life.augmented

Robotics and Drones with STM32

Giuseppe Messina

Arcangelo Bruna

Giuseppe Spampinato

Agenda

- # Overview
- # Kondo Robot
- # Nao
- # Will-E
- # Drones
- # Future Projects

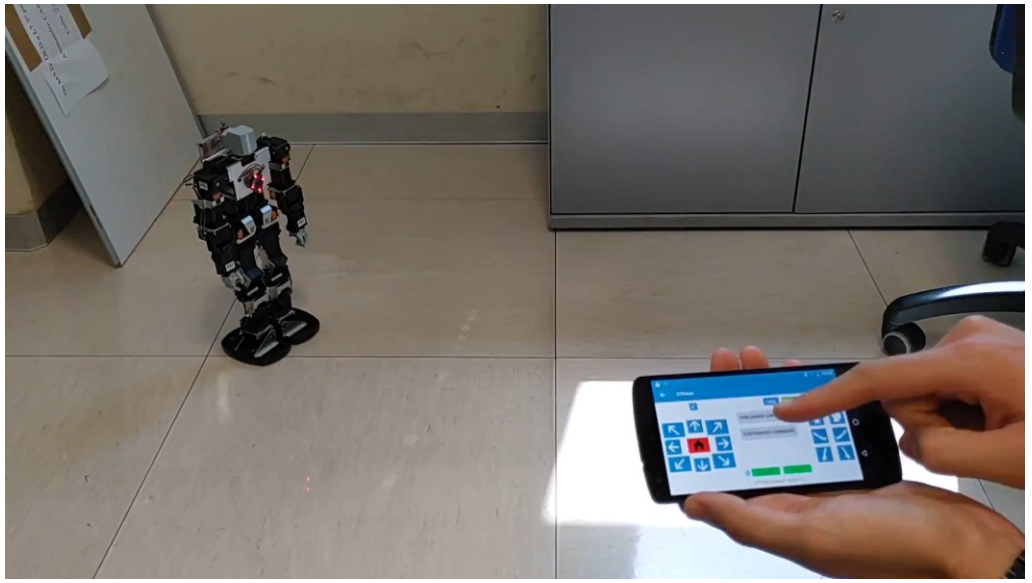
life.augmented

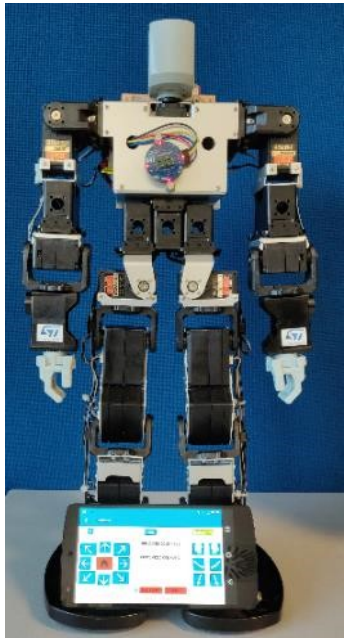
Overview

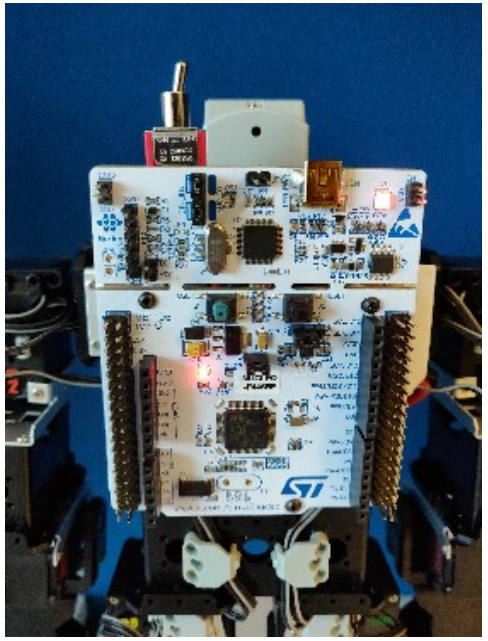
- The aim of Robotics Activities@ST is to integrate incrementally ST technologies and algorithms developed and to be developed into robotics environments.
- The current scenarios we are exploring regards
  - Indoor navigation and Simultaneous Localization and Mapping (SLAM)
  - Human Robot Interaction
  - Robot Assistance
- Objective: To be inline with AI/ML algorithms developed so far in ST.

Kondo Robot with ST-Nucleo

- Kondo-Bot
  - Integrated KWS with SensorTile and BlueCoin
  - Replaced Renesas based main board for motor control with STM32F44RE Nucleo board.
  - Tested STM32-Kondo-Bot with associated Android App.
  - Presented at **Maker Faire Rome 2019**





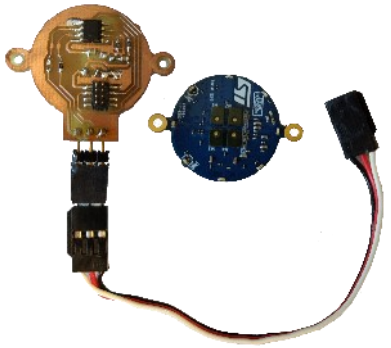


Kondo-KHR-3HV

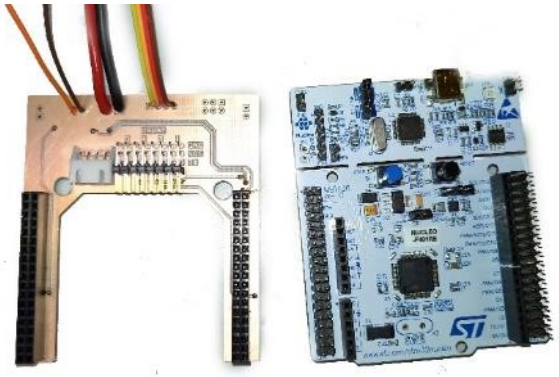
life.augmented

Customizations

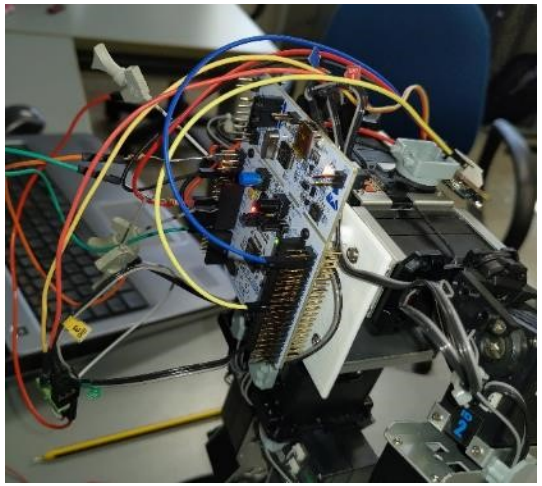
- Two expansion board have been developed to connect the BlueCoin and the Nucleo
  - For the BlueCoin we have used an expansion that allow to connect UART and Power from the Nucleo.
  - For the Nucleo board we have developed an expansion allowing the connection of 6 UART, Power and Battery check
- It could be done also without this expansions but it's less attractive



Bluecoin and Transceiver



Nucleo and Motor Connector

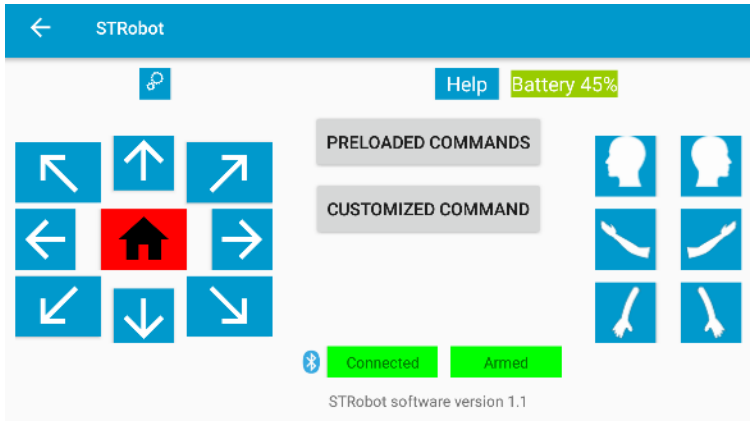


Without Customized Expansion Boards

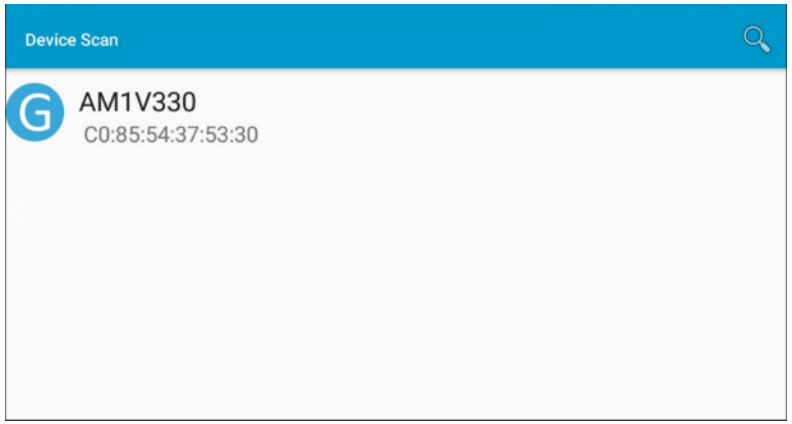
Android app

Android app

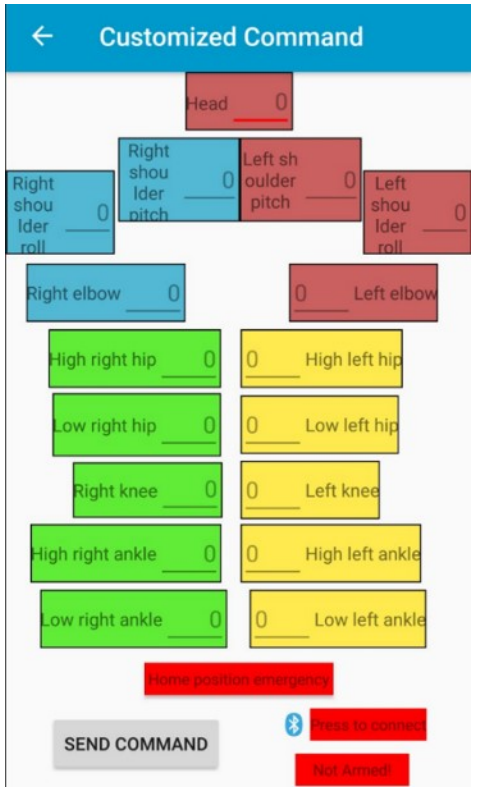
Main activity



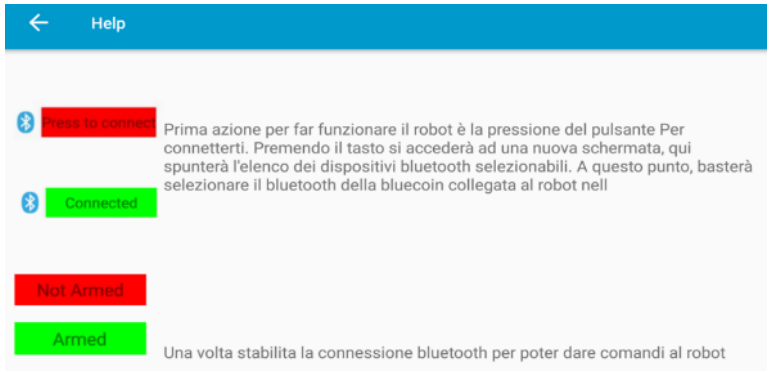
Device scan



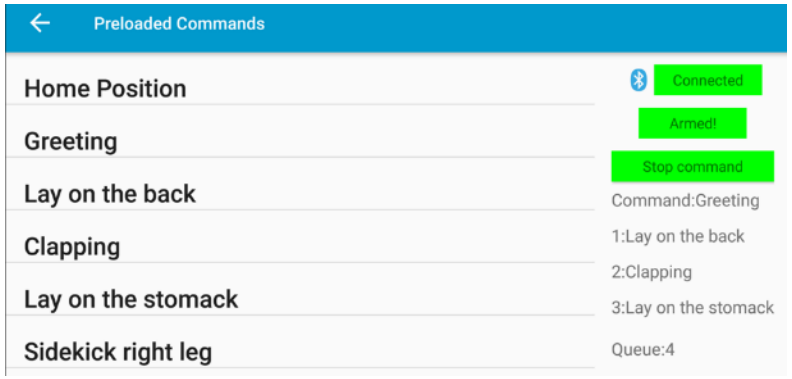
Customized Command



Help



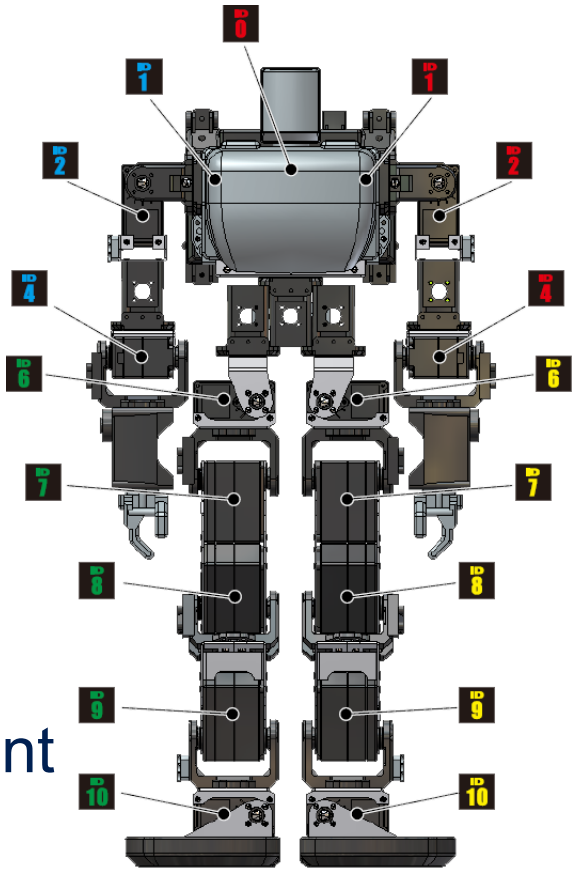
Preloaded commands

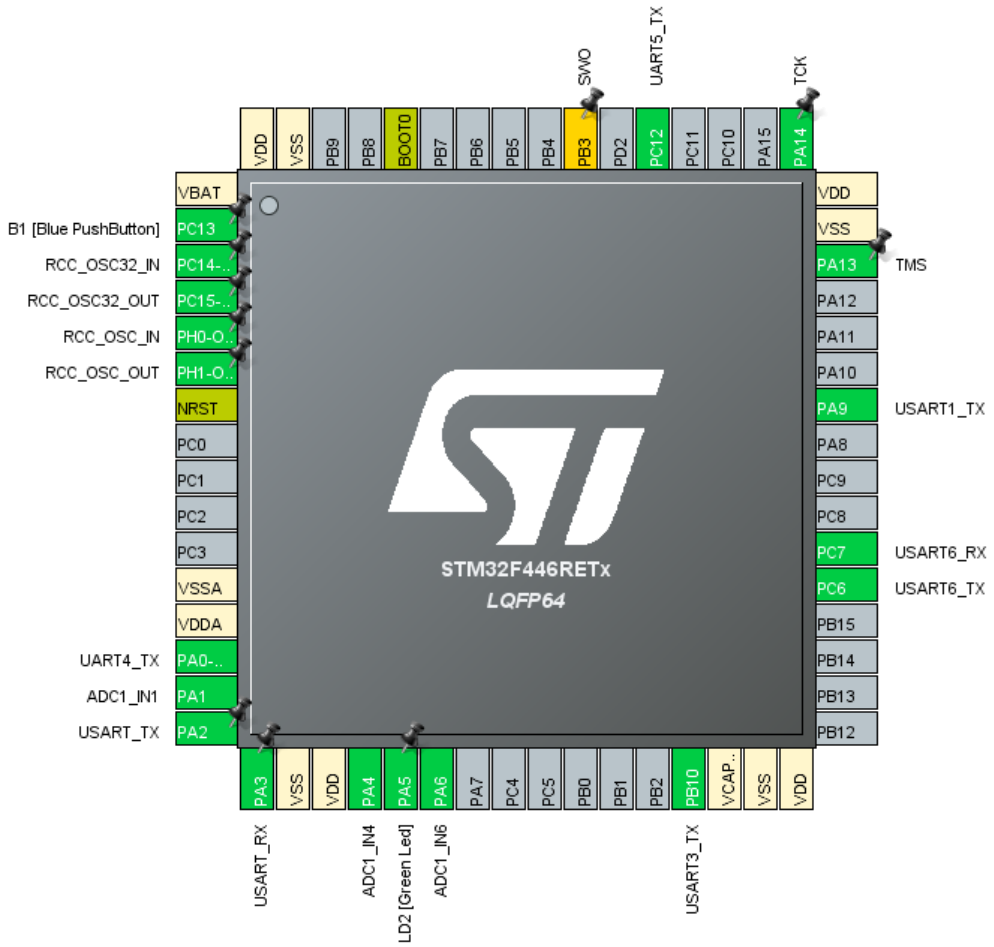


life.augmented

STM32CubeMx

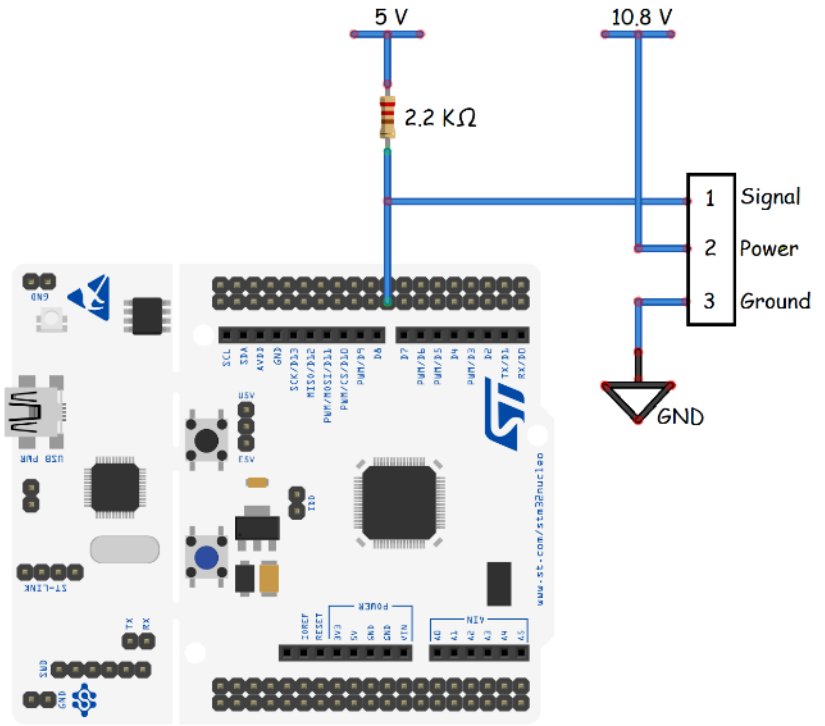
- Peripheral initialization:
  - **USART1 → red line**
  - **USART3 → blue line**
  - **UART4 → yellow line**
  - **UART5 → green line**
  - USART2 → USB debug
  - USART6 → Bluecoin communication
- ADC1 → battery management





ICS3.5 Communication Standard

- Max 1.25 bps high speed communication
- Max 32 servomotors connected on the same line
- Single data line used for serial communication (Half-Duplex)
- Multi-drop connection
- Signal level 5 V TTL
- ID setting
- Reading and writing parameters
- Position setting

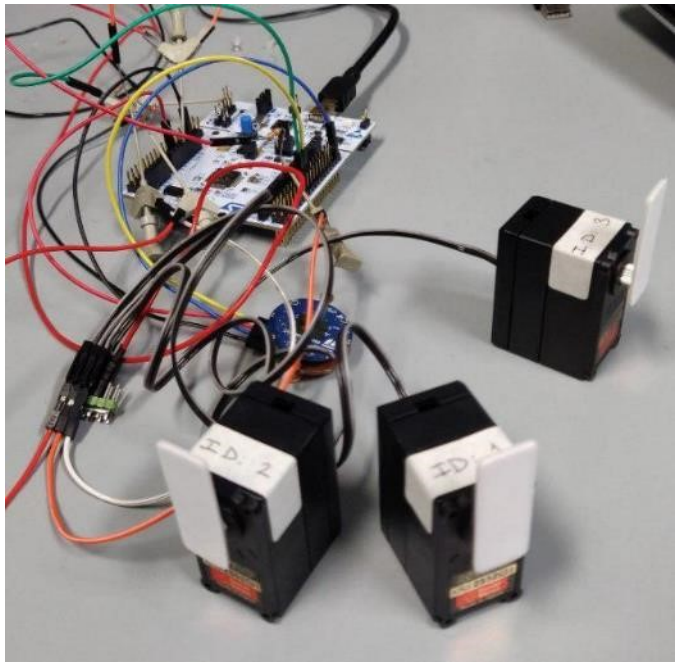


CMD	SC	DATA
Command header + ID	Subcommand	Data

Firmware implementation

- ICS.h:
  - ics_get_id()
  - ics_set_id()
  - ics_get_speed()
  - ics_set_speed()
  - ics_get_stretch()
  - ics_set_stretch()
  - ics_get_current()
  - ics_pos()
  - ics_free()

