新唐多旋翼开发应用

Product No.	**M4** 飞控应用
Function	开发软件及程序库说明

File Information

Name	**Nuvoton Fly-Controller Development Kit(简).pdf**
Project	**Cortex M4** 应用开发
Function	飞控软件及程序库说明
Purpose	
Author	沈子岚 **(tlshen)**

Revision History

Revision	Date	Comments
1.0	15/2/5	初版

Release Note:

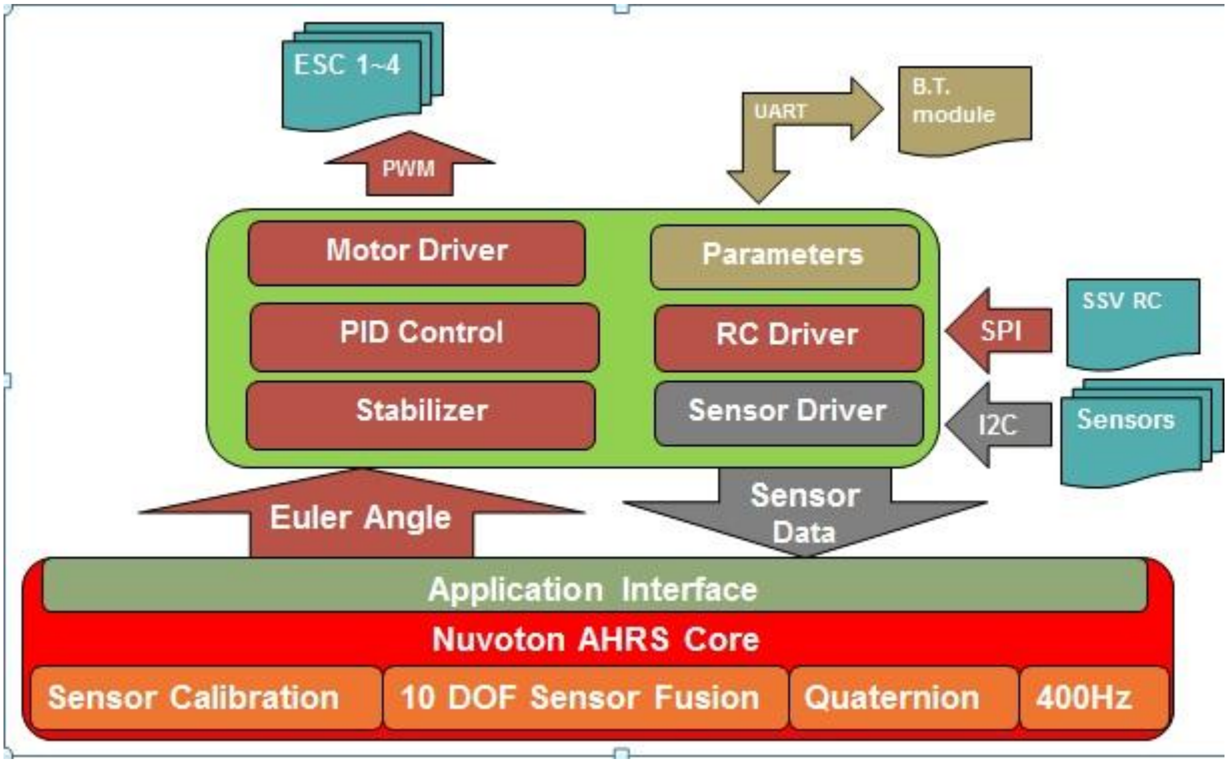
Revision 1.0	**(1st Draft Version)**

1. 多旋翼飞控开发环境总览

- 韧体开发环境与工具
  - ARM MDK uVision V4.60
  - Nuvoton Nu-Link ice adapter

- 程序软件开发包
  - 开发包压缩文件 NvtFly.7z
  - 开发说明文件 – NvtFly\Document
  - FCA 飞控端软件 - NvtFly\M451\M451BSP\NvtFly\FlyController\FUSIONSDK
  - FCA 2.4G 遥控端软件– NvtFly\NvtFly_2.4GTx\Trunk\Transmitter\SSV_TX
  - PC 端调适软件 – NvtFly\AhrsAP
  - AHRS 头文件 - NvtFly\M451\NvyFly\FlyController\AHRSLib\include
  - AHRS 实体库 - NvtFly\M451\NvyFly\FlyController\AHRSLib\obj

