Gyro compensation in OSX on RP2040

1. introduction

OSX already includes all the code to manage an IMU when the GY86 module is installed.
OSX computes a best estimate of the plane attitude and collects the gyro rates values on 3 axis.
OSX allows to stabilize a camera using 2 servos and the above IMU data…

It was really tempting to add a plane gyro stabilization 😊

The mod I propose here allows to stabilize a standard 3 axis plane (controlled with AILERONS + ELEVATOR + RUDDER). All this in addition of all native functions of OXS…

There are 2 modes of stabilization:

- RATE damping : the gyro controls roll and/or pitch axis and compensates any increase of angular rate on those axis
- Attitude HOLD : the gyro attempts to lock the attitude in roll and/or pitch axis

Each axis is independent of the other. (Eg: one axis can be stabilized in HOLD and the other in RATE or any other combination).

Please note that all this is still experimental. It has been tested in flight with success, but I would advice not to install it on expensive planes before more extensive test campaigns have been passed!

So far, control loop algorithms are managed with a P law (no PID, just P).
This being said, it behaves very well in flight during windy conditions. In RATE mode the plane is really more stable. The HOLD mode works too and looks promising, but I need further tests in flight to experiment with the gains and assess stability …

2. functionalities

Activating the gyro stabilization can be either done from CONFIG.h or the gyro stabilization mode can be selected in flight using two 3 positions switch (switch “mode” one for ROLL, one for PITCH).

If activation is programmed from the CONFIG.h file, it is permanent until a new firmware is loaded which disables this function. From CONFIG.h only the RATE mode is accessible.

If activation is done from a switch, it behaves like this:

- switch UP (MIN value): gyro RATE
- switch middle (neutral value): gyro OFF
- switch down (MAX value): gyro HOLD

Those compensations apply on:

- One or two AILERON(s) servo(s)
- One ELEVATOR servo

The gains in the control loops can be trimmed:

- Inside CONFIG.h, for RATE mode, when no switch is used to activate the function (a compilation is needed)
- On ground, simply adjusting the max/min values of the switches “mode”: min value stores the RATE gain, max value stores the HOLD gain. As we get independent switches to set ROLL and PITCH compensation, the gains are independent too.
- in flight on a EDGE-TX radio, using a “program” switch, a potentiometer and a LUA script which will encode the gains (signed values) of each mode and each axis and store them inside the min and max values of the associated channel.

Playing with config file, the direction of the second AILERON servo can be reversed (a compilation is needed)

The effect of the gyro compensation is lowered according to the stick position: the more the stick commanding the axis is far from its neutral position, le lower compensation is applied. The stick compensation can be trimmed inside the config file (a compilation is needed). The default value of 40% means that when the stick value is 40% then the gyro has no effect. At neutral, the gyro has a gain*100% effect.

TO DO : VTAIL and DELTA planes are not yet supported…

3. Configuration

In the config file we get the following parameters. (see extract in appendix 1)

a) Servos management

The GYRO_PITCH/ROLL/ROLL2_CHANNEL(s) are the channels programmed in the TX to move PITCH/ROLL/ROLL2 servos. OXS will use them to compute a neutral position of the servos and to estimate the sticks commands (or rather servo motion commands if some mixings are used between sticks…).

OXS will then add to the TX servo command the compensation required on each axis and will inject the PWM value on a RP2040 pin.

`GYRO_PITCH/ROLL_CHANNEL 2:` when uncommented, the gyro compensation is applied on the axis. The value of the channel shall be set to the same channel used in the TX to activate the axis servo.

The way the gyro compensation is implemented supposes (mandatory) that the channel used to activate the servo axis is also used as PWM output channel to activate the servo associated to the axis.

