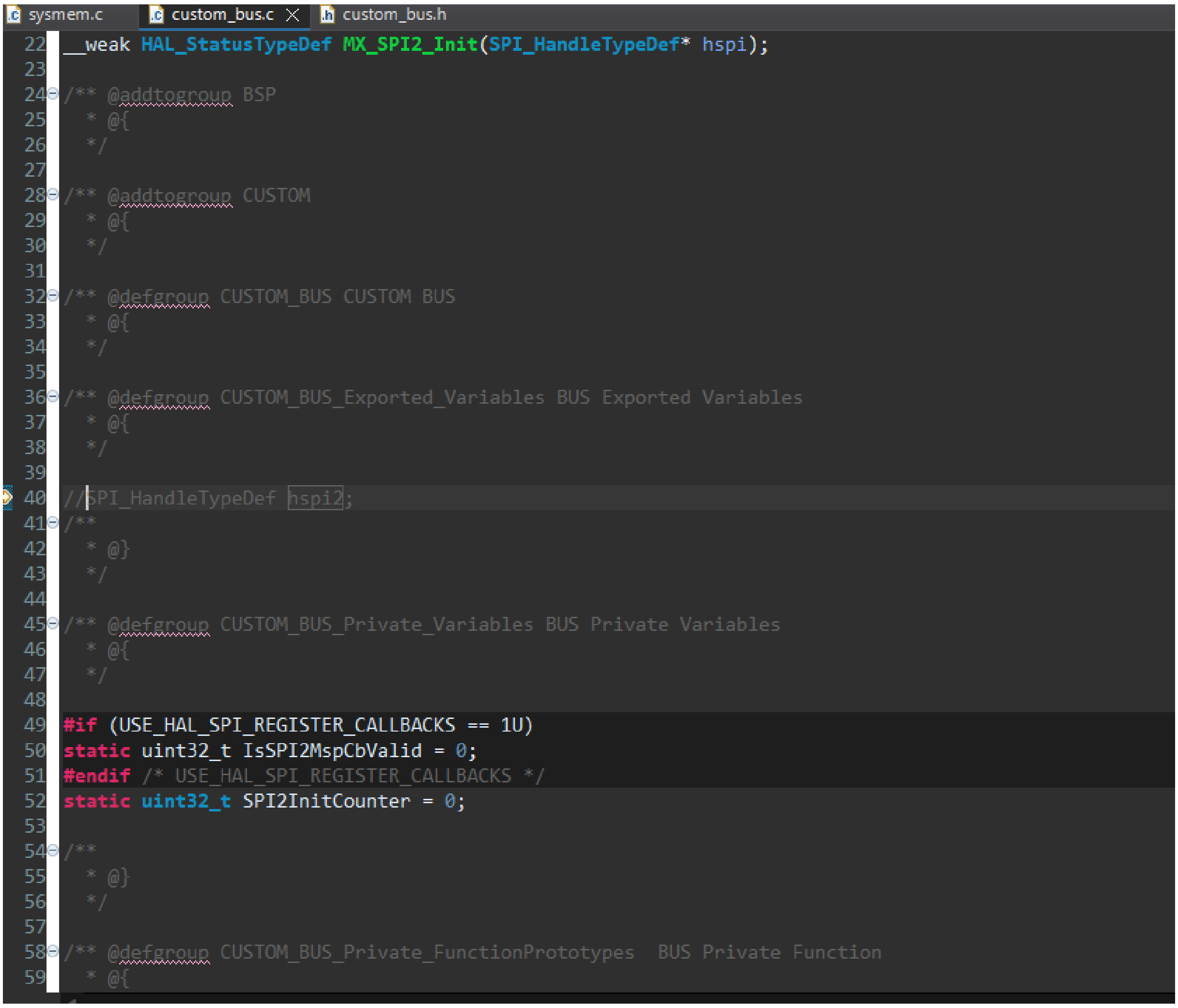
Figure 22. Firmware fix



Note: *if firmware is not present on your PC, you have to download and install it.*

Once firware is installed, open the file "custom_bus.h" at row 40 and comment SPI_HandleTypeDef hspi2 as in Figure 23.

Figure 23. **Custom bush**



When establishing your IDE workspace, ensure that the folder installation path is not too structured to avoid toolchain errors.

Further firmware development can be started using the reference projects included and the SWD as mandatory debug interface.

Note: The three environments have to be used with ST-LINK.

Revision history

Table 6. Document revision history

Date	Version	Changes
06-Dec-2017	1	Initial release.
15-Jan-2018	2	Updated Updated Figure 14. STSW-FCU001: sensor calibration.
11-Feb-2019	3	Updated *Section 3 STSW-FCU001 firmware project modules, Section 3.1 R/C receiver and PPM decoder, Figure 10. STSW-FCU001 firmware project architecture: target attitude module, Figure 12. STSW-FCU001 firmware project architecture: MEMs sensor module, Figure 16. STSWFCU001 firmware project architecture: AHRS module* and *Section 3.6 Flight PID control.*
07-Aug-2024	4	Updated Section Introduction, Section 2.1: Overview, Section 2.2: Architecture, Section 2.3: Folder structure, Section 3.1: R/C receiver and PPM decoder, Section 3.3: MEMS sensors and Section 4: Tool chains.
12-Sep-2025	5	Updated Section 4: Tool chains.

Contents

List of tables

List of figures