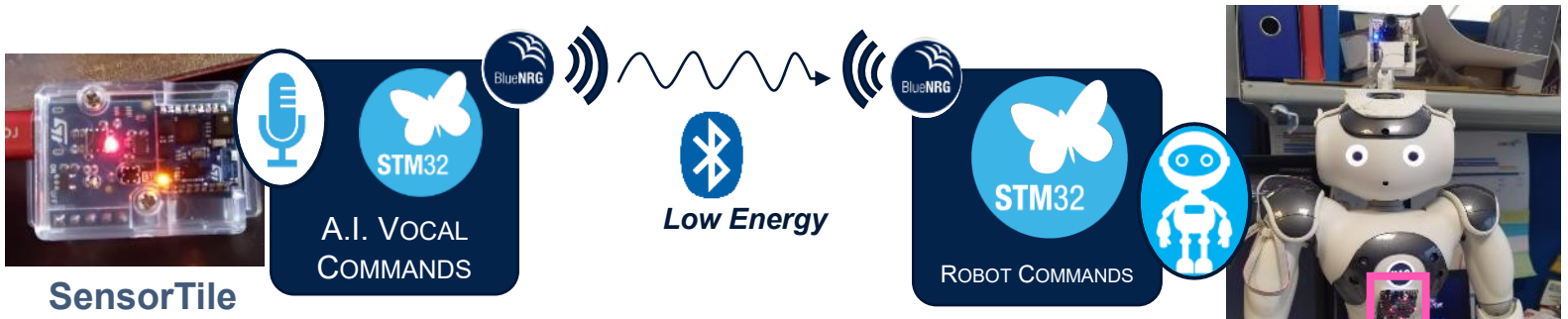
```
uint16_t ics_pos(ICSData *r, uint16_t id, uint16_t pos, UART_HandleTypeDef *huart)
{
    uint16_t byte_in = 3;
    uint16_t byte_out = 4;

    if(id>31)
    {
        snprintf(r->error,128,"ERROR: Invalid servo ID (0-31)");
        return -1;
    }

    if(pos>16383)
    {
        snprintf(r->error,128,"ERROR: Invalid servo position (0-16383)");
        return -1;
    }

    //build command
    r->swap[0] = id | ICS_CMD_POS;    //Command
    r->swap[1] = (pos >> 7) & 0x7F;    //POS_H: high 7 bits of pos
    r->swap[2] = pos & 0x7F;    //POS_L: low 7 bits of pos

    //send command
    HAL_UART_Transmit(huart, r->swap, byte_in, ICS_POS_TIMEOUT);

    for(int i=0; i < byte_in; i++)
        r->swap[i] = 0;

    //wait for the the response
    HAL_UART_Receive(huart, r->swap, byte_out, ICS_POS_TIMEOUT);

    //return the position
    return (((r->swap[2] & 0x7F) << 7) | (r->swap[3] & 0x7F));
}
```

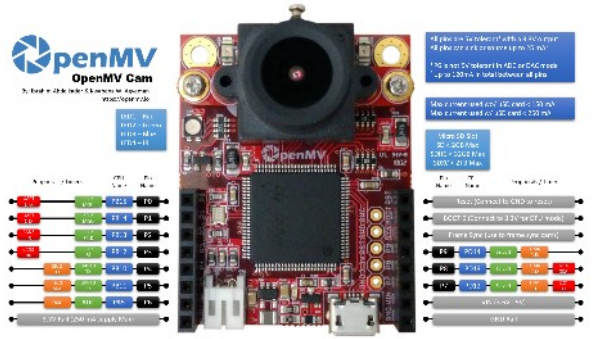


life.augmented

Nao Robot and ST tecnologies

- NAO
  - Based on Softbank Nao Robot, our project aims is to integrate as much as possible ST devices into the existing environment;
  - Integrated **TMOS sensor for Human Body temperature measurement**
  - The **OpenMv camera has been connected to Nao** enable Face and Smile detection and to perform a reaction simulated by the robot
  - **Vocal Command Recognition to pilot Nao movements**





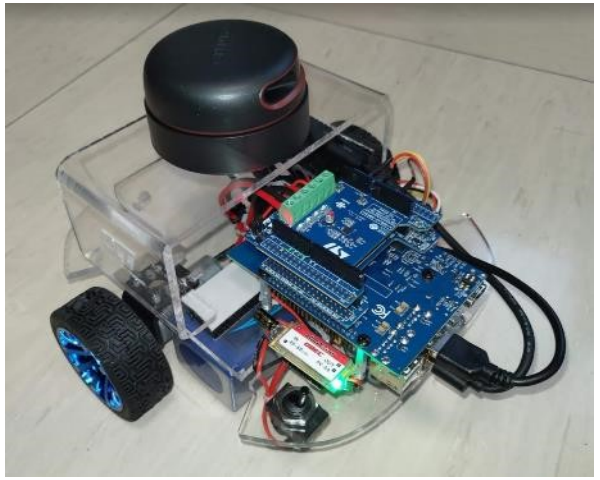
life.augmented

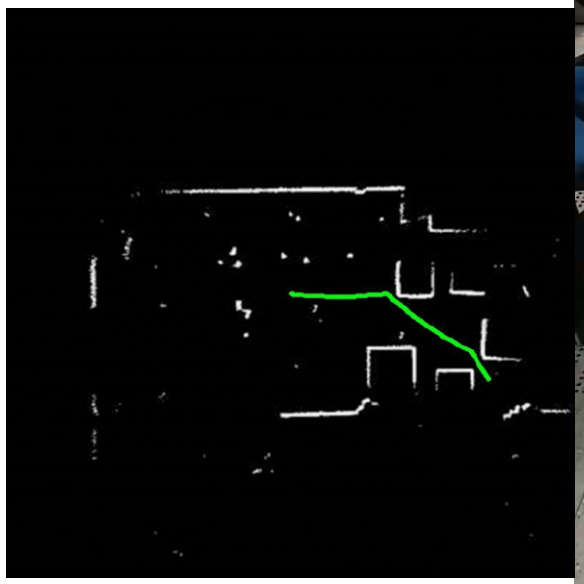
Wheeled Environment Robot (Will-E)

- Will-E
  - Demo application on 2D-SLAM using commercial LiDAR data has been finalized and integrated on Will-E. It works on Odroid XU4 Platform coupled with a Nucleo STM32F401 Board (version 1), and on STM32MP157C-DK2 (version 2)

Version 1

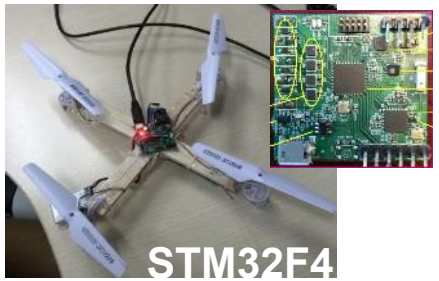
Version 2

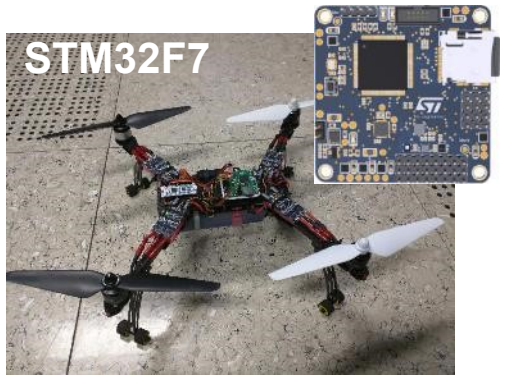


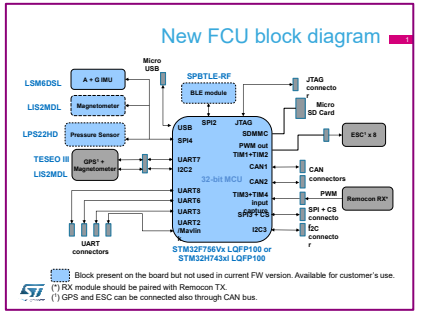


Final 2D-SLAM

Drones







Mobile World Congress
7/1/2016

Maiker Faire Rome 2017
12/2/2017

CES 2016
1/3/2016

CES 2017
1/3/2017

CES 2018
1/3/2018

STM32 Summit
4/24/2018

Maiker Faire Rome 2019
12/10/2019

2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 **2021**

Toy Drone Korea Project

Toy Drone China Project

Consumer Drone China Project

Consumer Drone PX4 Project

STEVALFCU001-v1

New STM32F7 Board

STEVAL-DRONE01-V1

STEVALDRONE01-V1

STEVALFCU001-V2

STEVALFCU001-v1

Completed Frozen

life.augmented

Basics of multicopters
Flying with STM32

Giuseppe Messina

Agenda

- # Basics of multicopters
- # STEVAL-FCU001V1 Hardware
- # STEVAL-FCU001V1 Software

life.augmented

Basics of multicopters

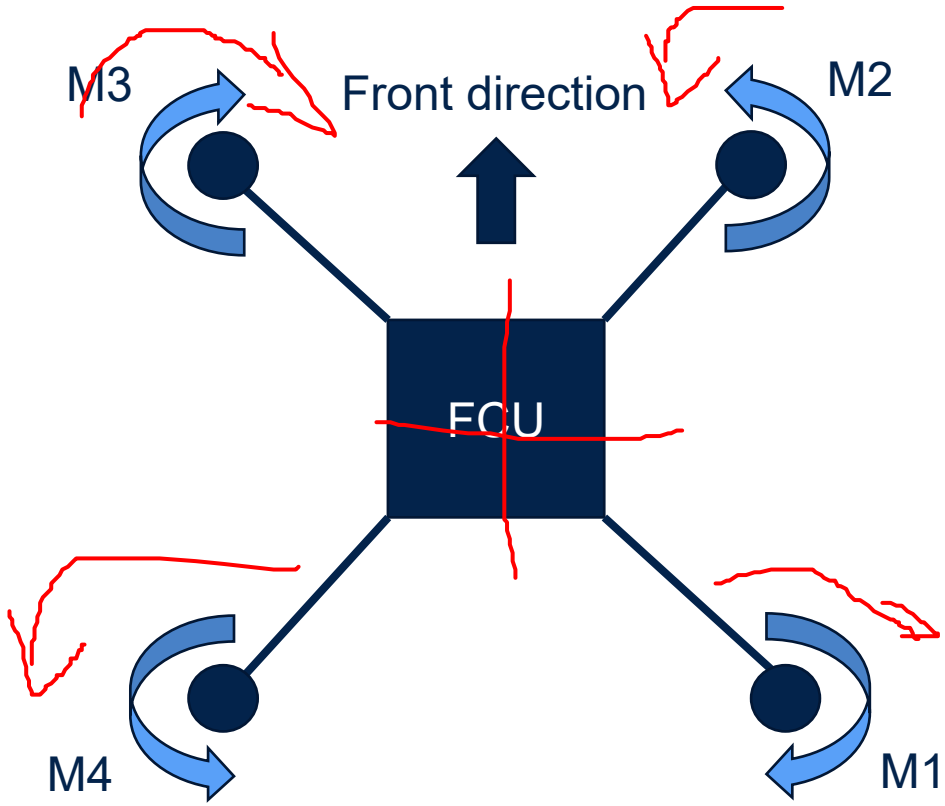
Definition of UAS, drone, multicopter

An *UAS (Unmanned Aircraft System)*, sometimes called a *drone*, is an aircraft without a human pilot on-board; instead, the UAS is controlled from an operator on the ground.

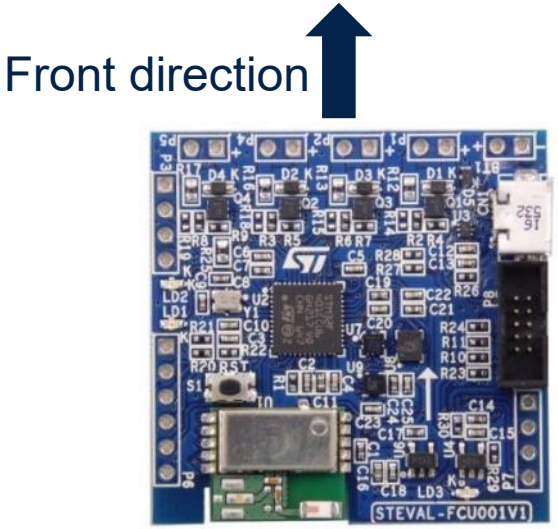
A *multicopter* or *multirotor* is a rotorcraft with more than two rotors. While single (and double) rotor helicopters are more complex to build because they need variable pitch rotors whose pitch varies as the blade rotates for flight stability and control, multicopters use fixed-pitch blades. Control of the multicopter is achieved by varying the relative speed of each rotor to change the thrust and torque produced by each.

A *quadcopter* is a 4-rotor multicopter. In this document we will also use the more popular term, *drone*.

Quadcopter basics: flight control dynamics

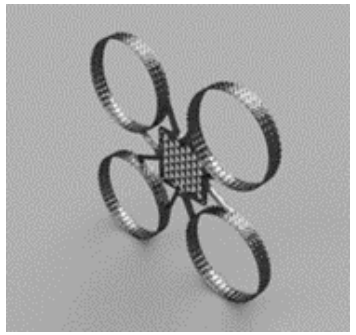


Each rotor produces both a thrust and a torque about its center of rotation. If all rotors are spinning at the same angular velocity, with M1 and M3 rotating CW and M2 and M4 rotating CCW, the angular acceleration about the yaw axis is exactly zero (so there is no need for a tail rotor like on conventional helicopters).



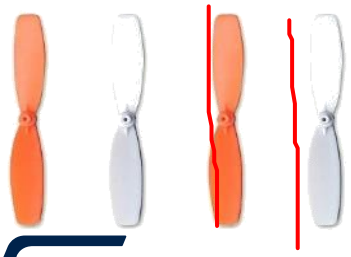
FCU: Flight Control Unit

Quadcopter basics



Frame: there are several frames of different size and materials available on the market. For simplest control algorithm the motors and propellers should be placed at equidistant intervals.

Motors: for small drones, *DC coreless motors* are usually used, while for bigger frames, *3-ph DC brushless* motors are preferred. The choice of the motor must be made together with the frame and propeller selection. Please be aware that some motors are designed to be used only in CW or CCW direction (especially DC brushed).



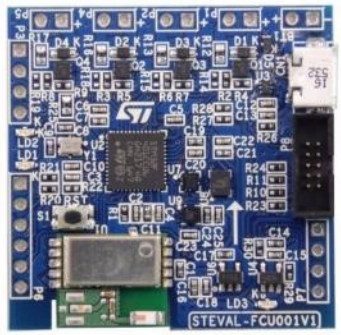
Propellers: are a type of fan that convert rotational motion into thrust. There is a wide choice with 2, 3, and 4 blades, different materials and shapes.

Remocon and RX receiver: it is possible control drones with Smartphones and/or remocons specific for multicopters, these are generally preferred from users because they offer better flight control sensitivity, meanwhile smartphone App is cheaper. Each remocon has an RX unit that must be connected to the FCU.

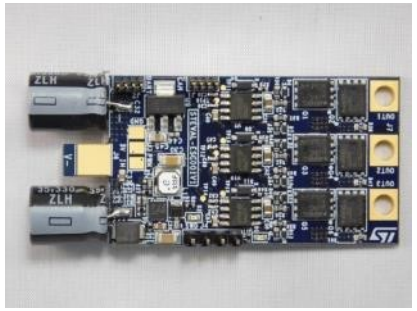
Battery: for multicopters *LiPo (Lithium Polymer) batteries* are commonly used. Depending on the size of the drone and motors, 1-, 2- or 3-cell batteries are selected, with different current capabilities and sizes. This kind of battery must be used with particular care because they can be damaged (and even explode) if overcharged or short-circuited.

life.augmented

Quadcopter basics: electronics



FCU (Flight Controller Unit): the main algorithm for flight control usually runs on a *32-bit microcontroller*, which also controls the speed of each motor by directly controlling them (*DC motor*) or by sending a signal to an external *ESC* (*Electronic Speed Controller for 3-ph DC brushless*). In order to stabilize the drone, data from *accelerometer* and *gyroscope* sensors are analyzed, while a *pressure sensor* may be used for altitude control.



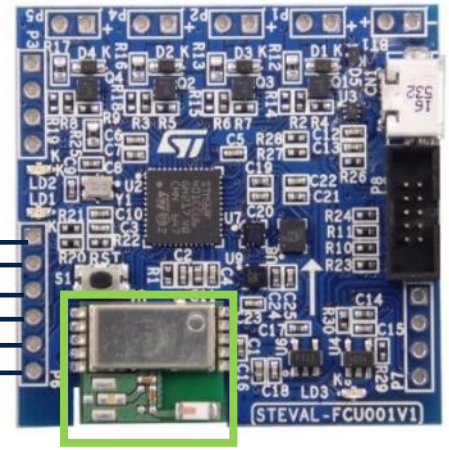
STEVAL-ESC001V1

ESC (Electronic speed controller): is used to drive a 3-ph DC brushless motor in sensorless mode. Generally, it receives an input signal up to 500Hz PWM from the FCU, with a pulse width varying from 1 ms (motor off) to 2 ms (motor full speed). The ESC is also connected directly to the battery, representing the Vbus for driving the motor.

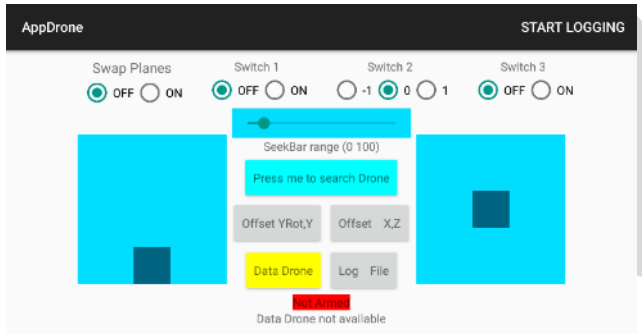
Most of them incorporate a so-called *BEC (battery eliminator circuit)*: basically, a DCDC converter to give the supply voltage to the FCU.

Quadcopter basics: Android App interface

Remote TX controller with proprietary RX modules (to be connected to the FCU).

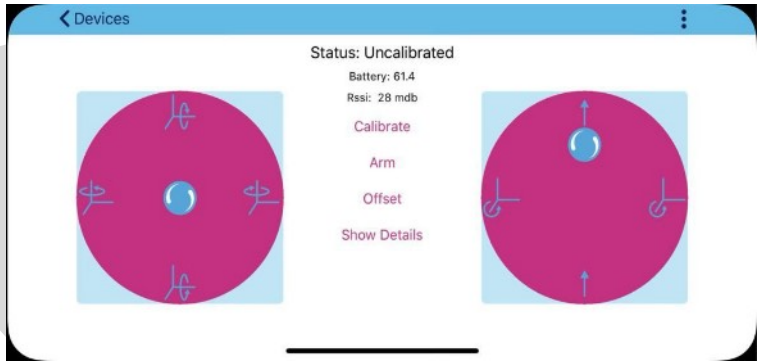


Bluetooth Low Energy module on the STEVAL-FCU001V1



App Android App with BLE connection has a layout similar to mechanical remocons commonly used.

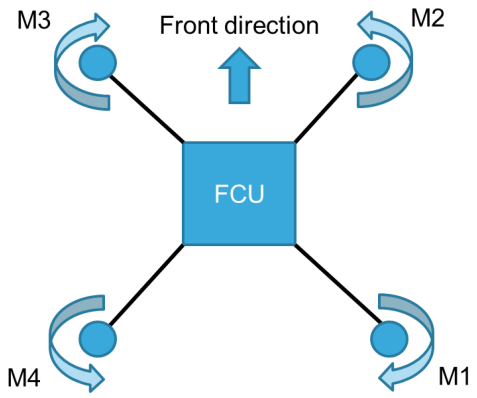
iPhone App is similar to the Android App.