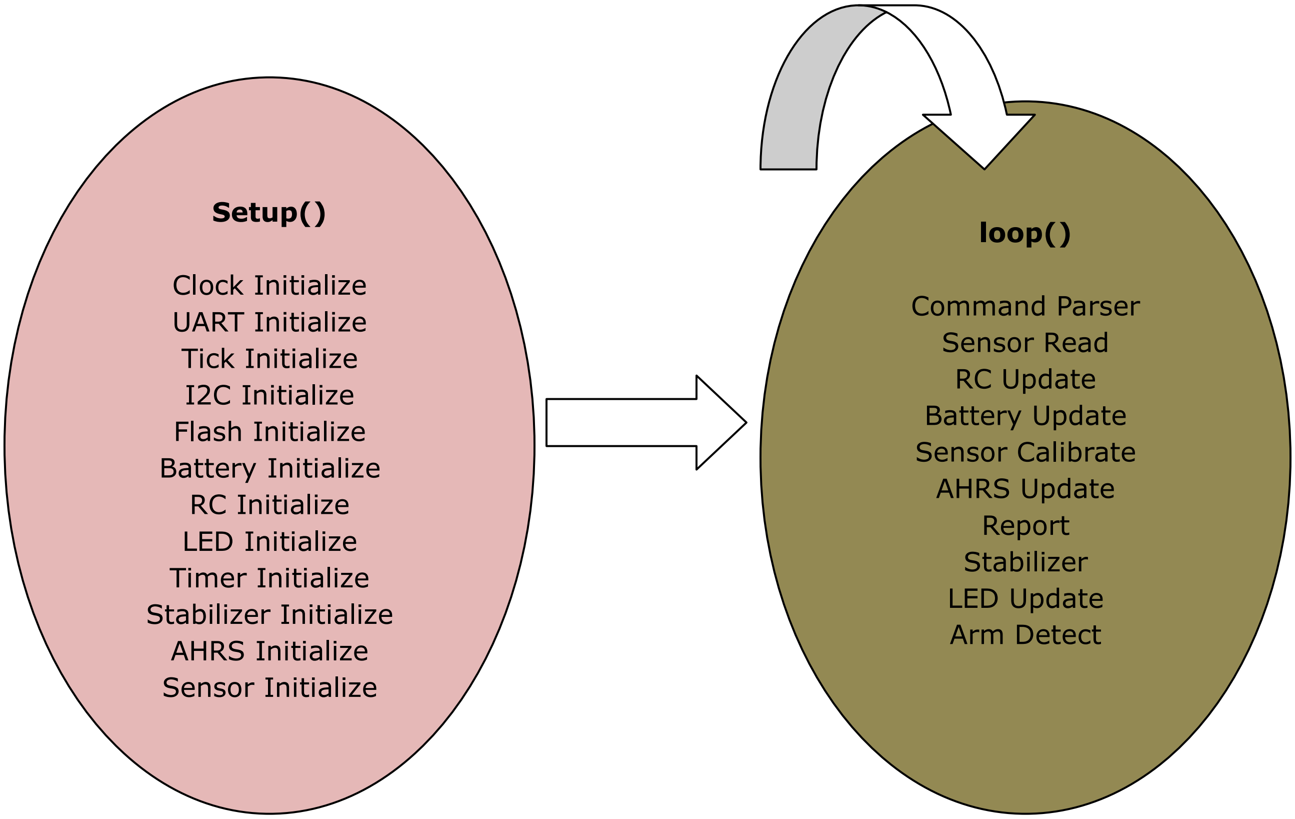
- 程序软件架构
  - 飞控应用层 (FCA), 提供使用者一个以新唐 Cortex-M4 为核心的多旋翼飞控架构并支持用户自行配置自有的传感器,无线收发控制器,电机,自稳核心以及其他各样用户自定义的功能延展.
  - Attitude and head Reference System library (姿态方向参考系统 AHRS) ,一个专注于传感器融合的数据库,支持九轴融合并以精准计算的四元数为运算核心.FCA 透过函式库提供的应用接口, 可以进行传感器校正以及传感器融合的尤拉角输出.