Example :

Suppose `#define GYRO_PITCH_CHANNEL 2, (often the case for ELEVATOR in AETR)`

This means that you have attached the ELEVATOR servo to channel 2 and you have to connect it to a RP2040 PWM pin that will output the channel 2
Eg : C2=8, then plug the ELEVATOR servo to pin 8

For AILERONS motion, planes may have 2 servos one on each wing. The first one shall be connected to `GYRO_ROLL_CHANNEL,` the second one to the channel used by TX as AIL2.
To set the direction of compensation, start with AIL one (set the gain positive/negative will reverse the direction). Then once done with AIL1, check if second servo is moving in the right direction. If not set GYRO_ROLL2_SIGN to -1

```
#define GYRO_ROLL2_SIGN -1                  // set it to -1 to reverse the
AILERON2 compensation
```

b) Gains management

Setting the gains

When using a switch to activate the gyro compensation on one axis, you have to uncomment the associated CONTROL_CHANNEL (`GYRO_PITCH/ROLL_CONTROL_CHANNEL)` which is the channel attached to the switch.

Example :

Suppose `#define GYRO_PITCH_CONTROL_CHANNEL 13,` the switch to control the pitch axis gyro mode is connected to channel 13.

Remember that

RATE mode is activated when OXS reads negative values of the switch
HOLD mode is activated when OXS reads positive values of the switch

Gains of the control loops are stored in the MIN/MAX values of the channel used to activate the gyro mode.
MIN/MAX values (at least in EDGE-TX) are programmed in percent in the range [-100, +100%]

Setting the MIN value will program the RATE mode gain
Setting the MAX value will program the HOLD mode gain
As gains can be positive or negative we chose to encode the gains with these laws:

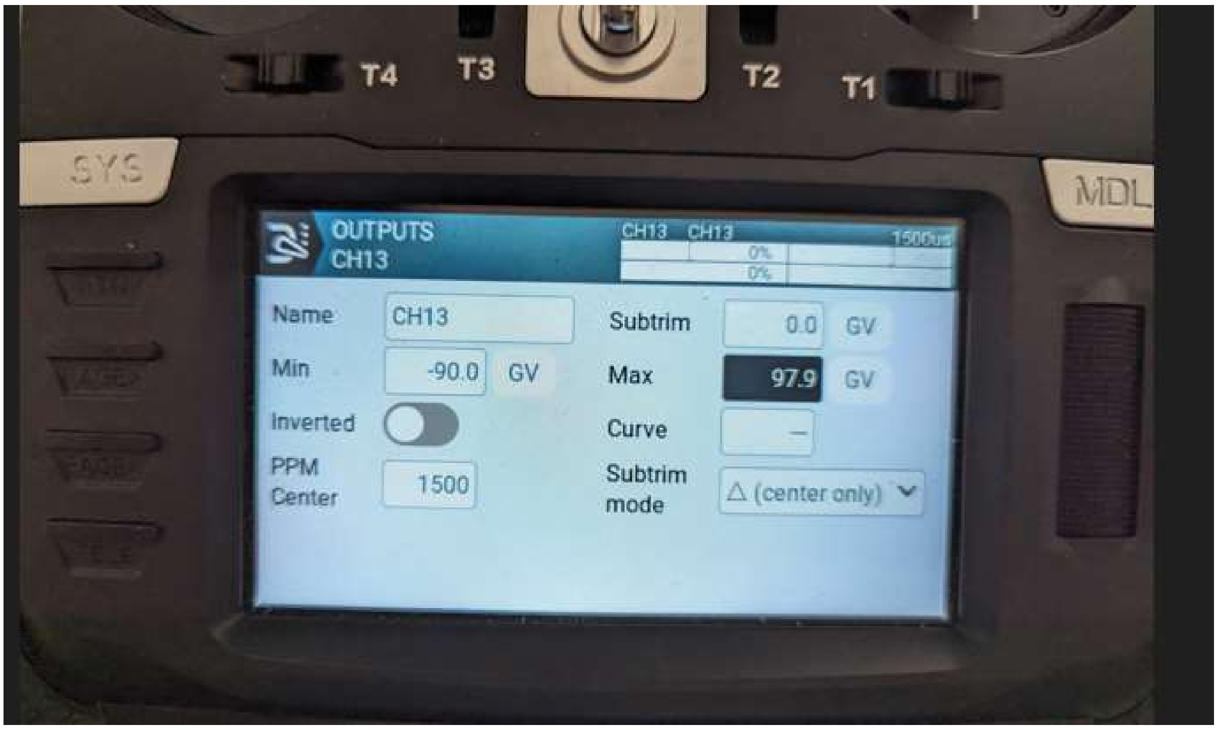
MIN value = - 80% + RATE_gain

MAX value = +80% + HOLD_gain

RATE_gain and HOLD_gain are expressed in the range [-20, 20]

Eg:

- we want to code a gain of 10 for PITCH RATE mode : set GYRO_PITCH_CONTROL_CHANNEL MAX value to 80+10 = 90
- we want to code a gain of -15 for PITCH RATE mode : set GYRO_PITCH_CONTROL_CHANNEL MAX value to 80-15 = 65
- we want to code a gain of -5 for ROLL HOLD mode : set GYRO_HOLD_CONTROL_CHANNEL MAX value to -80-5 = -95


OUTPUTS
CH13

Name	CH13	Subtrim	0.0 GV
Min	-90.0 GV	Max	97.9 GV
Inverted		Curve	—
PPM Center	1500	Subtrim mode	△ (center only)

- On this picture, gains for ROLL (here channel 13 switch) are:
  -90 + 80 = -10 for RATE
  97.9-80 = 17.9 for HOLD

To sum up, to encode the gain of RATE with a value of 20 steps and a sign so between [-20, 20], we will use the range [-100%, -60%] for the MIN value, a value of -80% means gain = 0 a value of -100% means gain = -20, a value of -60% means gain = +20

To encode the gain of HOLD with a value of 20 steps and a sign so between [-20, 20], we will use the range [60%, 100%] for the MAX value, a value of 80% means gain = 0 a value of 100% means gain = 20, a value of 60% means gain = -20

The same logic applies to set the RATE mode gains in CONFIG.h in case no switch is used to activate the gyro compensation.

```
#define GYRO_PITCH/ROLL_RATIO  20           // [-20,20] Ratio to use when
GYRO_PITCH/ROLL_CONTROL_CHANNEL is undefined;
```

Just change the value between -20 and +20, a negative sign will reverse the compensation. The bigger the absolute value, the more compensation is given to the axis. Remember that only RATE mode can be programmed this way, therefore OXS will compute an “equivalent of MIN value” adding -80% to the GYRO_PITCH/ROLL_RATIO

It is much more powerfull to use switches to activate the gyro compensations. You can program this manually on your TX as seen above or, cherry on the cake, with a EDGE-TX radio, set it **in flight** using a simple LUA script (FUNCTIONS scripts)

The script shall be launched by a switch (here SC) activating a Special Function to launch the script. The pot (here S2)

```
local function run()

    mytable = model.getOutput(12) -- read output settings for channel 13 (12 =
13-1)

    switchC = getValue('sc')  -- read the switch C value
    value = getValue('s2')*20/1024  --  read the pot S2 and convert it in
values in  +/- 20%

    if switchC > 10 then  -- set the gyro HOLD gain
        value = (value + 80)
        mytable.max = value*10  -- mytable.max is coded with a LSB = 1/10 % --
max (number) Maximum % * 10
    model.setOutput(12, mytable)
    -- model.setGlobalVariable(1, 0, value)
    end
    if switchC < 10 then                  -- set the gyro RATE gain
        value = (value -80)
        mytable.min = value*10
    model.setOutput(12, mytable)
    -- model.setGlobalVariable(2, 0, value)
    end

end
```