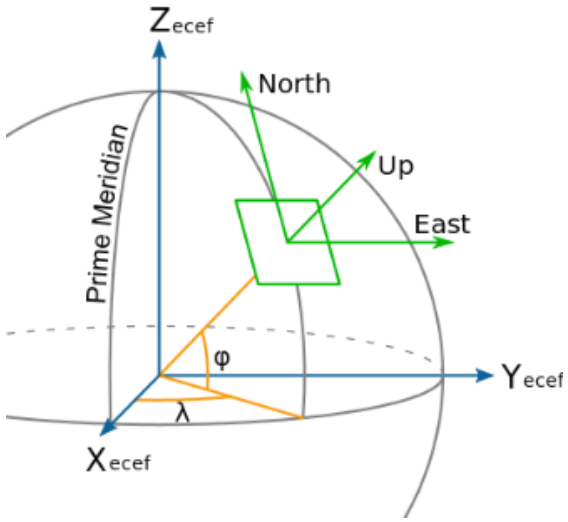
Note: currently Android and iPhone Apps are still in Beta version and can vary significantly over coming releases.

life.augmented

Quadcopter basics: flight control dynamics



- Coordinate System: ENU
- Frame Type: X-type
- 4 x Motor, rotation direction as shown
- Movement:
  - Forward / Backward (Pitch)
  - Left / Right (Roll)
  - Turn Left / Right (Yaw)
  - Up / Down

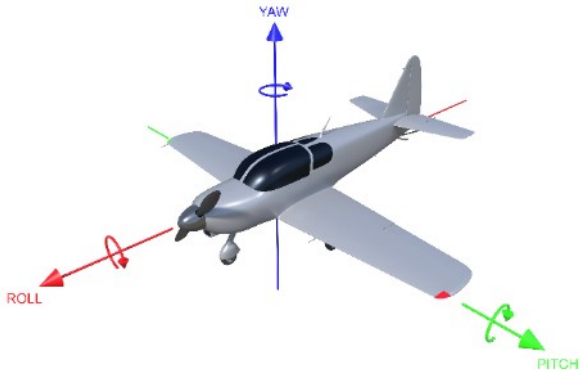


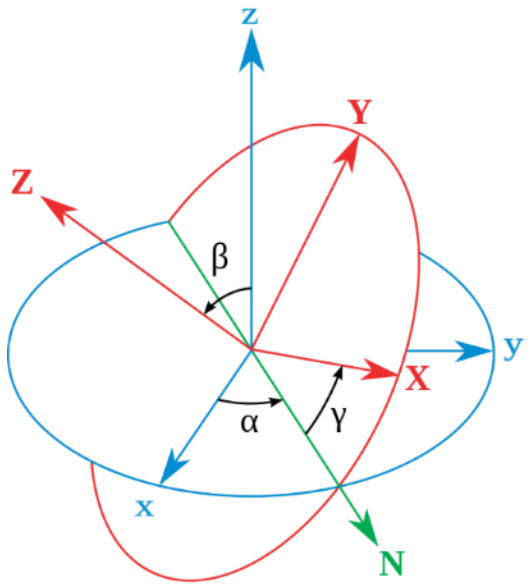
Motor / Action	1	2	3	4	Note
Pitch (x)	Slow (**Fast**)	**Fast** (Slow)	**Fast** (Slow)	Slow (**Fast**)	Move Backward (Forward)
Roll (y)	Slow (**Fast**)	Slow (**Fast**)	**Fast** (Slow)	**Fast** (Slow)	Move Right (Left)
Yaw (z)	**Fast** (Slow)	Slow (**Fast**)	**Fast** (Slow)	Slow (**Fast**)	Heading Left (Right)
Up / Down	**Fast** (Slow)	**Fast** (Slow)	**Fast** (Slow)	**Fast** (Slow)	Move Up (Down)

life.augmented

ENU: East North Up

Quadcopter basics: Euler angles





Euler angles – The fixed system (*xyz*) is in blue, while the rotating system (*XYZ*) is in red. In green are the line of nodes (*N).*
The Euler angles are:

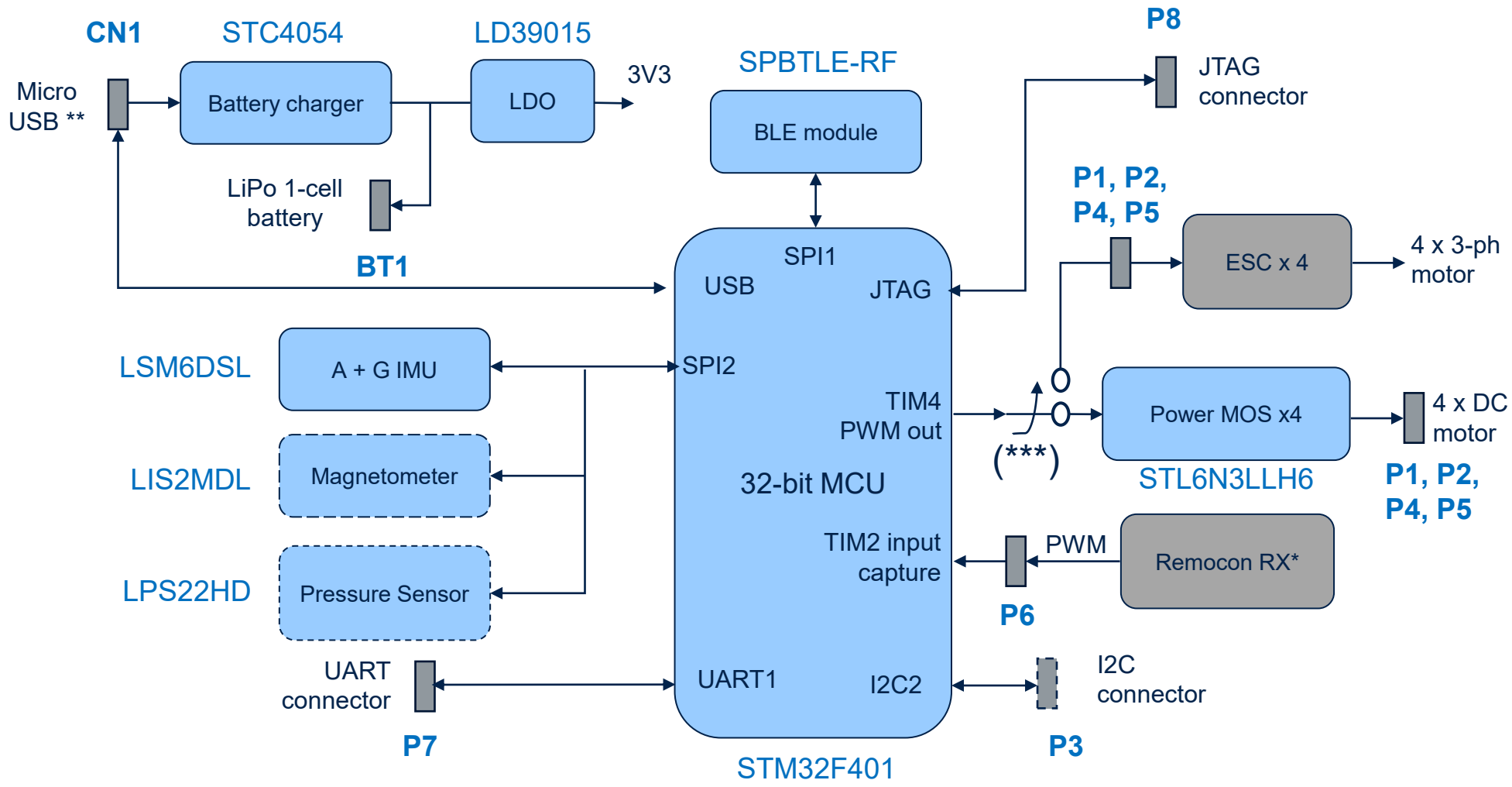
- α is the angle between x axis and line of nodes N.
- β is the angle between *z* and *Z* axis.
- γ is the angle between line of nodes N and X axis.

The Euler angles are very useful to represent the inclination of a quadcopter (or other flying vehicle) moving in space and will be used for the basic PID control algorithm.

life.augmented

STEVAL-FCU001V1 Hardware

STEVAL-FCU001V1 block diagram



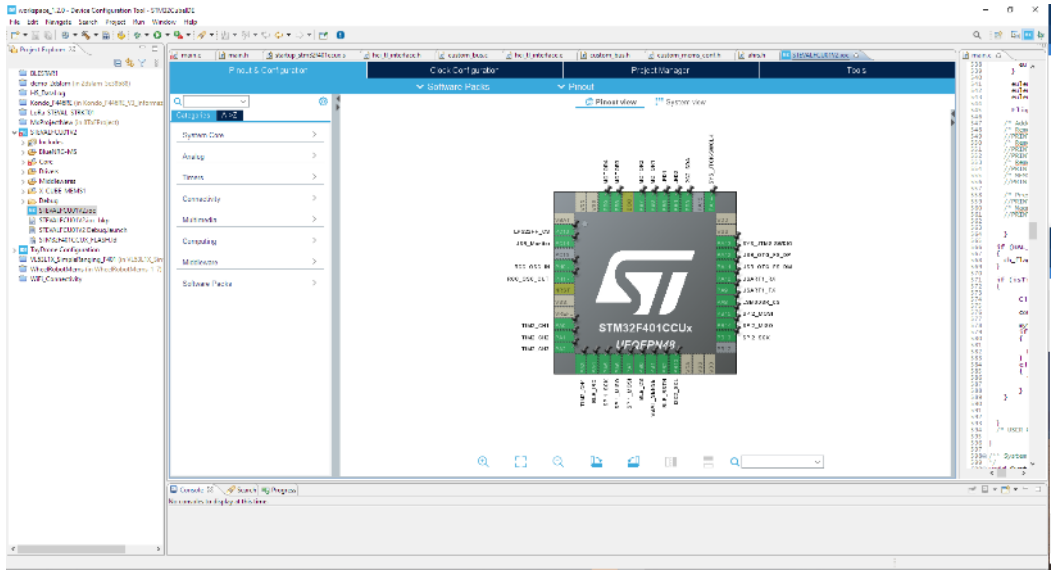
Block present on the board but not exploited in current FW version. HW available for customer use.

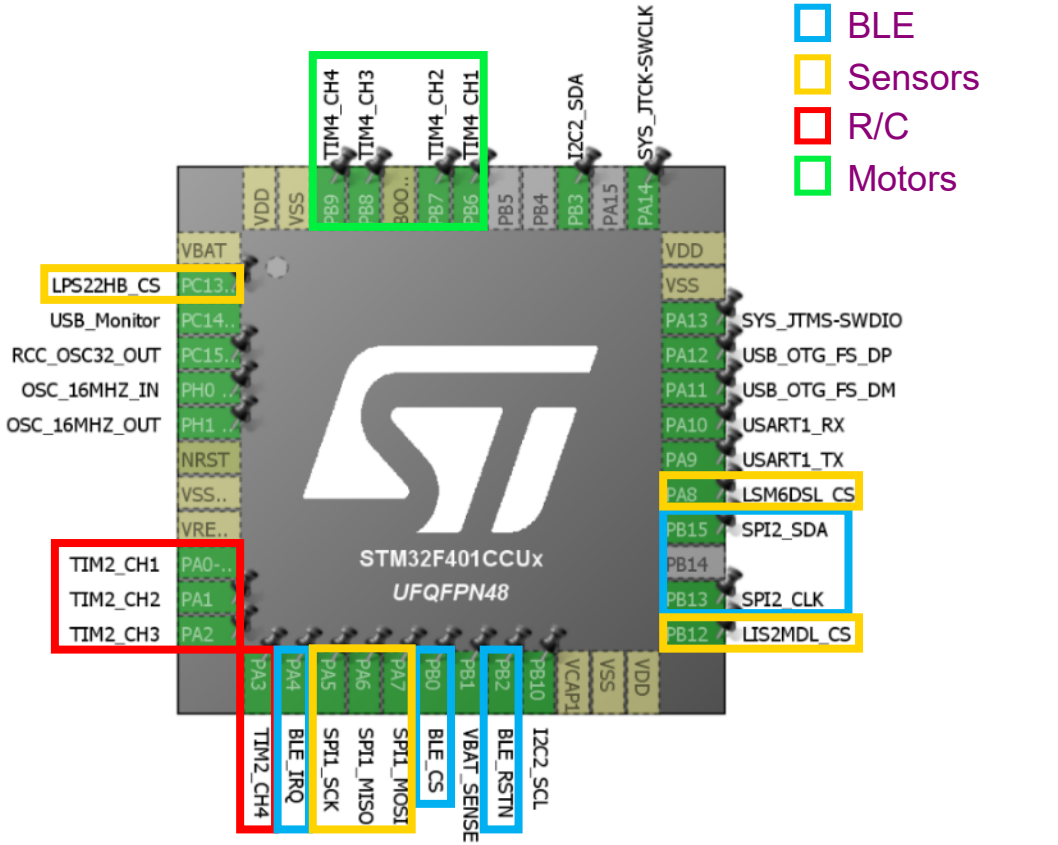
STM32F401 connections

STM32CubeMX:

- It is a graphic interface for configuring the STM32 peripherals and generating the relative source code
- It contains embedded STM32Cube libraries
- STM32CubeF4 is designed to be used with the STM32F4 family

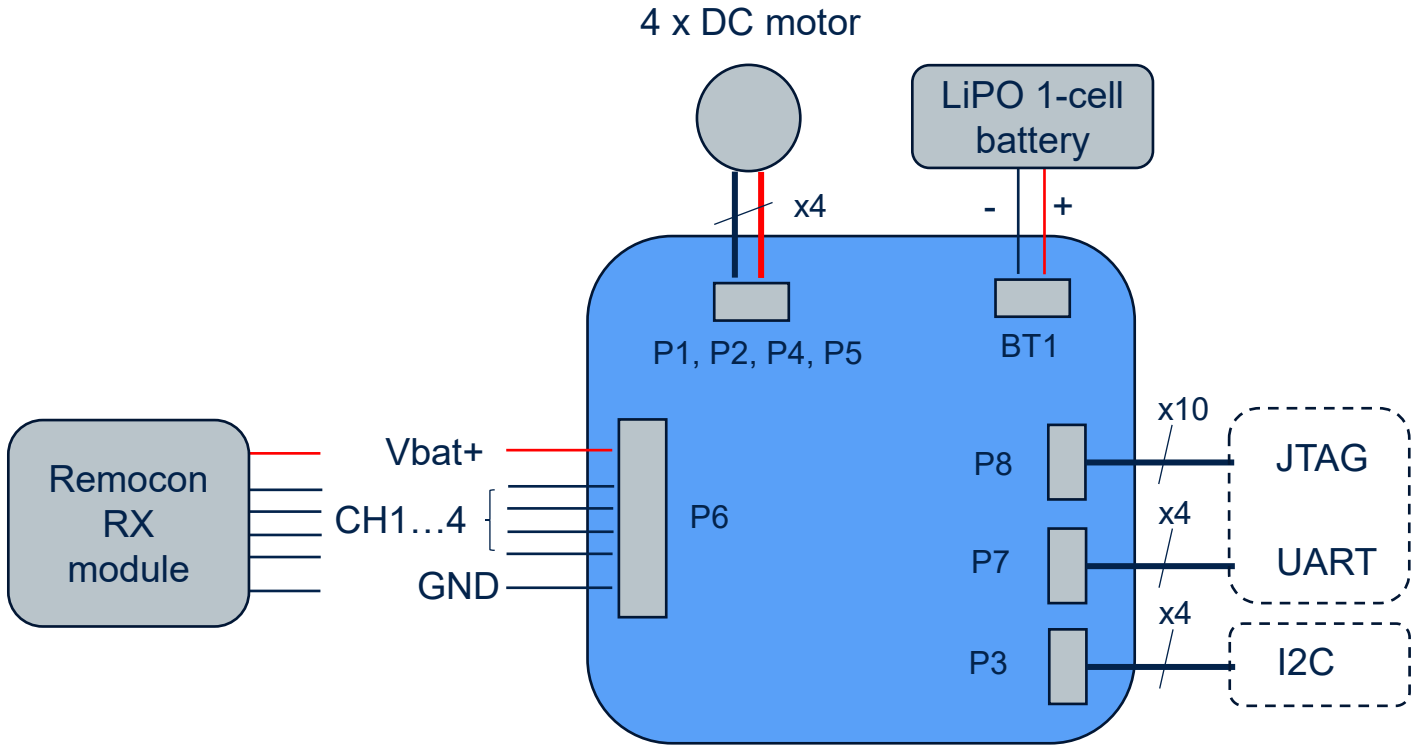
STM32CubeIDE:





life.augmented

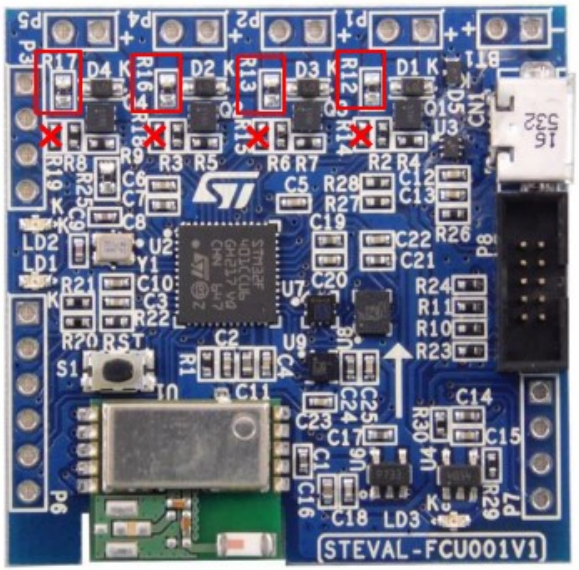
STEVAL-FCU001V1 application connection (3.7V DC motors)



STEVAL-FCU001V1
FCU eval board

Used only for debug purposes

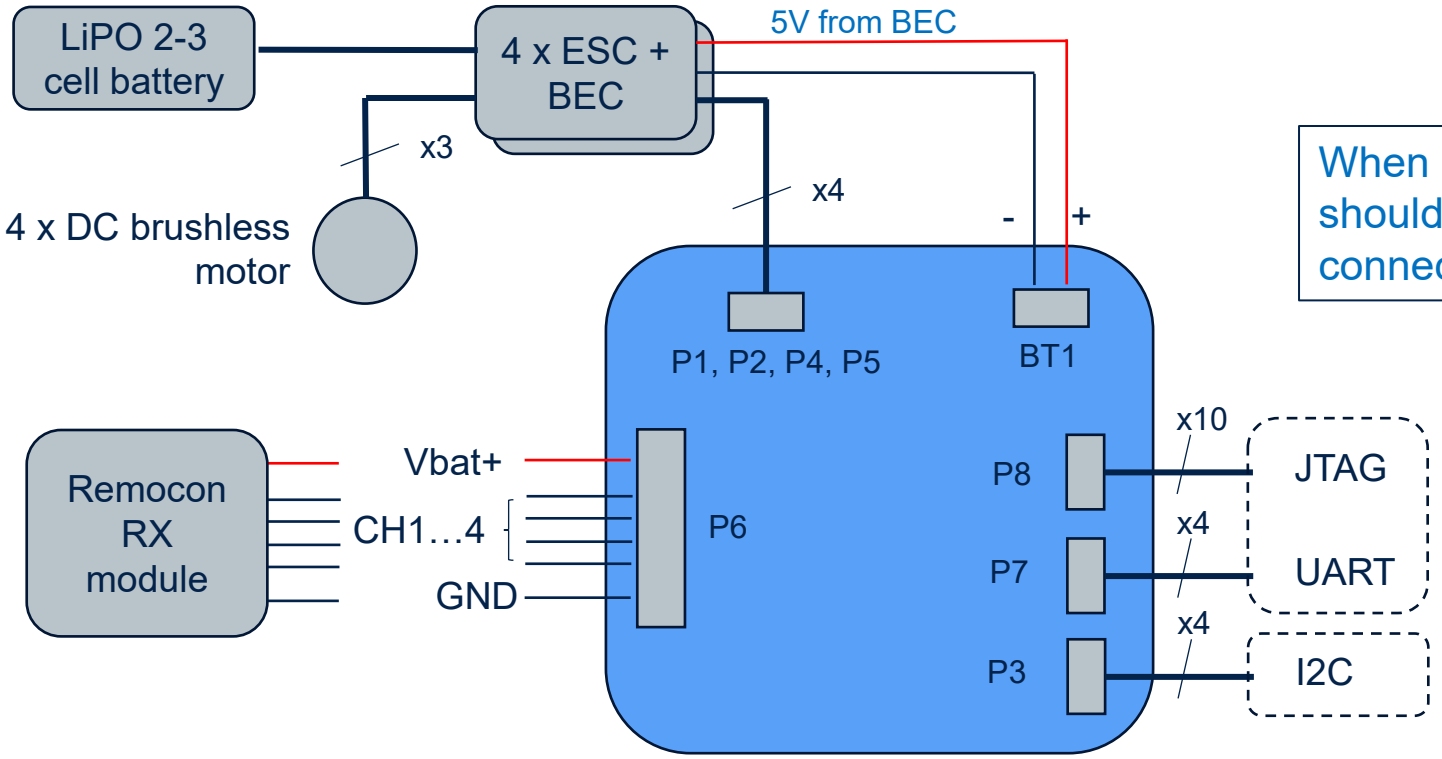
Available for future use



Resistor Configuration **R12, R13, R16, R17 mounted**
R14, R15, R18, R19 NOT mounted

life.augmented

STEVAL-FCU001V1 application connection (External ESC for DC brushless motors)

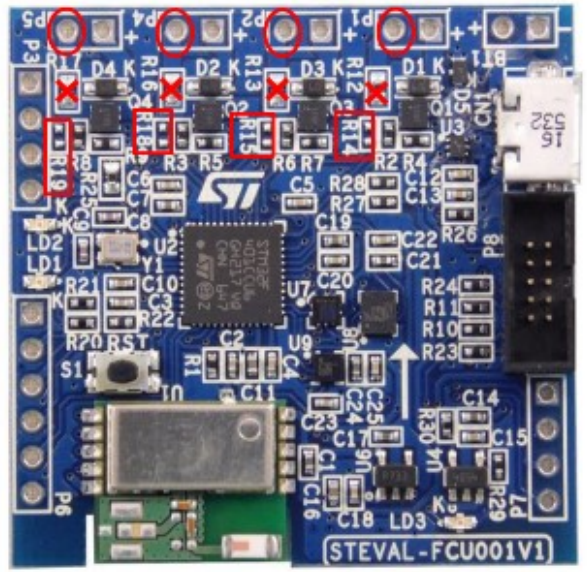


When connecting the ESC the PWM signal cable should be connected to pin2 of P1, P2, P4, P5 connectors (not the “+” pin).