ESC 1~4
PWM
UART
B.T. module
Motor Driver
Parameters
PID Control
RC Driver
Stabilizer
Sensor Driver
SPI
SSV RC
I2C
Sensors
Euler Angle
Sensor Data
Application Interface
Nuvoton AHRS Core
Sensor Calibration
10 DOF Sensor Fusion
Quaternion
400Hz

新唐飞控软件架构图

2. FCA 功能与运作

- **系统初始化 – Setup()**
  - ■ Setup() 作为一个平台的初始程序,包含各个周边的初始化, 飞控核心初始化以及 AHRS 库的初始化.
  - ■ Setup() 必须在 FCA 起始运作前进行呼叫,使用者亦可在此扩增自行定义的初始化程序.
- **主控循环 – loop()**
  - ■ 一个无穷循环,为 FCA 的主控程序,内部依序执行飞控相关的子程序.
  - ■ CommandProcess() – 处理飞控序列阜命令.
  - ■ SensorRead() – 读取传感器回报值
  - ■ ssv_rc_update() – 使用南方硅谷的 2.4G 传输时, 更新来自于接收端的遥控信号.
  - ■ UpdateBattery() – 透过 ADC 脚的读值侦测电池电量.
  - ■ SensorDynamicCalibrate() – 传感器进行动态校正.
  - ■ computeRC() – 转换接收端的遥控信号为飞控的控制信号.
  - ■ armDetect() – 飞行器起飞状态检测,包含翻覆时的电机停止检测..
  - ■ nvtUpdateAHRS(SENSOR_ACC|SENSOR_GYRO|SENSOR_MAG) – 使 AHRS 库进行姿态倾角计算.
  - ■ report_sensors() – 显示飞控相关信息如传感器,飞控核心,遥控端,电机等,显示内容可由序列阜命令
  - ■ stabilizer() – 自稳主程序,融合来自 AHRS 库的传感器融合姿态与来自接收器的使用者方向控制,可产生相对应的电机驱动行程量,使得飞行器在任何时间都能自行维持稳定..
  - ■ UpdateLED() – 将飞行器状态透过 LED 显示,包含上锁,解锁,飞行模式等.


Setup()
Clock Initialize
UART Initialize
Tick Initialize
I2C Initialize
Flash Initialize
Battery Initialize
RC Initialize
LED Initialize
Timer Initialize
Stabilizer Initialize
AHRS Initialize
Sensor Initialize
loop()
Command Parser
Sensor Read
RC Update
Battery Update
Sensor Calibrate
AHRS Update
Report
Stabilizer
LED Update
Arm Detect

FCA 运作概图

3. AHRS 功能与运作

- **系统初始化阶段呼叫**
  - nvtAHRSInit() ,AHRS 库的初始程序,必须要在 AHRS 运作前呼叫.
  - nvtMilisecondTick(), 作为 AHRS 库的定时器,由应用端每毫秒呼叫一次.