Appendix 1 : config.h

The other parameters are not yet used by the code (painted in grey)

```
// -------------- gyro stabilizer ----------------------------------------
// uncomment GYRO_PITCH_CHANNEL and/or GYRO_ROLL_CHANNEL if you want to
stabilize the plane on those axis

#define GYRO_PITCH_CHANNEL 2            // Channel used to control the PITCH
axis (ELE) from radio used as input then output to servo by OXS (2 if AETR)
#define GYRO_ROLL_CHANNEL 1             // Channel used to control the ROLL
axis (AIL) from radio used as input then output to servo by OXS (1 if AETR)
#define GYRO_ROLL2_CHANNEL 5            // Channel used to control a second
aileron; uncomment to activate - ROLL axis (AIL2) from radio used as input
then output to servo by OXS (usually 5 if AETR)
//#define GYRO_YAW_CHANNEL 4              // Channel used to control the YAW
axis (RUD) from radio used as input by OXS (usually 4 if AETR)

// uncomment GYRO_PITCH/ROLL_CONTROL_CHANNEL to control de gyro mode and gain
on this channel
//One switch is needed per axis to be controlled, so that axis are fully
independent
#define GYRO_PITCH_CONTROL_CHANNEL 13   // Channel used to control the
ELEVATOR (pitch) on OXS; uncomment to activate the gyro pitch stabilization
using a switch
#define GYRO_PITCH_RATIO  20           // [-20,20] Ratio to use when
GYRO_PITCH_CONTROL_CHANNEL is undefined;
                                        //increase/decrease the value in case
of under/over stabilization
                                        // change the sign if compensation
goes in the wrong direction
#define GYRO_PITCH_MAX 100              // adapt upper limit of servo travel
(should normally be the same value as on TX)
#define GYRO_PITCH_MIN -100             // adapt lower limit of servo travel
(should normally be the same value as on TX)

#define GYRO_ROLL_CONTROL_CHANNEL 12    // Channel used to control the AILERON
servo (roll) on OXS; uncomment to activate the gyro roll stabilization using a
switch
#define GYRO_ROLL2_SIGN 1               // set it to -1 to reverse the
AILERON2 compensation
```

```
#define GYRO_ROLL_RATIO  20            // [-20,20] Ratio to use when
GYRO_ROLL_CONTROL_CHANNEL is undefined;
                                        //increase/decrease the value in case
of under/over stabilization
                                        // change the sign if compensation
goes in the wrong direction
#define GYRO_ROLL_MAX 100               // adapt upper limit of servo travel
#define GYRO_ROLL_MIN -100              // adapt lower limit of servo travel

#define GYRO_STICK_ATTENUATION 40      // range of stick inputs in % where
gyro compensation is applied (linear decrease, closest to center more effect)

//#define GYRO_VTAIL                     // uncomment to get the VTAIL mixing
(TODO)
//#define GYRO_DELTA                     // uncomment to get the DELTA mixing
(TODO)

//Note:  when a channel is used to adjust a ratio (for gyro pitch or roll),
the ratio can vary from -100 (channel = 1800 µsec) up to +100 (channel =
2000%)
//       the sign of the ratio defines the direction of the compensation.
//       Setting the channel on 0% disables the compensation. This can e.g. be
done using a switch on the TX
//       setting the channel at 1100µsec +/-100µsec activates the angular rate
gyro stabilization
//       setting the channel at 1800µsec +/-100µsec activates the attitude
hold gyro stabilization
```

Appendix 2: impacts on code modules

Code has been forked from `VERSION "2.8.26"`

Added lines are traced with : `// added aeropic`

Impacted modules are:

Tools.h :

    (added GYRO_X/Y_ID(s))

Main.cpp

    Entry type GYRO_X/Y_IDs

Mpu.cpp

    Sent2core0 GYRO_X/Y_IDs

SBUS_out_PWM.cpp

    All the stuff is here !

Param.cpp

    gyroX and gyroY printed

config.h

    all required defines (see above)