Used only for debug purposes

Available for future use

STEVAL-FCU001V1
FCU eval board



Resistor Configuration

R12, R13, R16, R17 NOT mounted
R14, R15, R18, R19 mounted

life.augmented

STEVAL-FCU001V1 Software

STEVAL-FCU001V1 FW download

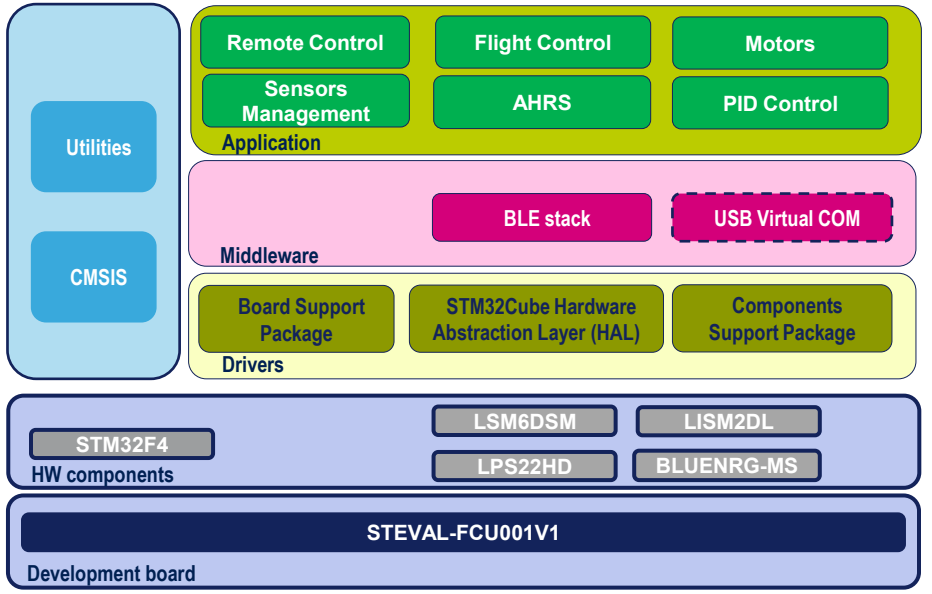
The source code of the STEVAL-FCU001V1 can be downloaded at the following link (or QR code):

https://github.com/STMicroelectronics-CentralLabs/ST_Drone_FCU_F401

Warning: The source FW code is intended to be used for evaluation of the board, and must be used or modified by experts. The FW is not designed and tested for commercial use and for the mass production of commercial products.

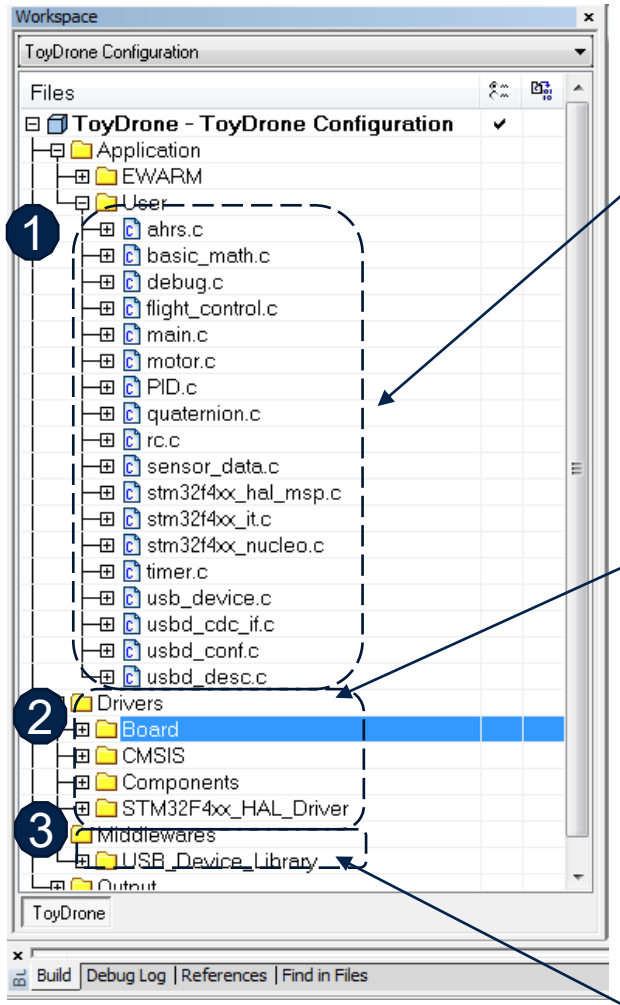
FW Architecture

- The firmware is structured in different levels:
  - ***Application**:* contains all the functions developed for the user to control the flight of the drone.
  - ***Middleware**:* libraries to manage connectivity (USB and BLE).
  - ***Drivers**:* the level nearest to the HW with all the functions managing the sensors and peripherals of the microcontroller.
- The *Utilities* and *CMSIS are functions directly related to the* STM32 architecture and corresponding Cortex-M4.



Module not yet implemented in current FW implementation.

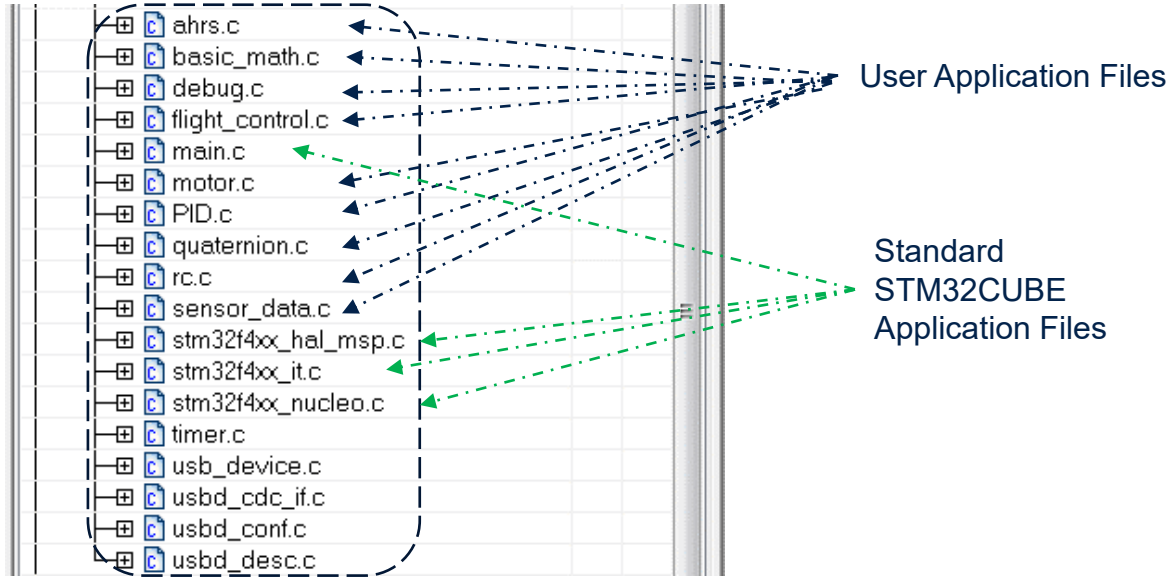
FW Project folder



Main user application. The main part of the FW project. The description of the FW implementation is focused on this section.

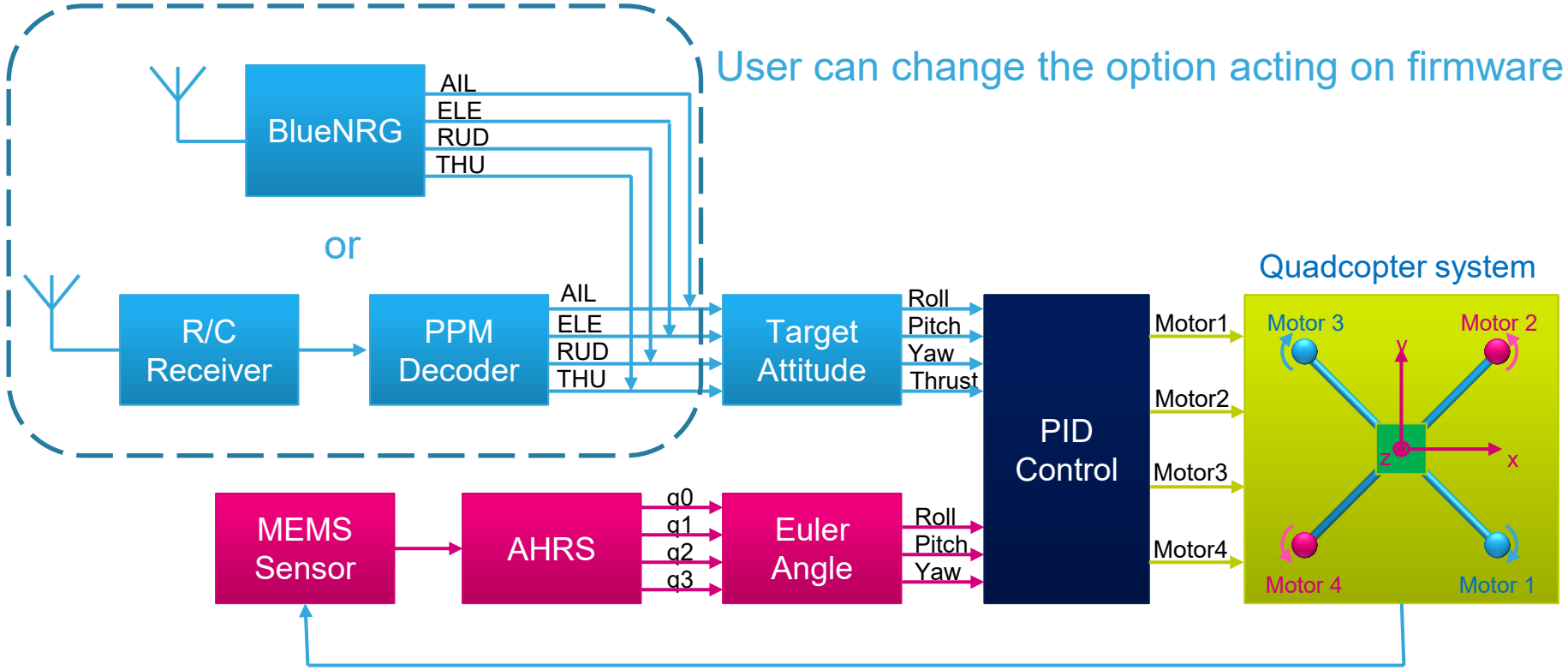
Device drivers and Board configuration (MEMs, BLE, etc.)

Middleware USB library. Not used at application level in this FW implementation.

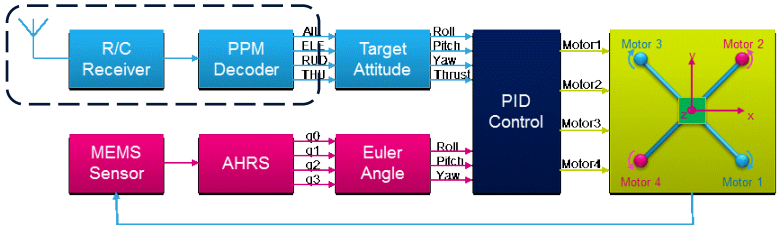


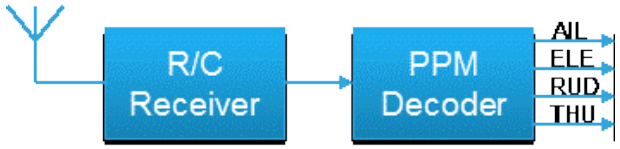
life.augmented

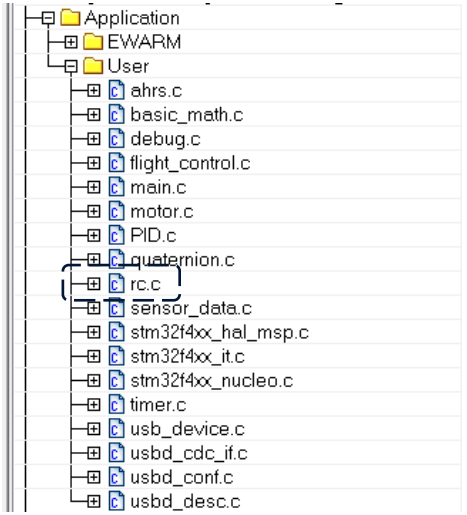
Overall FW block diagram



R/C receiver and PPM decoder







R/C Receiver and PPM decoder functions are implemented in *rc.c*. TIM2 (4 channels) input capture mode is used to calculate the pulse width of each PPM signal.
The pulse width is then decoded and the following R/C global variables are updated:

```
int16_t gAIL, gELE, gTHR, gRUD.
```

gAIL, gELE and gRUD are 0 centered (+/-0.5ms with 0.25us/LSB). gTHR is 0~1ms with 0.25us/LSB.

For details of TIM2 PPM decoding and data filtering, refer to the functions:

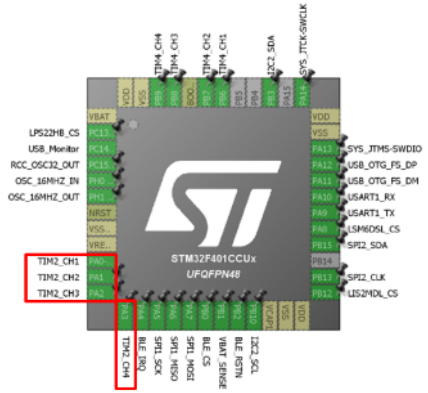
```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
void update_rc_data(int32_t idx)
```

Other variables that may be used to check remocon connection:

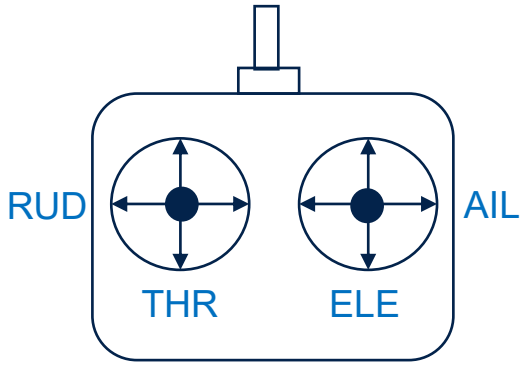
```
volatile int rc_timeout;        >> R/C timeout counter
char rc_connection_flag;        >> R/C connection status
```

life.augmented

R/C channels and R/C calibration



Channel	Function	Control	GPIO
CH1	Roll	AIL	PA0
CH2	Pitch	ELE	PA1
CH3	Thrust	THR	PA2
CH4	Yaw	RUD	PA3



The RC channels are mapped in the file *config_drone.h*, where it is also possible to define the timeout interval (if no signal from remocon motor OFF):

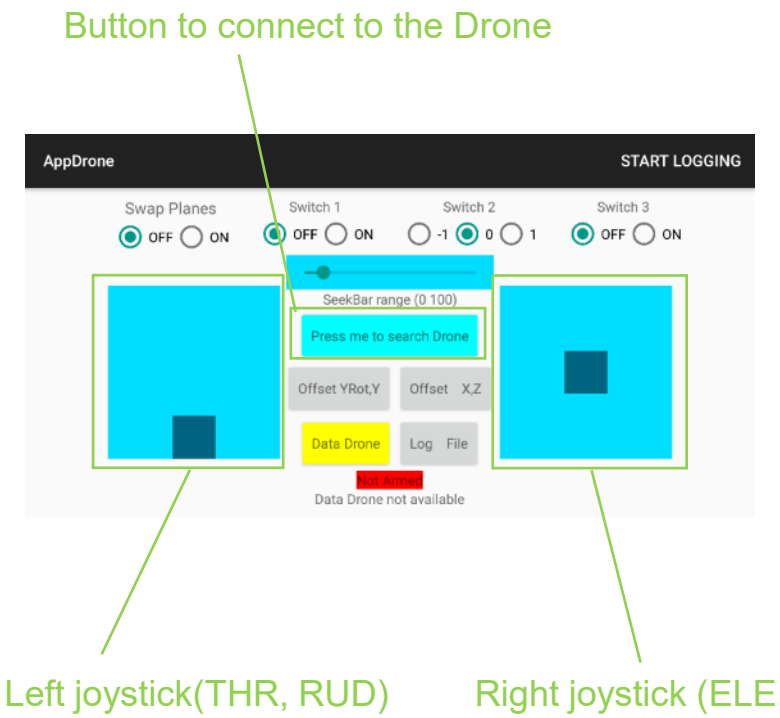
```
#define RC_TIMEOUT_VALUE    30
// Define the GPIO port for R/C input capture channels
#define RC_CHANNEL1_PORT    GPIOA
#define RC_CHANNEL1_PIN     GPIO_PIN_0
...
```

You can also calibrate the Remocon by fine tuning certain 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
```

life.augmented

Bluetooth ST AppDrone



- When using BLE to control the drone, the difference is in the presence of the BLE stack to manage the communication, and in the absence of the PPM Decoder, since the values are given directly by the BLE module with the variables:

```
int16_t gAIL, gELE, gTHR, gRUD.
```

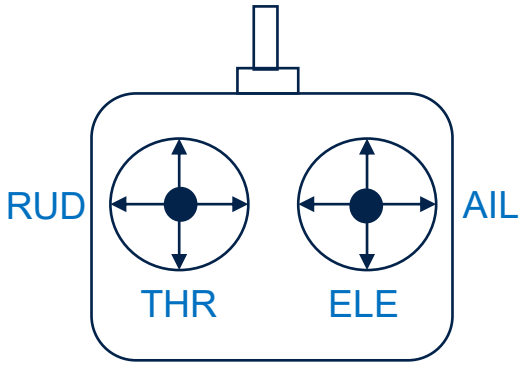
- To switch from the R/C to the BLE app configuration, simply change a `define` in the file «`rc.h`»:

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

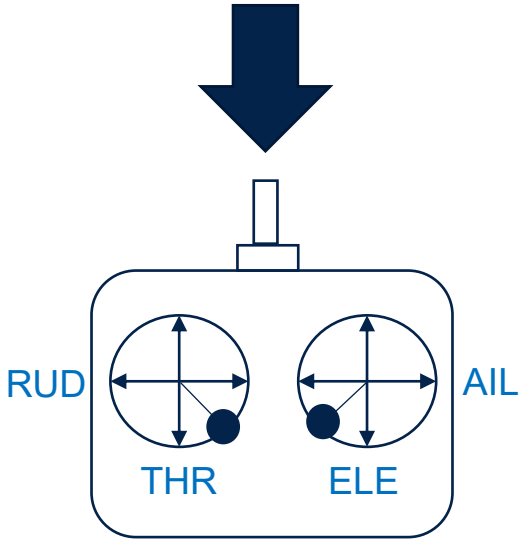
Warning that currently the Android App is still in Beta version and can vary significantly over coming releases.

life.augmented

Sensor calibration procedure



"Standby"



"Calibration"

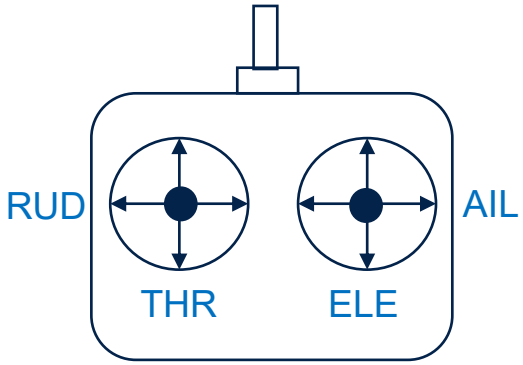
If the battery is applied to the FCU (or reset button pushed) when the quadcopter is not on a flat surface (or the FCU not mounted flat on the frame), the AHRS Euler angle will have an offset. To eliminate it, run a simple calibration procedure through the remocon after startup.

Calibration procedure may be customized in the function *update_rc_data*:

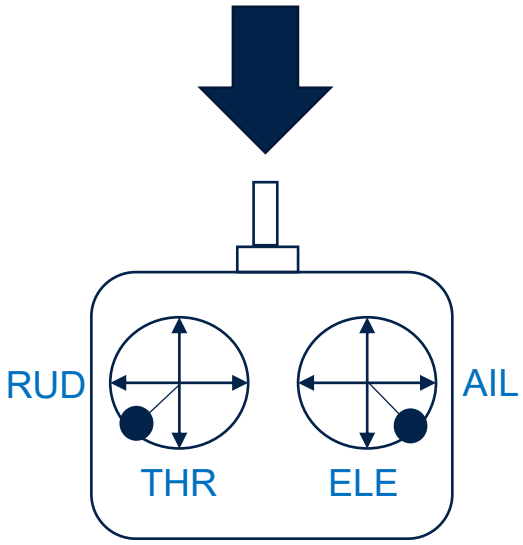
```
if ( (gTHR == 0) && (gELE < -RC_CAL_THRESHOLD) && (gAIL >
RC_CAL_THRESHOLD) && (gRUD < -RC_CAL_THRESHOLD))
  {
    rc_cal_flag = 1;
  }
```

life.augmented

Motor arming procedure



"Standby"



"Armed"

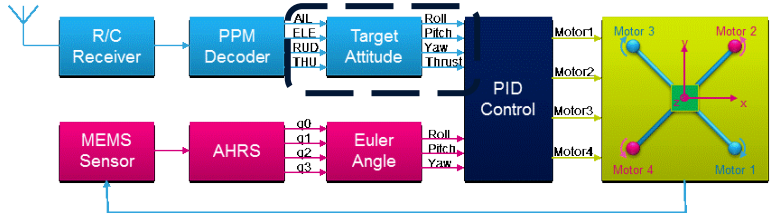
For safety reasons (in order to avoid any damage or injury when the quadcopter battery is inserted and remocon is switched ON), an *Arming* procedure is always implemented: motors are switched off until a certain sequence of commands on remocon is applied.

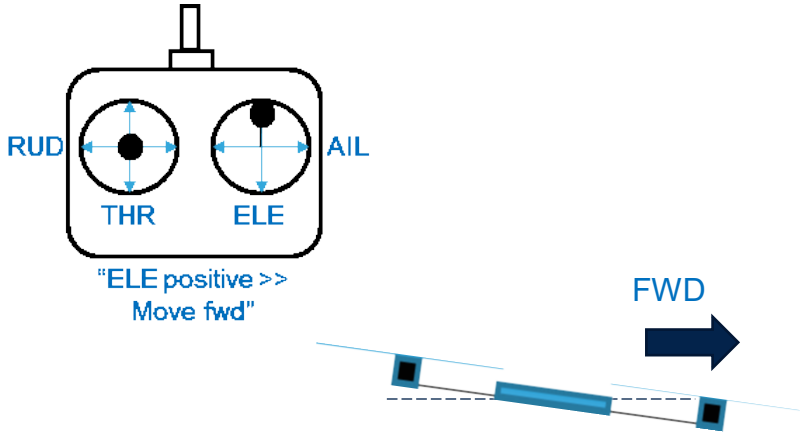
Arming procedure may be customized in the function *update_rc_data*:

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

life.augmented

Target attitude





Conversion from Remocon Euler angle to AHRS Euler angle (needed for setting the target of the PID control loop) is implemented in the *rc.c* file in the function:

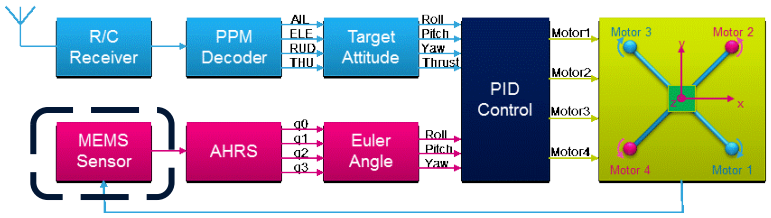
void GetTargetEulerAngle(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs)

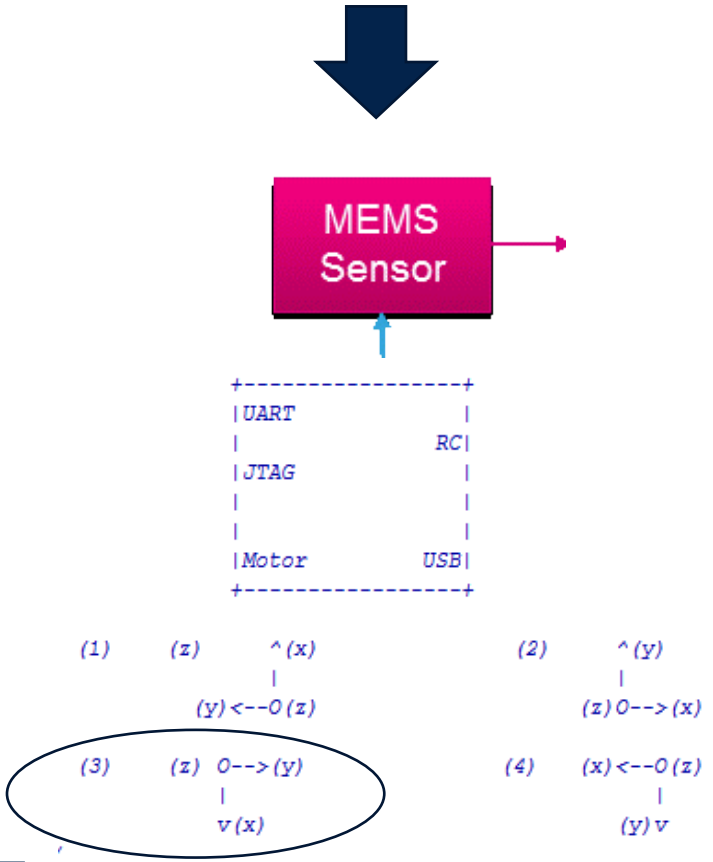
The data from the Remocon (i.e. `gELE`) is normalized to the full scale value and converted in angle:

```
 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 change of sign (`-t1`) is related to the fact than increasing ELE (plus compared to Mid level) to move forward, corresponds to a negative target angle for pitch.

MEMS sensor





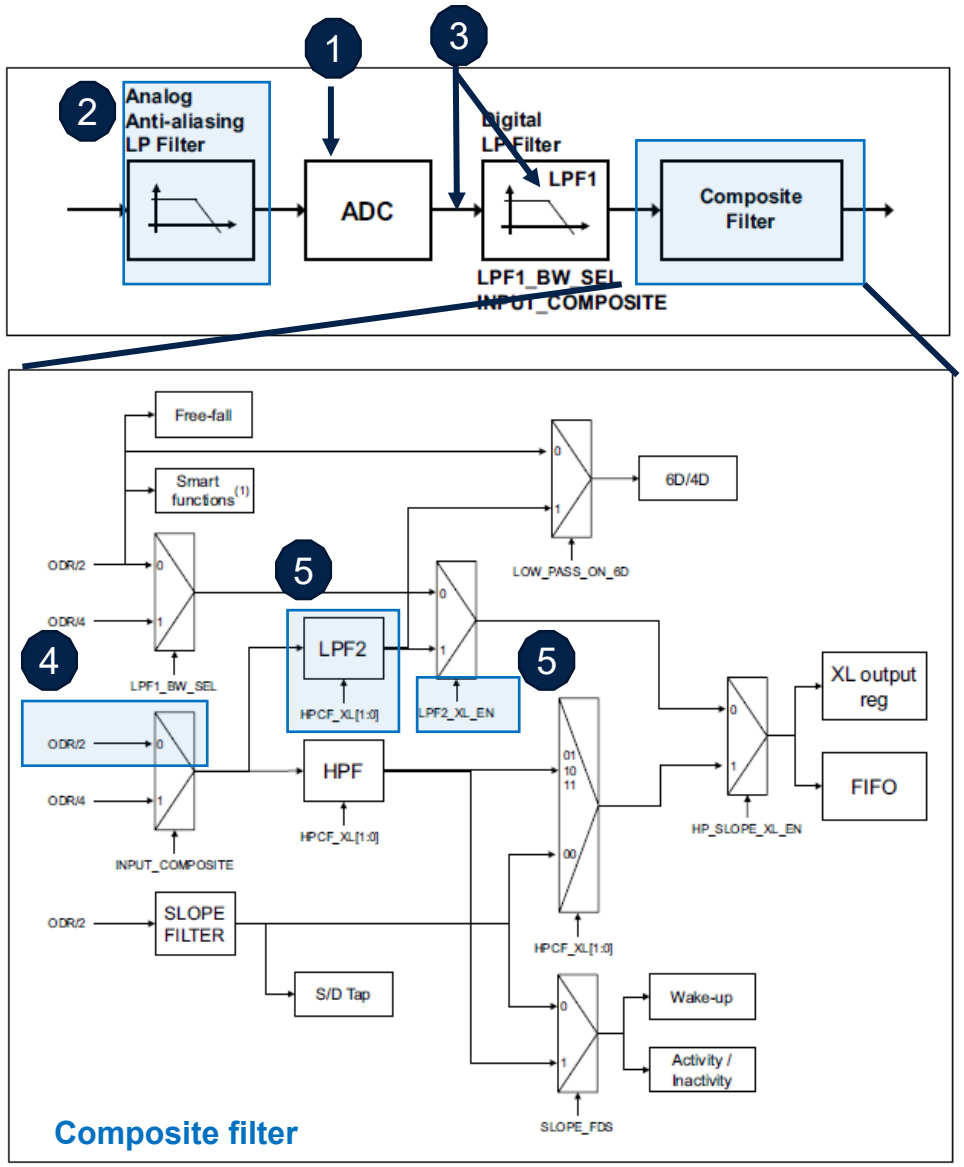
The main sensors used for flight stabilization are the accelerometer and gyroscope (LSM6DSL device). Sensor drivers are implemented in the function *lsm6dsl.c*.
In the FW project there are also drivers for magnetometer and pressure sensors (*lis2mdl.c* and *lps22hb.c*), but they are not used in the current FW implementation.
Initialization of the sensors is performed in the following function in the *board.c* file:

```
IMU_6AXES_StatusTypeDef BSP_IMU_6AXES_Init(void)
```

RAW data from 6-axis acc+gyro sensors are then converted into [*mg]* and [*mdps]* units and translated to the coordinate system used (orientation of FCU board vs motor configuration) in *sensor_data.c*. Default Coordinate system that corresponds to configuration described in previous assembling description is #3.

life.augmented

Accelerometer and gyroscope sensor setup



The Flight control algorithm is based on the data generated by accelerometer and gyroscope sensors. It is important to ensure that sensors are well tuned. It is especially important that RAW data coming from sensors are not impacted by mechanical noise generated by the vibration of the motors and propagated through the frame.

In the figure block diagram of the LSM6DSL accelerometer chain.

Accelerometer sensor setup

- Accelerometer key parameter settings:
  - Accelerometer full-scale selection FS ***default ±2g >> set to ±4g***
  - Analog Filter Bandwith ***default 1.5KHz >> keep default***
  - Output data rate and power mode selection ODR ***default Power down mode >> set to 1.6kHz***
  - Composite filter input selection ***default ODR/2 >> keep default***
  - Accelerometer low-pass filter LPF2 selection ***default >> set to Low pass filter enabled @ ODR/50 or ODR/100 depending on mechanical vibration noise***

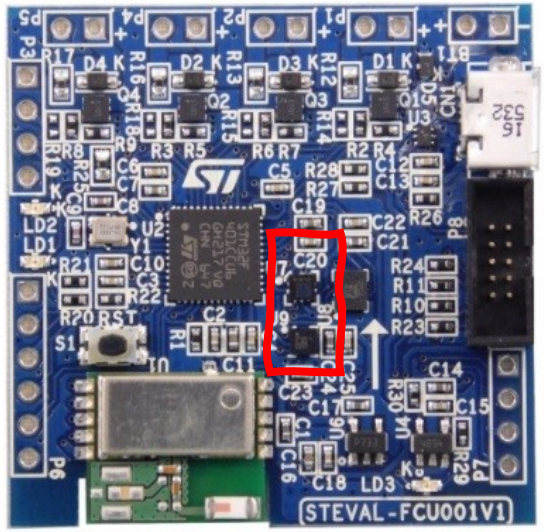
```
BSP_ACCELERO_Set_ODR_Value(LSM6DSL_X_0_handle, 1660.0);       /* ODR 1.6kHz */
BSP_ACCELERO_Set_FS(LSM6DSL_X_0_handle, FS_MID);                    /* FS 4g */
LSM6DSL_ACC_GYRO_W_InComposit(LSM6DSL_X_0_handle, LSM6DSL_ACC_GYRO_IN_ODR_DIV_2);   /* ODR/2 low pass filtered sent to composite filter */
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);
```

Gyroscope sensor setup

- Gyroscope key parameter settings:
  - Gyroscope Full Scale selection ***default 245 dps >> set to 2000dps***
  - Gyroscope output data rate selection ODR ***default power down >> set to 104Hz***
  - Gyroscope's low-pass filter bandwidth selection LPF1 FTYPE ***>> set to Narrow (10b)***

```
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 */
```

Use of Pressure sensor and Magnetometer



To use the Pressure sensor (for altitude estimation) and Magnetometer (for Compass), it is necessary to enable the sensors in the *config_drone.h*.

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

The data from sensors are updated at the rate of 800Hz (using Timer9 interrupt):

```
ReadSensorRawData(&acc, &gyro, &mag, &pre);
```

So if it's needed for example to display the data on UART with a printf for debugging purpose:

```
PRINTF("Pressure [atm] = %f\n\n",pre);
PRINTF("Magnetometer X = %d\tY = %d\tZ = %d\n\n", mag.AXIS_X, mag.AXIS_Y, mag.AXIS_Z);
```

Note Pressure data are expressed in *atmospheres*, while magnetometer data in *degrees* (from north magnetic pole).

life.augmented

Sensor calibration after startup (1/2)

Expression	Value	Location
pre	1.03042163E+3	0x20000B38
p_id	Error (col 1): Un...	
⊞ mag	<struct>	0x200009F8
VBAT_Sen...	Error (col 1): Un...	
VBAT	Error (col 1): Un...	
gELE	−1	0x20000B54
gAIL	−422	0x20000B52
gRUD	199	0x20000B58
gTHR	0	0x20000B56
⊟ motor_pwm	<struct>	0x2000097C
motor1_p...	0.0	0x2000097C
motor2_p...	0.0	0x20000980
motor3_p...	0.0	0x20000984
motor4_p...	0.0	0x20000988
⊟ euler_ahrs	<struct>	0x200009C8
thx	4.57809686E−1	0x200009C8
thy	−1.09683387E−2	0x200009CC
thz	0.0	0x200009D0
⊞ ahrs	<struct>	0x2000094C
⊞ pid	<struct>	0x20000498

Expression	Value	Location
pre	1.03048217E+3	0x20000B38
p_id	Error (col 1): Un...	
⊞ mag	<struct>	0x200009F8
VBAT_Sen...	Error (col 1): Un...	
VBAT	Error (col 1): Un...	
gELE	0	0x20000B54
gAIL	−422	0x20000B52
gRUD	199	0x20000B58
gTHR	0	0x20000B56
⊟ motor_pwm	<struct>	0x2000097C
motor1_p...	0.0	0x2000097C
motor2_p...	0.0	0x20000980
motor3_p...	0.0	0x20000984
motor4_p...	0.0	0x20000988
⊟ euler_ahrs	<struct>	0x200009C8
thx	−7.1783561722E−4	0x200009C8
thy	−6.69909815E−4	0x200009CC
thz	0.0	0x200009D0
⊞ ahrs	<struct>	0x2000094C
⊞ pid	<struct>	0x20000498

After inserting the battery, it is important to put the quadcopter on a flat surface and press the reset button. In fact, just after the reset, if the quadcopter has a certain inclination, there will be an offset in the *euler_ahrs* coordinates.
This happens because an automatic sensor calibration is performed immediately following startup.

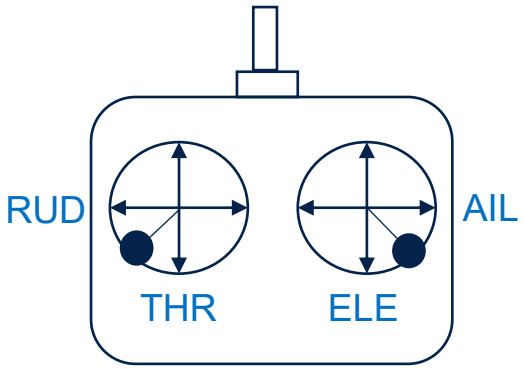
WRONG
Inclined position at startup >> euler_ahrs.thx = 0.457 * 53 = 24.2 deg

CORRECT
Flat position at startup >> euler_ahrs.thx = -0.0371 * 53 = -1.96 deg

life.augmented

Sensor calibration after startup (2/2)

```
if(sensor_init_cali == 0)
  {
    sensor_init_cali_count++;
    if(sensor_init_cali_count > 800)
    {
      // Read sensor data and prepare for specific coodinate 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;
      }
    }
```



"Sensor calibration"

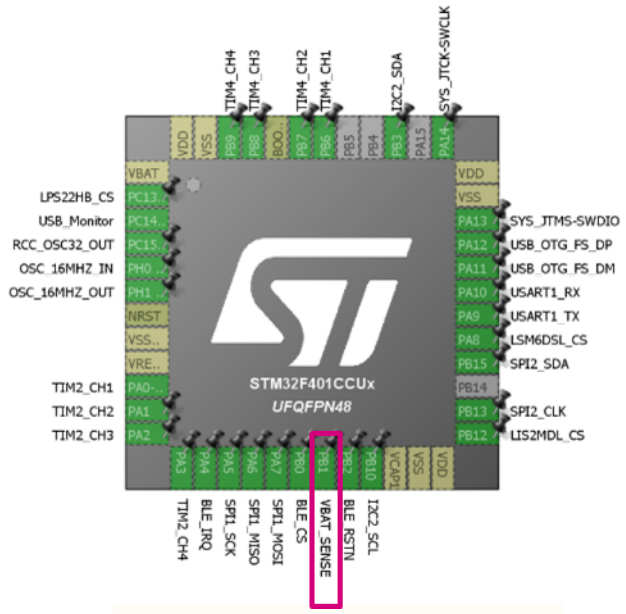
Looking into detail, in the *HAL_TIM_PeriodElapsedCallback* function, both the accelerometer and gyroscope are calibrated:

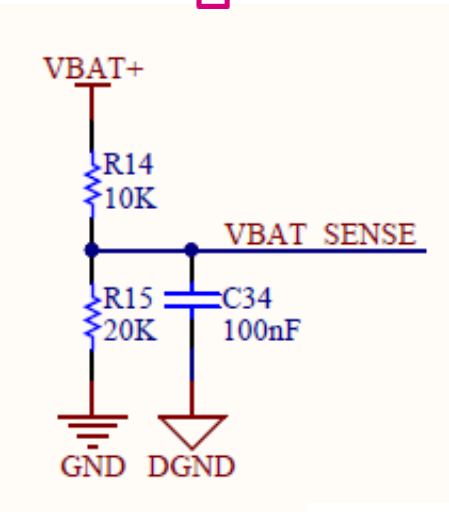
1. When the *sensor_init_cali* variable is 0, just after the system startup (battery inserted or reset button pushed).
2. When the *rc_cal_flag* variable is 1, manual calibration launched before the motors are armed if the sticks are in a certain position (check *update_rc_data* function in *rc.c* file):

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

life.augmented

Battery voltage monitoring





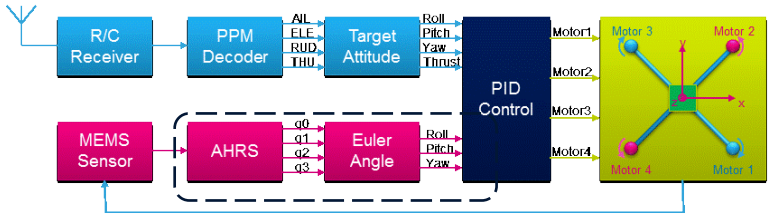
It is sometimes important to monitor the level of the battery for telemetry purposes, or to trigger certain actions (stop the motor, sound alarm, …).