- **传感器初始设定阶段呼叫**
  - FCA 于操作 ACC 前须藉由 AHRS 库进行 ACC 设定, FCA 须设定下列参数于 AHRS 库,包含比例,偏移及重力比例. FCA 可藉由 AHRS 库函数 nvtSetAccScale(), nvtSetAccOffset() 及 nvtSetAccG_PER_LSB() 完成设定.
  - FCA 于操作 GYRO 前须藉由 AHRS 库进行 GYRO 设定, FCA 须设定下列参数于 AHRS 库包含比例,偏移及角速度比例. FCA 可藉由 AHRS 库函数 nvtSetGyroScale(), nvtSetGyroOffset() 及 nvtSetGYRODegPLSB() 完成设定.
  - FCA 于操作 MAG 前须藉由 AHRS 库进行 MAG 设定, FCA 须设定一个大小为 10 的一维参数矩阵提供 AHRS 库做为 MAG 校正矩阵.另外需设定一个高斯比例系数提供 AHRS 库一个基于传感器读值的高斯转换. FCA 可藉由 AHRS 库函数 nvtSetMagCalMatrix() 及 nvtSetMagGaussPLSB() 完成设定.

- **传感器融合阶段呼叫**
  - 传感器读值输入 – FCA 处里传感器融合以前需先输入传感器读值于 AHRS 库,以下为各传感器的输入呼叫函式:
    - ✧ nvtInputSensorRawBARO() - BARO 传感器读值输入.
    - ✧ nvtInputSensorRawACC() - ACC 传感器读值输入.

  - ✧ nvtInputSensorRawMAG() - MAG 传感器读值输入.
  - ✧ nvtInputSensorRawGYRO() - GYRO 传感器读值输入.

- ■ *AHRS 姿态顷角计算* – 当各个指定的传感器读值已经输入 AHRS 后, FCA 调用库函式 nvtUpdateAHRS() 启动传感器融合程序,AHRS 库以四元数运算传感器读值并计算出 Euler 姿态顷角.

- ■ *AHRS output* – 当 AHRS 库完成姿态顷角计算, FCA 可调用库函数 nvtGetEulerRPY()取得 Euler 姿态顷角, 包含翻滚(Roll),俯仰(Pitch),及偏摆(Yaw). 另外 FCA 也可调用库函数 nvtGetVelocity() 取得三轴的速度信息,通常使用于定高程序的辅助计算.

- ● **传感器校正阶段呼叫**

AHRS 库提供相关的传感器校正程序,FCA 可藉由校正程序对各传感器进行校正以取得更为精确的姿态顷角信息,有助于飞行器的稳定 ,若传感器未完成校正程序时,AHRS 库仍可以正常运作.

- ■ ACC 校正 – ACC 有两种校正方式, 一种称为 “Fast Z” 校正,另一种称为“Full Face” 校正. FCA 于 ACC 校正前需要先呼叫库函式 nvtCalACCInit() 以初始化 ACC 校正程序. 进行“Full Face” 校正时 FCA 需要输入六个面的 ACC 读值 (+X, -X, +Y, -Y, +Z, -Z),此时的 ACC 读值的理想值应为地球重力值(1G),读值与 1G 的差即为误差,AHRS 库校正程序目的即是消除误差. 校正程序为 ,为每一面独立取得感测值,以第 N(0 ~5)面为例,
  - ✧ FCA 读取 ACC 报值并输入 AHRS 库, 输入 ACC 校正库函式为 nvtCalACCBufferFill(N),
  - ✧ 检查回传值, “STATUS_BUFFER_FILLED” 回传表示单面校正完成.
  - ✧ 翻下一面(未取过值),并重复此程序.
  - ✧ 回传值“STATUS_CAL_DONE” 表示六面皆取值完成,并且 AHRS 库完成”Full Face”校正运算.

  “Fast Z”运算提供 FCA 一个较为简略但快速的计算方式,只针对 Z 轴做单面的校正,当指定的输入

面编号为’0’时,AHRS 库会片面启动校正程序,并计算出结果.此时 FSA 可终止校正程序.

无论是”Fast Z” 或是”Full Face”, FCA 可调用库函数 nvtGetAccOffset() 及 nvtGetAccScale()取得 AHRS 库的 ACC 校正参数.