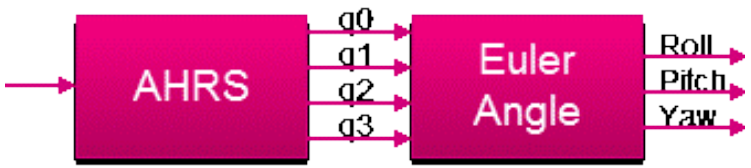
For this reason, the ADC1 channel 9 (PB1) is connected to Vbat through a resistor partition.
For debug purposes, a simple code fragment is inserted in the *main.c* to perform a single ADC reading(*) and send the data converted to UART:

```
HAL_ADC_Start(&hadc1);
if (HAL_ADC_PollForConversion(&hadc1, 1000000) == HAL_OK)
   {
        VBAT_Sense = HAL_ADC_GetValue(&hadc1);
        VBAT = (((VBAT_Sense*3.3)/4095)*(BAT_RUP+BAT_RDW))/BAT_RDW;
        PRINTF("Battery voltage = %fV\r", VBAT);
   }
   HAL_ADC_Stop(&hadc1);
```

(*) It may eventually be possible to use the ADC with direct conversion to DMA, without launching the ADC conversion each time.

AHRS





$$\begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix} = \begin{bmatrix} atan2\left(2q_2q_3 + 2q_0q_1, q_3^2 - q_2^2 - q_1^2 + q_0^2\right) \\ -asin\left(2q_1q_3 - 2q_0q_2\right) \\ atan2\left(2q_1q_2 + 2q_0q_3, q_1^2 + q_0^2 - q_3^2 - q_2^2\right) \end{bmatrix}$$

The RAW data from the Accelerometer and Gyroscope are passed to the AHRS algorithm :

```
ahrs_fusion_ag(&acc_ahrs, &gyro_ahrs, &ahrs);
```

function present in the *ahrs.c.*

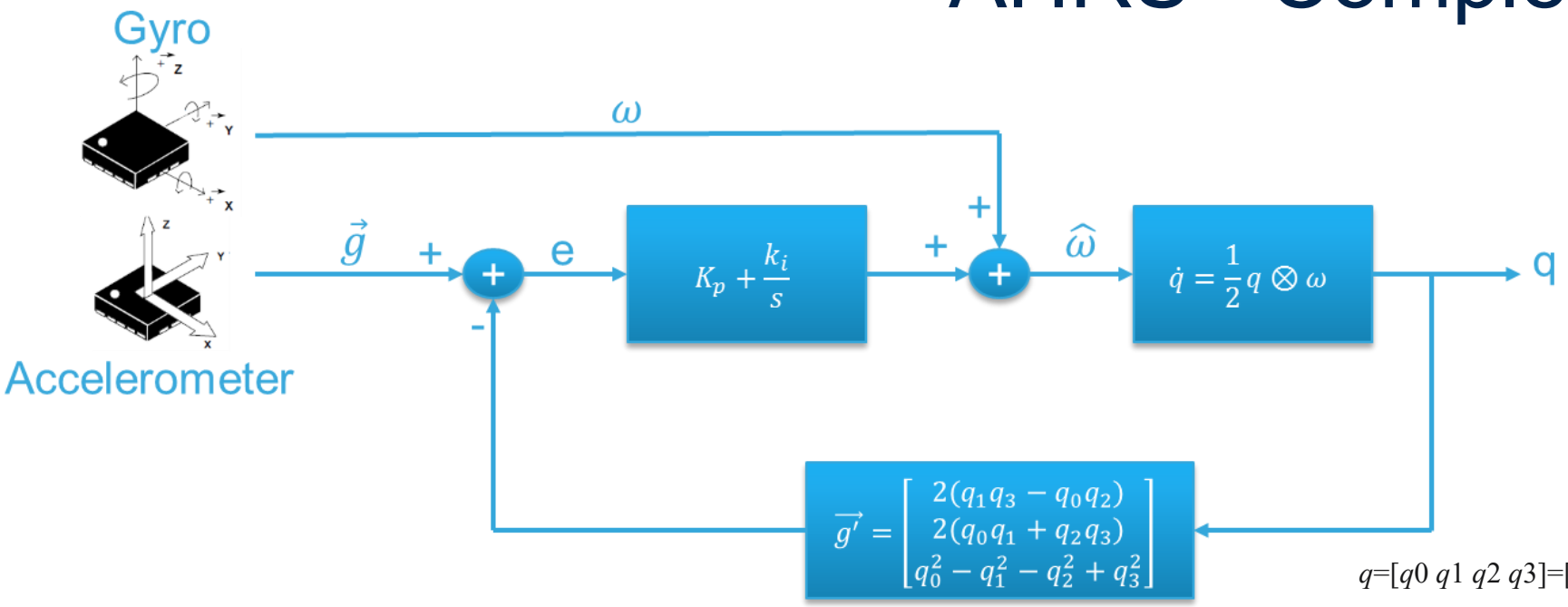
In order to perform the stabilization algorithm for the drone, data from AHRS quaternion must be translated to Euler angle:

```
QuaternionToEuler(&ahrs.q, &euler_ahrs);
```

function present in *quaternion.c*

Attitude & Heading Reference System (AHRS) is the key algorithm for a UAV system. The algorithm chosen for the current FW of STEVAL_DRN01 board is the “Complementary filter”.

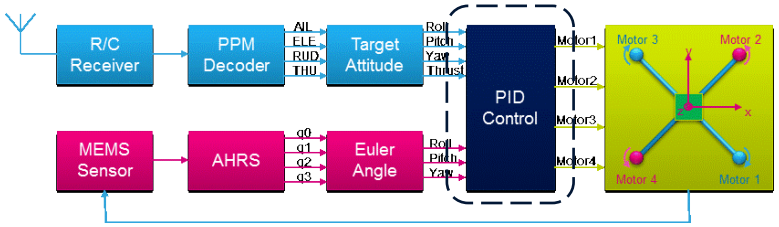
AHRS - Complementary Filter

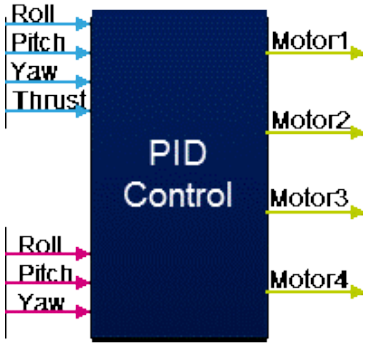


$q=[q0\ q1\ q2\ q3]=[cos\alpha/2 \quad rx*sin\alpha/2 \quad ry*sin\alpha/2 \quad rz*sin\alpha/2]$

- Accelerometer and gyroscope can calculate the attitude independently.
- Accelerometer is always influenced by high-frequency noise, and gyroscope has a low-frequency offset.
- Gyroscope results are accurate in short-term, but will generate an offset over time. Acc is not accurate in short-term because of noise, but is reliable over time.
- Based on Mahoney. Use the gyroscope to get the attitude, use the acc to calibrate the attitude (the accel. Measurement of gravity is used to estimate the gyro drift on pitch and roll).

Flight PID Control





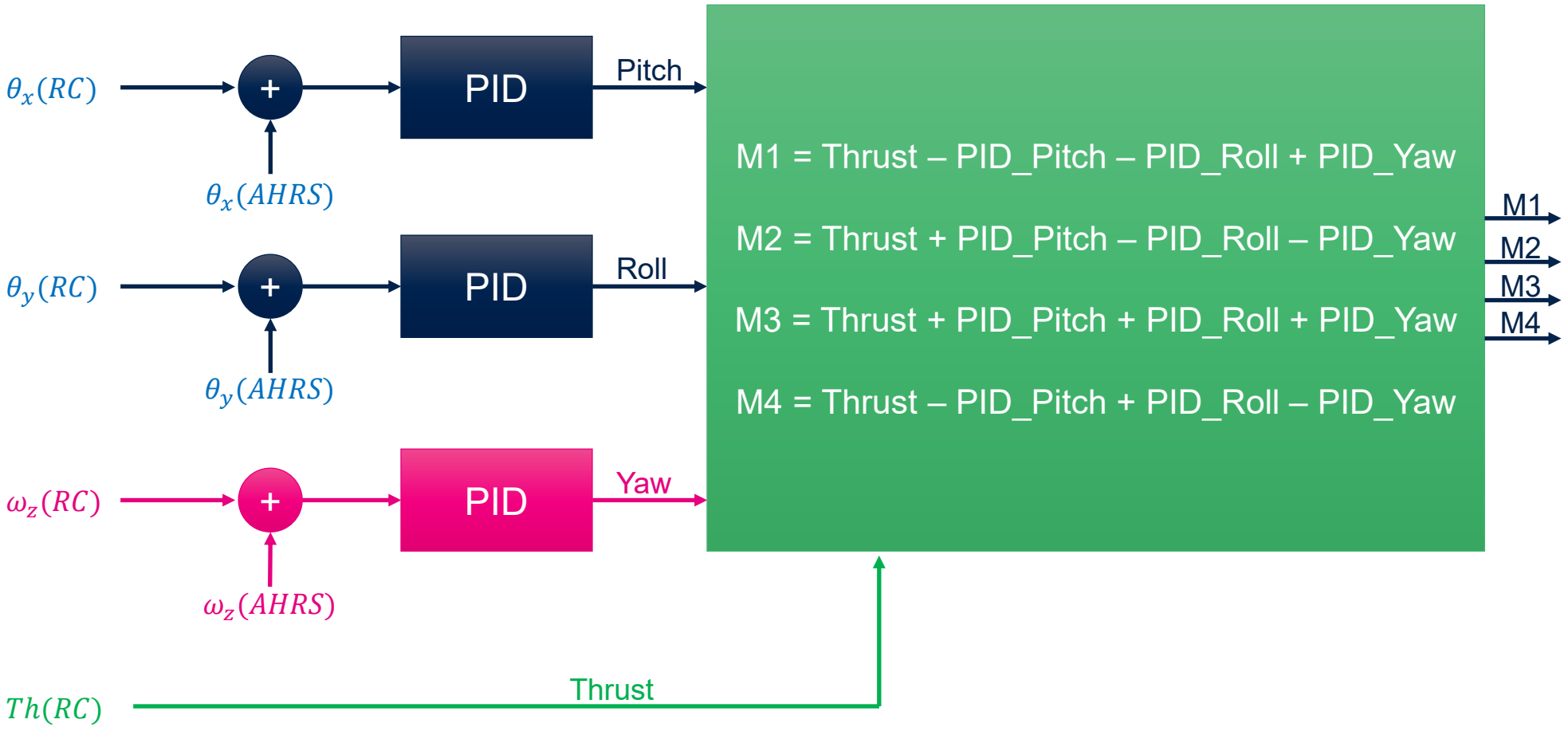
The stabilization algorithm is performed with PID control. Target for the PID is given by the Euler angle imposed by the Remocon, which is compared to the Euler angle of the actual drone position.

```
FlightControlPID_OuterLoop(&euler_rc_fil, &euler_ahrs, &ahrs, &pid);
```

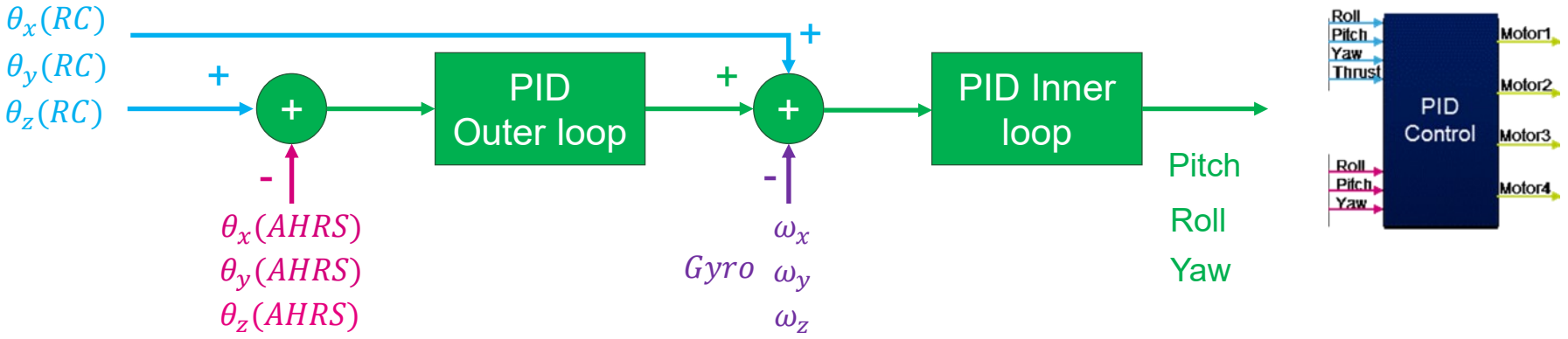
function present in *flight_control.c*.
The output of the PID is the value of the 4 motors's speed (PWM HW signal for ESC or Mosfet driving) for the quadcopter:

```
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;
```

Flight PID Control



Flight PID Control



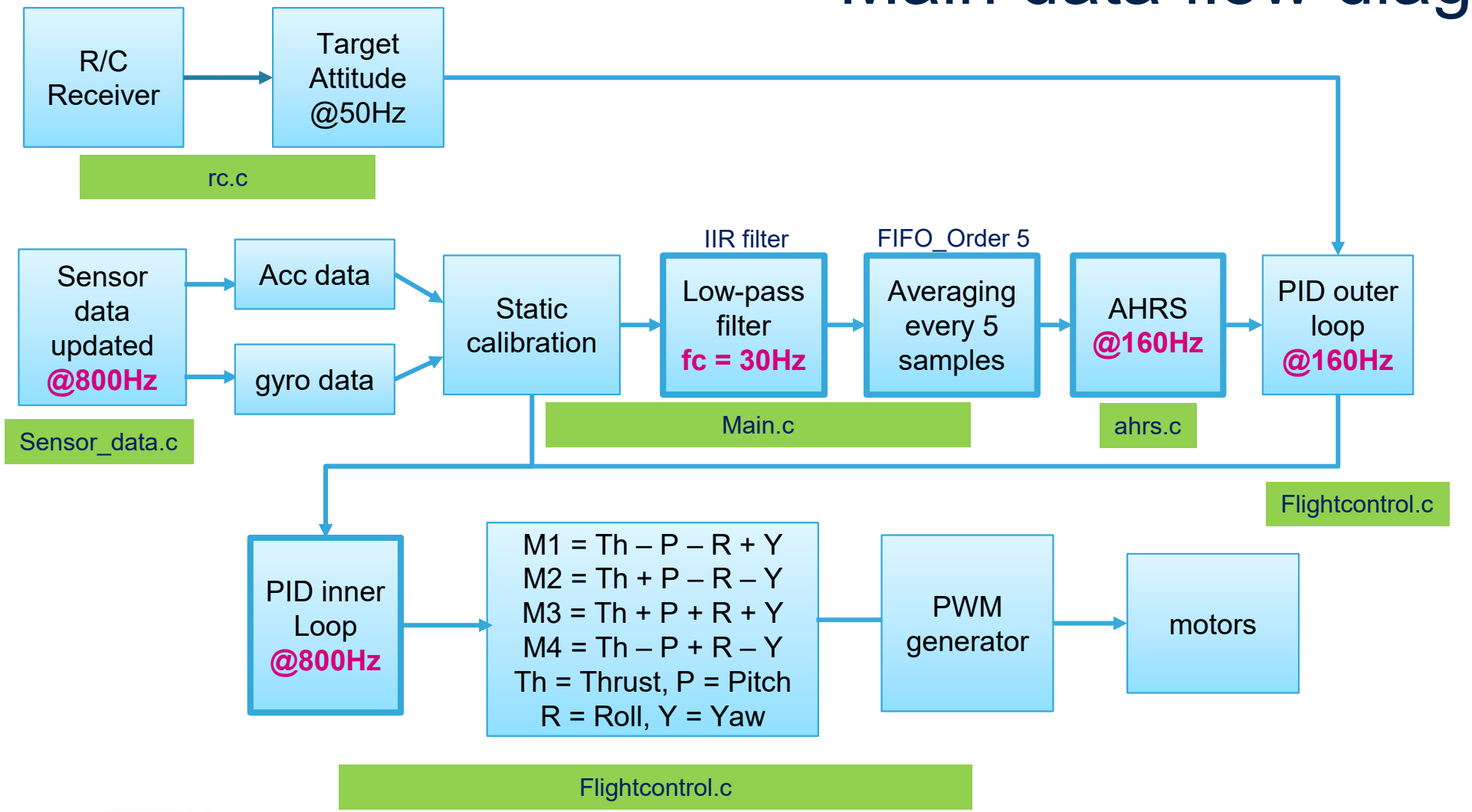
PID control is performed in two separate stages:

- *PID Outer loop*: comparing the target Euler angles given by Remocon and the Euler angles of the AHRS system, it controls the incline angle
- *PID Inner loop*: used to track the angular rate.

Using a mathematical model of the drone system, it is almost impossible to make the system stable in a single loop. The inner loop has to be added to the outer loop in order to stabilize the system. The relative functions are present in *flight_control.c*:

```
void FlightControlPID_OuterLoop(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs, AHRS_State_TypeDef *ahrs,
P_PI_PIDControlTypeDef *pid);
void FlightControlPID_innerLoop(EulerAngleTypeDef *euler_rc, Gyro_Rad *gyro_rad, AHRS_State_TypeDef *ahrs,
P_PI_PIDControlTypeDef *pid, MotorControlTypeDef *motor_pwm);
```

STEVAL-FCU001V1 FW
Main data flow diagram



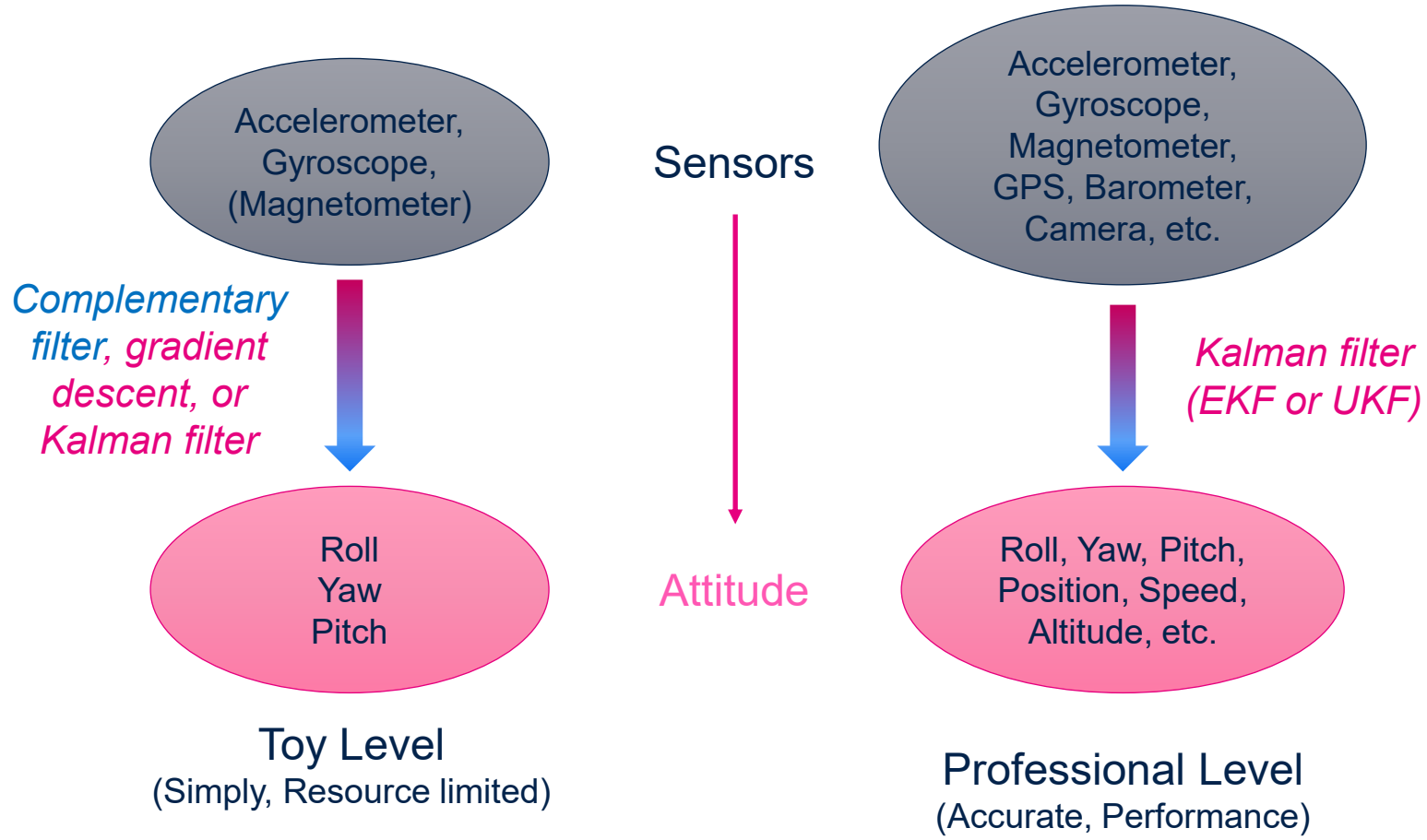
Blocks in **bolds** are the Key ones for stabilization and anti-drifting.

Thank you

life.augmented

Appendix A
AHRS EKF Formula details

AHRS Algorithm



AHRS - EKF (1)

- 13 States, 3 Measurements
- Gyro data is G input, Accelerometer data A is the measurement

State: $x = \begin{bmatrix} Q \\ G_{0ff} \end{bmatrix}$, where

$Q = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ is the quaternion;

$G_{off} = [g_{x\_off} \quad g_{y\_off} \quad g_{z\_off}]^T$ is the gyro bias error

Measurement: $z = [A]$, where $A = [a_x \quad a_y \quad a_z]^T$ is the gravity measured by accelerometer

State Equation

$$\hat{x}_{k+1} = \begin{bmatrix} \hat{Q}_{k+1} \\ \hat{G}_{off\ k+1} \end{bmatrix} = \begin{bmatrix} Q_k + \frac{1}{2} Q_k \otimes (G_k - G_{b\ k}) \cdot T \\ G_{off\ k} \end{bmatrix}$$

Measurement Equation

$$\hat{z}_{k+1} = \hat{Q}^*_{K+1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \hat{Q}_{K+1}$$

State Update

$$x_{k+1} = \hat{x}_{k+1} + K_{k+1}(z_{k+1} - \hat{z}_{k+1})$$

AHRS - EKF (2)

- 13 States, 3 Measurements
- Gyro data is G input, Quaternion from accelerometer & magnetometer data $Z_k = Q_M$ is the measurement

State: $x = \begin{bmatrix} Q \\ G_{off} \\ M_e \\ M_{off} \end{bmatrix}$, where $Q = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ is the quaternion;

$G_{off} = [g_{x\_off} \quad g_{y\_off} \quad g_{z\_off}]^T$ is the gyro bias error; $M_e = [m_x \quad m_y \quad m_z]^T$ is true earth magnetic field measured in body frame; $M_{off} = [m_{x_{off}} \quad m_{y_{off}} \quad m_{z_{off}}]^T$ is the hard iron offset

Measurement: $z = [q]$, where $A = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ is the quaternion estimated by gravity & magnetic field

State Equation $\hat{x}_{k+1} = \begin{bmatrix} \hat{Q}_{k+1} \\ \hat{G}_{b\,k+1} \\ M_{e\,k+1} \\ M_{off\,k+1} \end{bmatrix} = \begin{bmatrix} Q_k + \frac{1}{2} Q_k \otimes (G_k - G_{b\,k}) \cdot T \\ G_{b\,k} \\ M_{e\,k} - T\Omega_k M_{e\,k} \\ M_{off\,k} \end{bmatrix}$

Measurement Equation $\hat{z}_{k+1} = \hat{Q}_{k+1}$

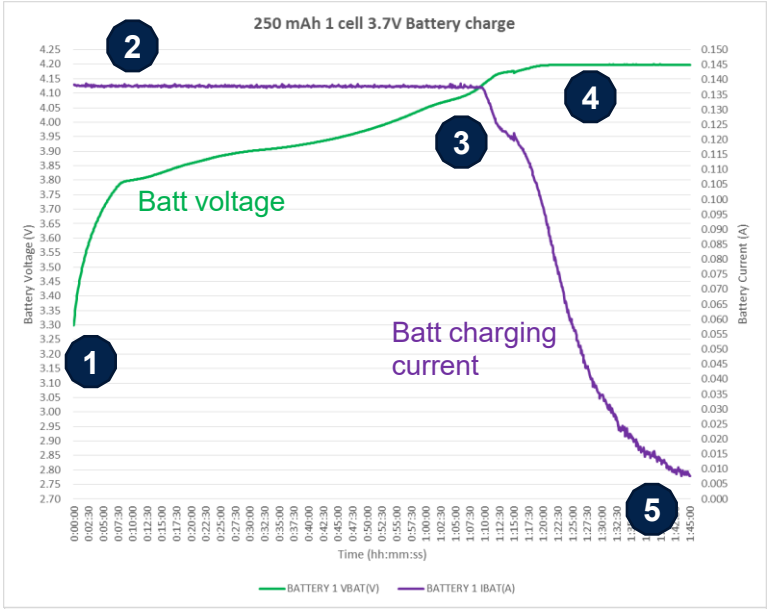
State update $x_{k+1} = \hat{x}_{k+1} + K_{k+1}(z_{k+1} - \hat{z}_{k+1})$

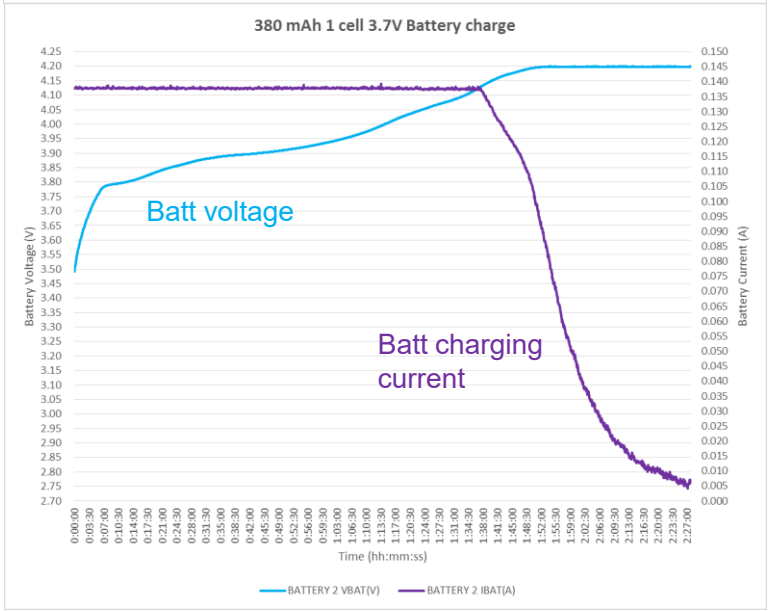
life.augmented

Appendix B
On board Battery charger for LiPO

life.augmented

Charge of a LiPO battery





The diagram shows a typical charging curve for a LiPo battery. It consists of the following steps:

1. Battery discharged at cutoff voltage (2.7V).
2. Charge current set to ~135mA (*).
3. When Battery voltage reaches ~4.1V the charging switches from constant current mode to constant voltage mode. Battery voltage increases slowly till 4.2V and Charging current decrease.
4. 4.2V voltage at maximum charge
5. Charge current reached termination (1/10 of charging current) value, it just keeps battery charge.

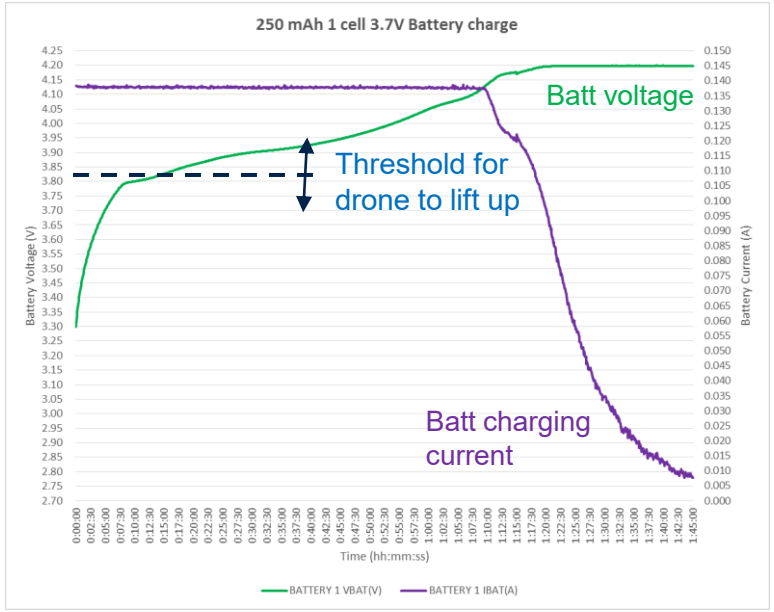
The two diagrams are relative to 2 different batteries:

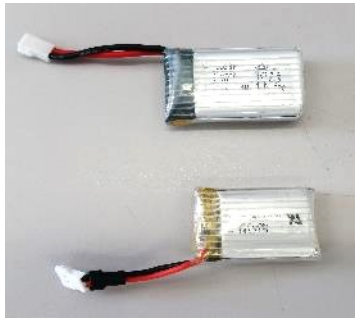
- 250mAh battery >> ~1h 25m from full discharge to full charge
- 380mAh battery >> ~2h from full discharge to full charge

(*) Charge current can be changed by the resistor connected to PROG pin of STLC4054.

life.augmented

Use of LiPO battery in drone application





380mAh 1 cell Batt >> 11g

250mAh 1 cell Batt >> 8g

Note the following tips when using a LiPO Battery in the application:

1. Depending on several factors (drone total weight, motors and propeller used, …), the drone may not be able to lift (or just hovering at few cm from ground) at a voltage greater than full discharge value (i.e. 3.5V or 3.6V). It will be necessary to charge the battery from this voltage level, it will shorten battery charging time.

2. Choosing a larger battery will increase flight time, but attention must be paid to the trade off with the total weight.

3. After flight, battery should be immediately disconnected from FCU board. If it remains connected for a long time and battery voltage drops below 2.7V, there is the risk of permanently damaging the battery to the point that it can no longer be charged.

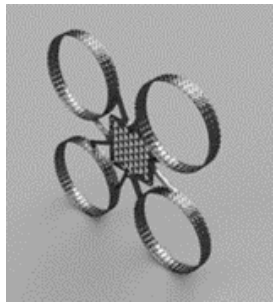
4. It is recommended that charge current should not exceed ~1/3 of maximum capacity (i.e. 380mAh >> charging current 130mA).

life.augmented

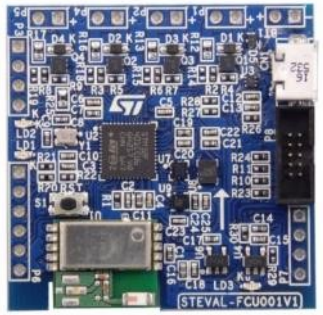
Appendix C
Assemble the Drone

Assemble the items: mechanical structure

The items below represent possible assembly items for a small drone and have already been tested with the current FW version, demonstrating stable flight performance.



3D mechanical frame



STEVAL-FCU001V1 board

1-cell battery 3.7V 600mAh 30C

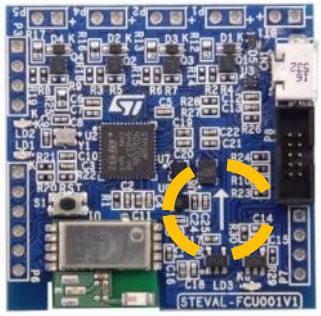
2 x CW, 2 x CCW coreless DC motor 8.5x20mm

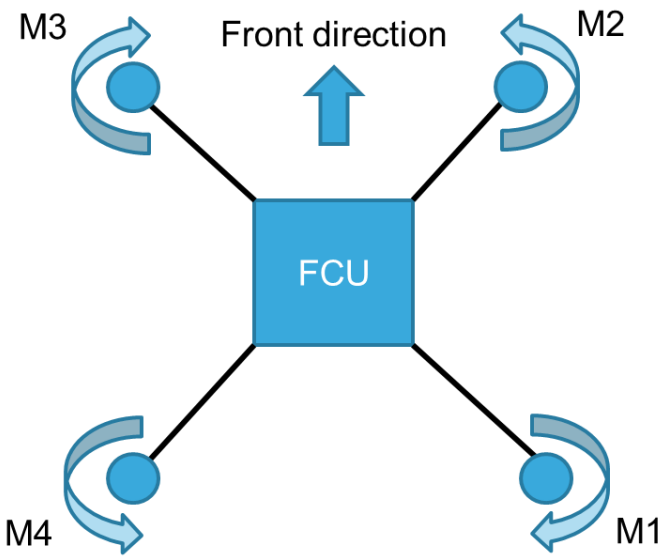
65mm propeller

Remote Controller or Smartphone App

Warning: You may choose different items but need to ensure any necessary FW adjustments

Step 1: mounting STEVAL-FCU001V1 board on frame



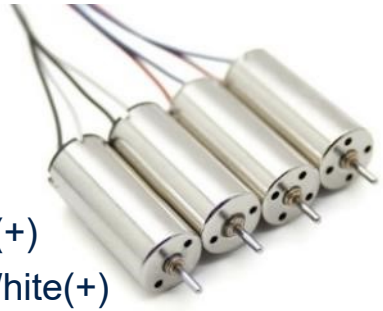


- The STEVAL-FCU001V1 must be mounted in the center of the drone, equidistant from the four motors.
- Front direction is indicated on the STEVAL-FCU001V1 by a **small arrow**.
- The board must be mounted as flat as possible and well anchored to the frame.
- You can use an adhesive sponge between the frame and the FCU board to minimize mechanical vibration from motors.
- Current FW supports the common “x” configuration shown here. To use a “+” configuration, some modifications to the FW are needed (*).

(*) Please refer to the FW explanation session for more details.

Step 2: DC motor connection

DC motors should be connected to P5, P4, P2 and P1 following the sequence in the figure.



CW: Blue(-)/Red(+)
CCW: Black(-)/White(+)

Warning: Check the documentation of Motor Manufacturer for wire color assignment

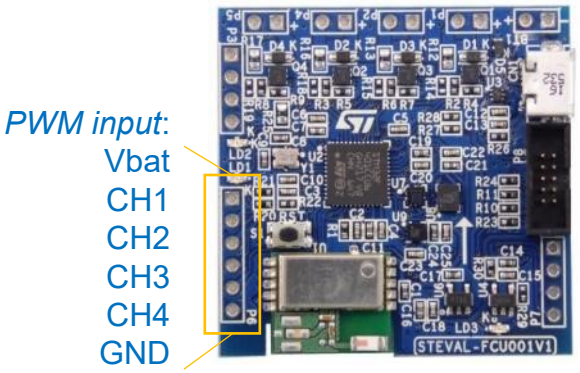
Warning: A CW DC motor can rotate in CCW direction if “+” and “-” are reversed, but the brushes are designed for CW rotation and prolonged use in the opposite direction may reduce its lifetime

Note: This configuration is valid also for 3-ph DC brushless motors connected through external ESC

Note: This configuration respects the directions and Motor placement indicated on page before

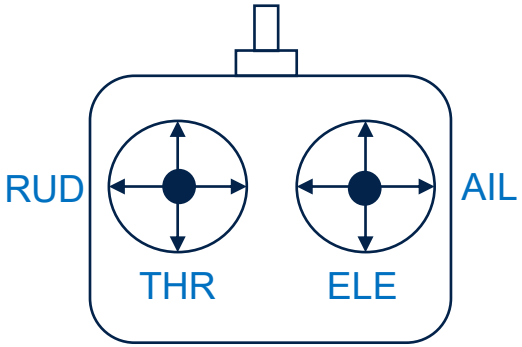
life.augmented

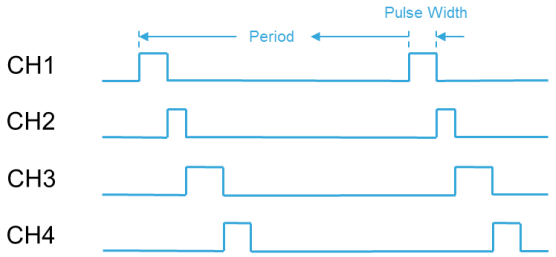
Step 3: (optional) Remocon RX module connection



- RX module should be connected to the PWM input connector.
- You should verify that transmitter is programmed in order to have the 4 RC channels assigned as per the table below.
- Current FW is compatible with PPM (Pulse Period Modulation) receiver.

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





Period: ~5..20ms
(50..200Hz)
Pulse Width: 1~2ms

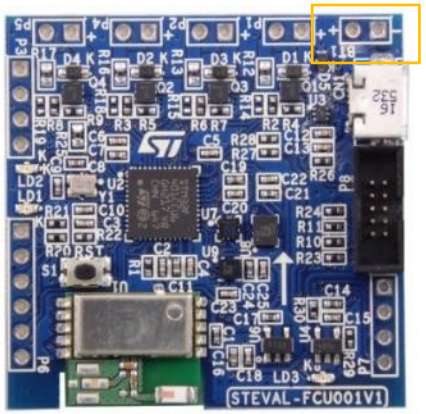
Note: The above figure refers to a Type2 remocon, commonly used for quadcopters. In case of Type1, the commands are switched.

L: Left R: Right U: Up D: Down

life.augmented

Step 4: Battery connection

BT1 connector
BAT input



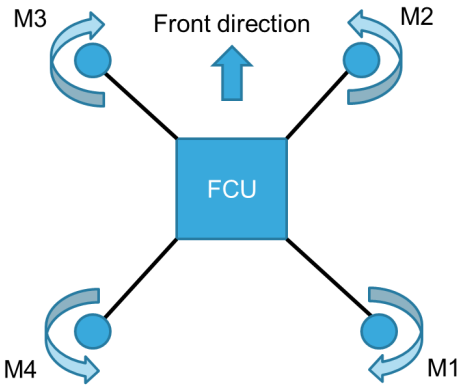
A LiPO 1-cell battery can be connected directly to connector BT1 on STEVAL-FCU001V1 board and battery can be charged via USB thanks to the on-board charger.
In case of 2-cell or 3-cell, the battery should be connected to the ESC and Vbat should be connected to the 5V (max input voltage 5.5V) generated by the BEC of the ESC.

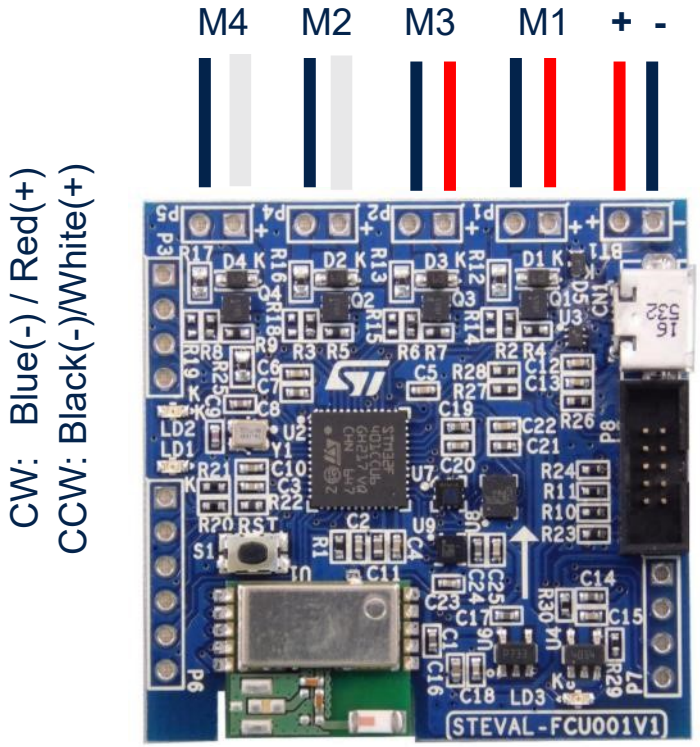
Warning: a reverse battery protection diode is not mounted, check carefully before connecting the battery!

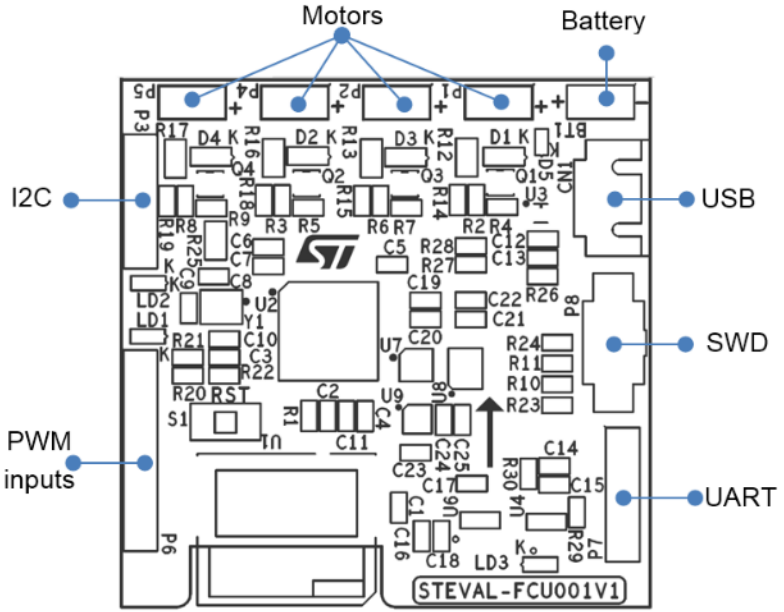
Warning: LiPO batteries can be damaged and even explode if they are short-circuited or overcharged. They should also always be kept in a safe bag when not used or during charging. Please read detailed information from battery maker or on the Internet before handling a LiPO battery.