- GYRO 校正 – GYRO 校正有两种类型，一种称为“Scale” (比例)校正，另一种称为“Drift”(飘移)校正. 进行比例校正前 FCA 需呼叫库函式 nvtCalGyroInit(). “Scale” 校正是为了补偿 GRYO 在转动时的角速度误差,经由比例参数对于三轴读值乘积的补偿加以修正此误差. 校正程序以 Z 轴为例,将传感器静止平放,重复下列步骤

  - ✧ 读取 GYRO 感测值并输入 AHRS 库
  - ✧ 呼叫库函式 nvtGyroScaleCalibrate()
  - ✧ 将传感器以 Z 轴为中心缓慢平稳旋转
  - ✧ 传感器实际旋转角度达到 1080 度(3 圈)时停止旋转,静置并等待回传值“STATUS_GYRO_CAL_DONE”,完成校正.若传感器旋转角度未达 3 圈,回到步骤一.

  X 轴及 Y 轴可依此程序类推

  “Drift”飘移校正是为了补偿 GYRO 静止时产生的角速度误差,”Drift”校正程序须将传感器静止平放,重复步骤

  - ✧ 读取 GYRO 感测值并输入 AHRS 库
  - ✧ 呼叫库函数 nvtGyroCenterCalibrate() 并检查回传值，回传值为“STATUS_GYRO_CAL_DONE”即完成校正程序,否则回步骤一.

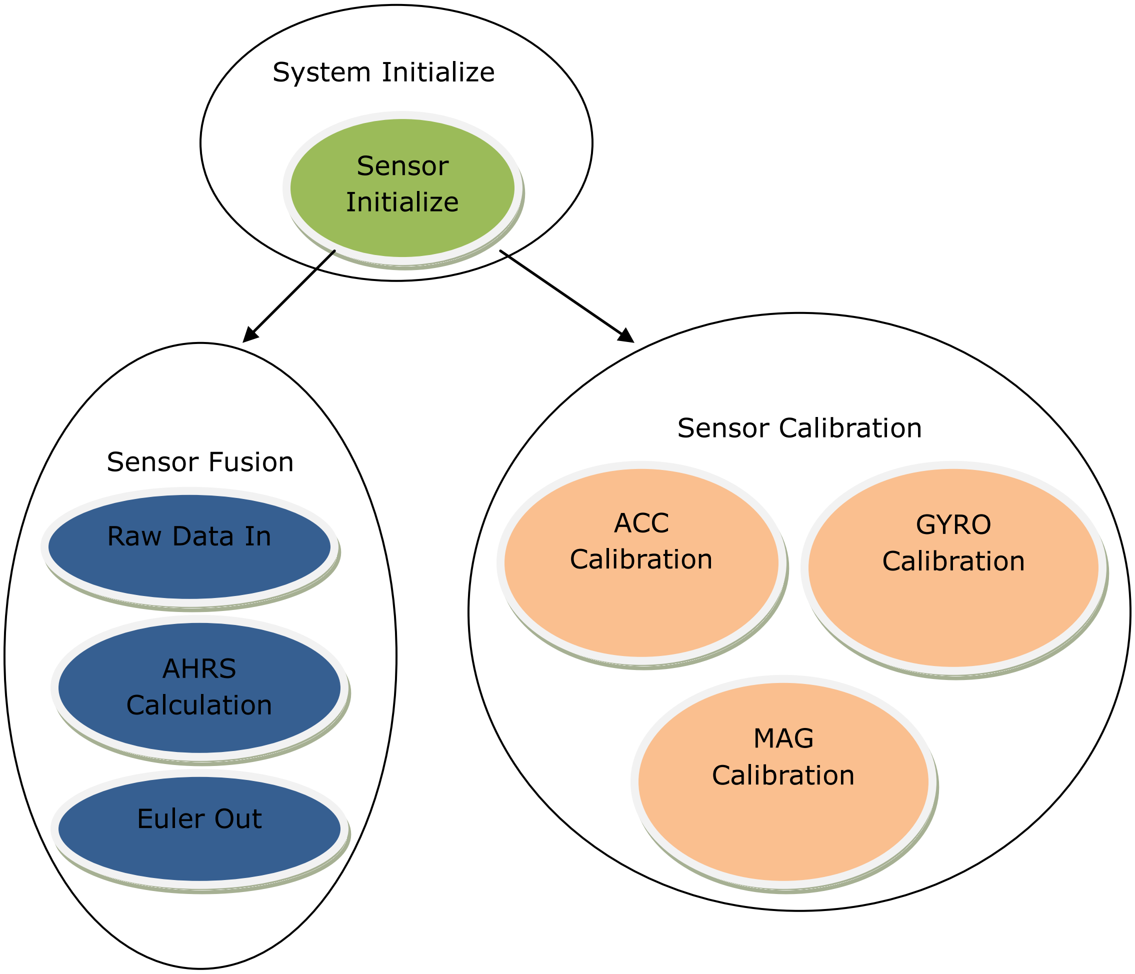
  FCA 可呼叫库函数 nvtGetCalibratedGYRO()取得校正后的 GYRO 值