Warning: as soon as the battery is discharged after a flight (when there is not enough power to make the drone fly), disconnect the battery from the board immediately. If the battery remains connected and is completely discharged, it will be damaged and you will no longer be able to recharge.

Resume: Motor and Battery connections

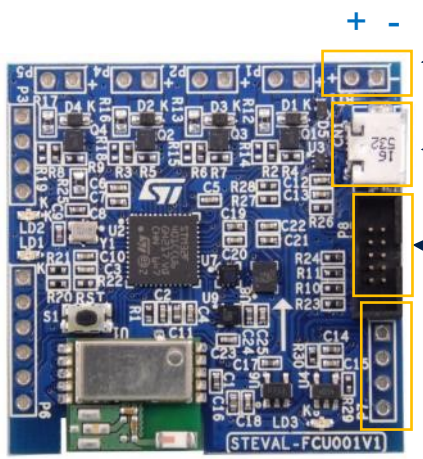






Step 5: JTAG connection and SW download

To download and debug the FW on the STM32F401 MCU, a JTAG with micro connector (SWD) should be connected, and a voltage should be supplied to the board through a 1S LiPo battery (3.7V).



1-cell battery (or 3.7V supply voltage by external power supply) must be connected.

Connect microUSB cable to avoid to discharge the battery.

Cable for ST-Link/V2 included in the STEVAL-FCU001V1 package.

(optionally) UART can be used for debug and datalog

Note: if only the microUSB cable is connected with no battery or supply voltage connected to the BT1 connector, the STM32 supply voltage may be unstable and not work properly in debugging.

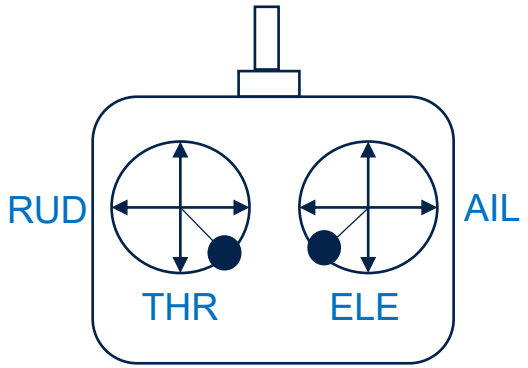
Note: In order to avoid complete LiPo battery discharge during a debugging session, always connect the microUSB cable (connected to PC or charger) to keep charging the battery.

Step 6: Calibration procedure

After startup (with STEVAL-FCU001V1 already mounted on the quadcopter), it is necessary to run a sensor calibration procedure.
Place the quadcopter on a flat surface, press the reset button and move the remocon control levers to the positions shown in the figure below.
The STEVAL-FCU001V1 will take the flat surface as reference and determine any offsets for the AHRS Euler angle.

euler_ahrs	<struct>
thx	2.7870795133E-1
thy	7.9544568433E-3
thz	0.0

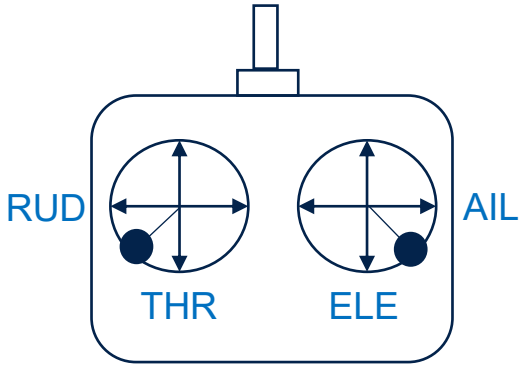
euler_ahrs	<struct>
thx	4.0701283144E-3
thy	-3.6119404833E-4
thz	0.0



Note: the same procedure could be performed by using the Smartphone App placing the levers in same position.

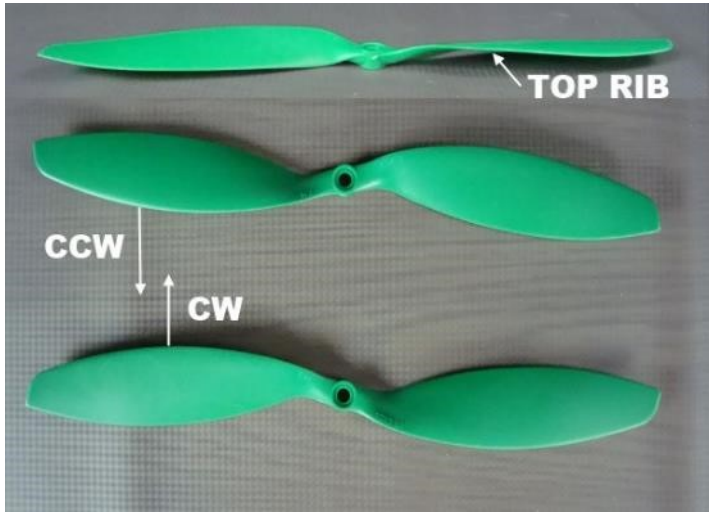
Step 7: Arming procedure

The first operational tests should be performed without the propellers mounted to verify that connections are correct, the motors function and the remocon connection is established.
For safety reasons, the drone is initially *Disarmed*: even if connection with Remocon TX is established and throttle is at full scale, the motors are intentionally *not driven* (red LED on the STEVAL-FCU001V1 blinks).
To allow flight, you must perform an *Arming* procedure: in the current FW implementation, arming is achieved by moving the remocon levers to the positions shown in the figure below for at least 2 seconds (the red LED will stop blinking and remain ON).

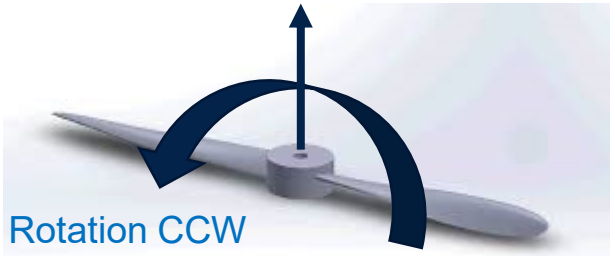


Note: the same procedure could be performed by using the Smartphone App placing the levers in same position.

Step 8: mounting the propeller



Thrust direction



Rotation CCW
Top RIB is advancing.

Care must be taken to mount the propeller blades correctly: the propeller is designed according to Bernoulli's principle, where a positive thrust perpendicular to the rotation axis is achieved when the top advances in the sense of rotation.
You can recognize CW or CCW orientation by observing the top rib direction (CCW blades are also usually marked with an "R", reversed).

Warning: Before mounting the propellers, disconnect the battery for safety.

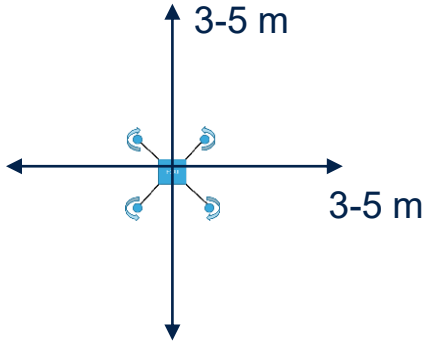
Warning: if you mount CW propellers on a CCW motor or viceversa, your quadcopter will not fly.

life.augmented

Step 9: first test flight

Proceed with your first test flight in an open area with no one in the immediate vicinity after you have followed all the safety procedures during setup and have completed the arming procedure.

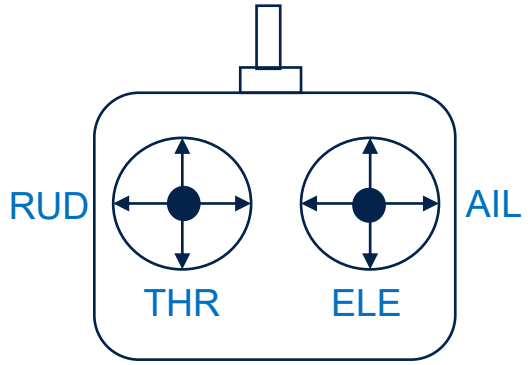
- First make the drone hover at 2m altitude and ensure that it remains stable.
- Then proceed with simple flight manoeuvres like forming a cross at 2-3 m altitude.



Troubleshooting: if the drone seems unstable and starts to oscillate, try reducing the PID proportional gain, refer to the FW description section and PID tuning tutorials on the Internet.

Appendix D:
Fine tuning and customization

R/C calibration (1/2)



Channel	Control	GPIO
CH1	AIL	PA0
CH2	ELE	PA1
CH3	THR	PA2
CH4	RUD	PA3

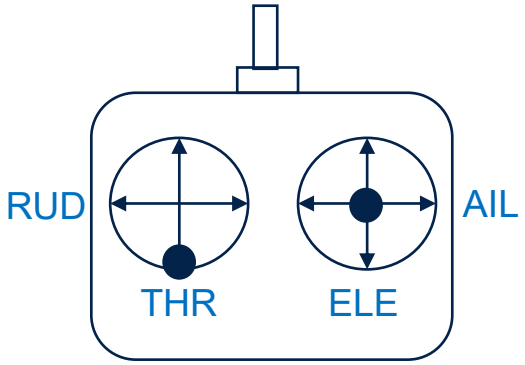
(*) Some remoconds also allow you to set the channel assignments.

After connecting a new Remocon receiver to the FCU, one of the first operations to perform is to check if the channels are connected correctly and then calibrate the offset for each channel.

1) ***Remocon TX and RX binding***: at first it is better to ensure that the TX and RX *binding* operation is successful, otherwise no signals will be generated by RX receiver to the FCU. For this operation, please refer to the remocon manufacturer instructions.
2) ***Channels connection***: after connecting the RX receiver to the FCU, you must verify that the channel connections: by moving the joysticks, verify that the THR, RUD, … are in line with the variables gAIL, gELE, gTHR, gRUD (in *rc.c*). In case of incorrect connections, it is possible to modify the HW connection or by SW(*) in the function:

```
void update_rc_data(int32_t idx)
```

R/C calibration (2/2)



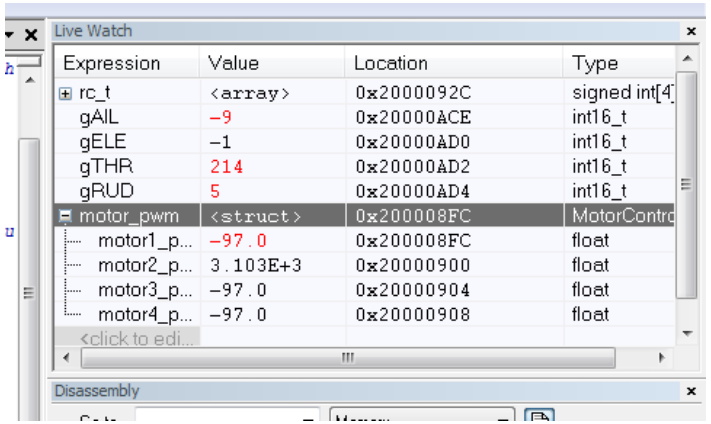
3) ***Channels offset calibration***: it is necessary to adjust the offset for each channel(*).
   a. Position the left joystick to the center-bottom
   b. Position the right joystick in the center
   c. Take note of the values of the variables gAIL, gELE, gTHR, gRUD (in *rc.c*). If the remocon is calibrated correctly, each value should be *close to zero*.
   d. If values are not close to zero, adjust the offset by modifying the value of the following constants in *rc.h*:

```
#define AIL_MIDDLE  6269
#define ELE_MIDDLE  6051
#define RUD_MIDDLE  5949
#define THR_BOTTOM  4437
```

(*) This operation is usually done by the PC GUI of commercial FCU available in the market.

life.augmented

Motor connection check



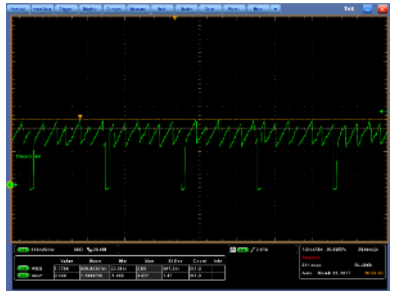
Wrong connection situation.

It is easy to connect the motors to the FCU incorrectly (correct configuration described in "*Step2 – DC motor connection*" section).

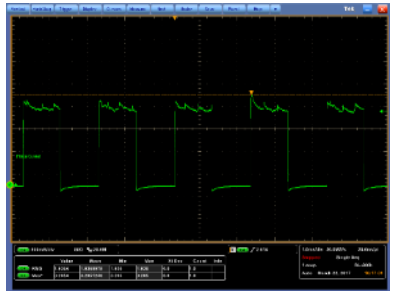
This can be verified by increasing the Throttle value (just enough to spin the motor but not to lift the quadcopter), and forcing an inclination. The figure shows what happens when there is a *wrong* connection: the motor in lower position is stopped while the one in higher position is spinning >> it should be opposite to stabilize the drone.

life.augmented

DC motor configuration – MOS Current



DC motor current Full Throttle (Rg = 1Kohm)



DC motor current Full Throttle (Rg = 0ohm)

It may be necessary to perform a small adjustment to the gate resistor to MOSFET according to the DC motor used.
If Rg is too large, the MOSFET may not switch on properly and available current for DC motor may be sufficient to lift the quadcopter.
At the same time, the current capability to switch on the MOS is linked to the STM32 GPIO current capability.

Note that the current needed may vary with the DC motor used, size of propeller and overall weight of the drone. For a small minidrone frame, a default Rg of 100 ohm is the preferred all-round value.

Stabilization PID tuning

```
#define ROLL_PID_KP1      3
#define ROLL_PID_KI1      0
//#define ROLL_PID_KP2    800 /* default */
#define ROLL_PID_KP2      100 /* test minidrone */
//#define ROLL_PID_KI2    400 /* default */
#define ROLL_PID_KI2      100 /* test minidrone */
#define ROLL_PID_KD2      10
…

#define PITCH_PID_KP1     ROLL_PID_KP1
#define PITCH_PID_KI1     ROLL_PID_KI1
#define PITCH_PID_KP2     ROLL_PID_KP2
#define PITCH_PID_KI2     ROLL_PID_KI2
#define PITCH_PID_KD2     ROLL_PID_KD2
```

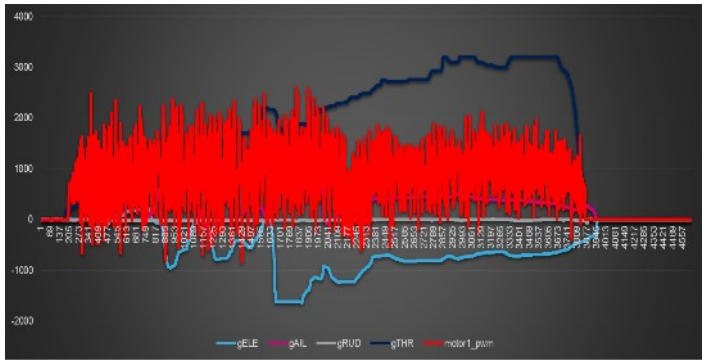
For every configuration of quadcopter frame/motor/FCU, it is necessary to fine tune the PID parameters (in *flight_control.h*) to find the best settings that allow the drone to fly with good stability and responsiveness.

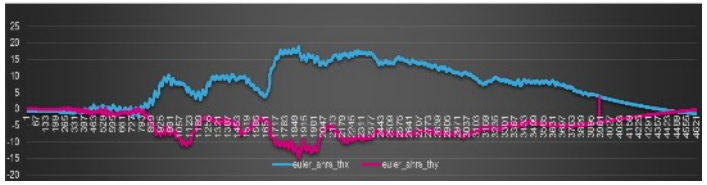
You must first check that the drone *hovers* (only Throttle command applied from Remocon) without any oscillations.

1. start with low value of KP (same value for x and y axis if quadcopter design is symmetrical and mechanically well balanced, as in most cases) and smaller value of KI.
2. Increase KP without experiencing oscillations.
3. Increase KI value without experiencing oscillations.
4. You should keep the same PID value for roll and pitch in most cases.

Note: In most of use cases (except racing drones in which very high response is needed) it is better to keep the KD value at zero to ensure good stability.

Datalog with UART



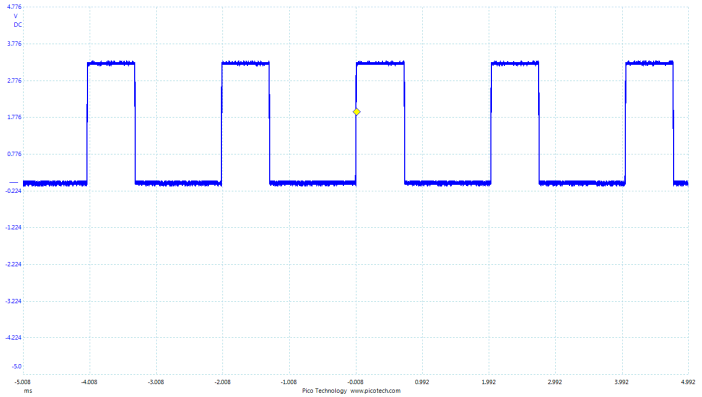


During test and debugging, could be useful to make a data log and plot of certain variables like Remocon input, MEMS raw data, Euler angle, Motor PWM, …

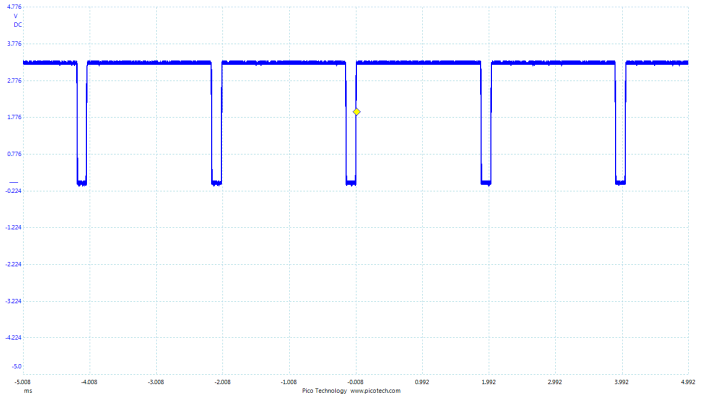
There are three way to do this:

1. Connect the board to a PC through an USB to UART converter (namely Virtual COM port). There is an UART on the board, see Step 5 in Assembly Section.
2. By using a `printf` within the code and using the console offered by IDE. Note: this could introduce a delay in the program flow impacting the performance (please refer to Timings in the Total Data Flow diagram).
3. Use the on-board BLE module, but the user should add a telemetry to the actual communication protocol.

External ESC FW fine tuning



Motor PWM on pin2 or P1, P2, P4, P5
(THR to minimum value)



Motor PWM on pin2 or P1, P2, P4, P5
(THR to maximum value)

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

Most ESCs accept an input signal with frequency varying from 50 to 400Hz, and the Ton time of the PWM is proportional to the motor speed. The *minimum Ton time* for which the ESC is *armed* (when not armed usually the ESC emits a particular alarm beep) must also be checked.
It may 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*)

life.augmented

Accelerometer sensor fine tuning – Low vibration frame

If the frame does not experience high vibration from the motors, it is also possible to use alternative settings for the MEMS accelerometer:

- Accelerometer full-scale selection FS ***default ±2g >> set to ±4g***
- Analog Filter Bandwidth ***default 1.5KHz >> set to 400Hz***
- Output data rate and power mode selection ODR ***default Power down mode >> set to 1.6kHz***
- Composite filter input selection ***default ODR/2 >> set to ODR/4***
- Accelerometer low-pass filter LPF2 selection ***default >> set to Low pass filter enabled @ ODR/50***

```
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_4);
/* ODR/4 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_DIV4);
/* Low pass filter @ ODR/50 */

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 | 0x01;
/* Set LSB to 1 >> Analog filter 400Hz*/
BSP_ACCELERO_Write_Reg(LSM6DSL_X_0_handle, 0x10, tmp_6axis_reg_value);
```

Fine tuning for sport/racing use

- In the default settings, the maximum PITCH and ROLL angles are limited to 20 degrees, while the maximum YAW is limited to the max rotation speed of 100 deg/s. These values are good for beginners to gain confidence with the quadcopter.
- For more expert pilots (and for racing/sport use), it is possible to increase these values in the rc.h:

```
/ Maximum roll/pitch 20deg
#define PITCH_MAX_DEG   20
/* change to 45 (angle degree) for higher forward/backward speed */
#define ROLL_MAX_DEG    20
/* change to 45 (angle degree) for higher right/left speed */
#define YAW_MAX_DEG     (60.0*SENSOR_SAMPLING_TIME)
/* change to 120 or 180 for higher rotation speed */
#define YAW_MIN_RAD     0.0872
#define EULER_Z_TH      600
```