- MAG 校正 – 进行 MAG 校正前,FCA 需要首先呼叫库函式 nvtCalMAGInit() ,之后拿起传感器分别以三个轴(X,Y,Z)为轴心为主,进行任意角度旋转(之后也可以不以固定轴心任意旋转),目的是完成一个类球体的均匀高斯地磁场取样,取样愈平均,校准误差越小.MAG 校准程序如下,

  - ✧ 读取 MAG 感测值并输入 AHRS 库

- 呼叫库函式 nvtCalMAGBufferFill()
- 旋转传感器并检查回传值,回传值为“STATUS_CAL_DONE”即完成校正程序,否则回步骤一.

FCA 做完校正后须调用库函式 nvtGetMagCalQFactor() 以取得一个校正系数 Q,系数值从 255(最差) 到 0(最好). 较正系数 Q < 10 对于 FCA 是一个建议的参考值. 对于一个不满意的校正系数,FCA 可重复进行校正程序以达到期望的校正质量.



AHRS 运作概图

4. AHRS 库接口函式规格

- **AHRS 表头档及函式库**
  - “AHRSLib.h” – 此为 AHRS 表头档, FCA 必须引入此表头档以连结 AHRS 函式库.
  - “AHRSLib.lib” – AHRS 函式库本体, FCA 编译时必须连结 AHRS 函式库本体.
- **Function Interfaces**

NAME	**nvtAHRSInit**	
SYNOPSIS	**void nvtAHRSInit(void)**	
Description	Do the state initialize of AHRS library. This function should be called before using AHRS library.	
Parameters	**None**	
Return Value	void	

NAME	**nvtMillisecondTick**	
SYNOPSIS	**void nvtMillisecondTick(void)**	
Description	AHRS needs applications call this function every millisecond to provide a millisecond tick for AHRS operation.	
Parameters	**None**	
Return Value	void	

NAME	**nvtSetAccScale**
SYNOPSIS	**void nvtSetAccScale(float* AccScale)**
Description	Set ACC scale factor to AHRS and compensate ACC scale error.

Parameters	**Float* AccScale{3}**	ACC scale X,Y and Z
Return Value	void	

NAME	**nvtSetAccOffset**	
SYNOPSIS	**void nvtSetAccOffset(float* AccMean)**	
Description	Set ACC offset factor to AHRS and compensate ACC mean error.	
Parameters	**Float* AccMean{3}**	ACC mean X, Y, and Z
Return Value	void	

NAME	**nvtSetAccG_PER_LSB**	
SYNOPSIS	**void nvtSetAccG_PER_LSB(float G_PER_LSB)**	
Description	Tells the AHRS how raw data to be converted into a unified gravity (1G)	
Parameters	**Float G_PER_LSB**	A mapping factor from sensor raw to 1G, usually depend on sensor setting
Return Value	void	

NAME	**nvtSetGyroScale**	
SYNOPSIS	**void nvtSetGyroScale(float* GyroScale)**	
Description	Set GYRO scale factor to AHRS and compensate GYRO scale error.	
Parameters	**Float* GyroScale{3}**	GYRO scale X,Y and Z
Return Value	void	

NAME	**nvtSetGyroOffset**	
SYNOPSIS	**void nvtSetGyroOffset(float* GyroMean)**	
Description	Set GYRO offset factor to AHRS and compensate GYRO mean error.	
Parameters	**Float* GyroMean {3}**	GYRO mean X, Y, and Z
Return Value	void	

NAME	**nvtSetGYRODegPLSB**	
SYNOPSIS	**void nvtSetGYRODegPLSB(float DPLSB)**	
Description	Tells the AHRS how raw data to be converted into a unified degree (1 degree)	
Parameters	**Float DPLSB**	A mapping factor from sensor raw to one degree, usually depend on sensor settings
Return Value	void	

NAME	**nvtSetMagCalMatrix**	
SYNOPSIS	**void nvtSetMagCalMatrix(float* MagCalMatrix)**	
Description	Set MAG calibration matrix to AHRS and compensate MAG soft iron and hard iron.	
Parameters	**Float* MagCalMatrix {10}**	Array pointer to 10 float parameters for MAG gauss distortion fix.
Return Value	void	

NAME	**nvtSetMagGaussPLSB**

SYNOPSIS	**void nvtSetMagGaussPLSB(float)**	
Description	Tells the AHRS how raw data to be converted into a unified gauss (1 gauss)	
Parameters	**Float**	A mapping factor from sensor raw to one gauss, usually depend on sensor setting
Return Value	void	

NAME	**nvtInputSensorRawBARO**	
SYNOPSIS	**void nvtInputSensorRawBARO(int16_t *raw)**	
Description	Input BARO raw data into AHRS	
Parameters	**Int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtInputSensorRawBARO**	
SYNOPSIS	**void nvtInputSensorRawBARO(int16_t *raw)**	
Description	Input BARO raw data into AHRS	
Parameters	**Int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtInputSensorRawACC**
SYNOPSIS	**void nvtInputSensorRawACC(int16_t *raw)**

Description	Input ACC raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: ACC X Raw[1]: ACC Y Raw[1]: ACC Z
Return Value	void	

NAME	**nvtInputSensorRawGYRO**	
SYNOPSIS	**void nvtInputSensorRawGYRO(int16_t *raw)**	
Description	Input GYRO raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: GYRO X Raw[1]: GYRO Y Raw[1]: GYRO Z
Return Value	void	

NAME	**nvtInputSensorRawMAG**	
SYNOPSIS	**void nvtInputSensorRawMAG(int16_t *raw)**	
Description	Input MAG raw data into AHRS	
Parameters	**Int16* raw{3}**	Raw[0]: MAG X Raw[1]: MAG Y Raw[1]: MAG Z
Return Value	void	

NAME	**nvtGetSensorRawBARO**

SYNOPSIS	**void nvtGetSensorRawBARO(int16_t *raw)**	
Description	Get BARO raw data from AHRS	
Parameters	**int16* raw{2}**	Raw[0]:Raw temperature from sensor Raw[1]:Raw pressure from sensor
Return Value	void	

NAME	**nvtGetSensorRawACC**	
SYNOPSIS	**void nvtGetSensorRawACC(int16_t *raw)**	
Description	Get ACC raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: ACC X Raw[1]: ACC Y Raw[1]: ACC Z
Return Value	void	

NAME	**nvtGetSensorRawGYRO**	
SYNOPSIS	**void nvtGetSensorRawGYRO(int16_t *raw)**	
Description	Get GYRO raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: GYRO X Raw[1]: GYRO Y Raw[1]: GYRO Z
Return Value	void	

NAME	**nvtGetSensorRawMAG**

SYNOPSIS	**void nvtGetSensorRawMAG(int16_t *raw)**	
Description	Get MAG raw data from AHRS	
Parameters	**int16* raw{3}**	Raw[0]: MAG X Raw[1]: MAG Y Raw[1]: MAG Z
Return Value	void	

NAME	**nvtUpdateAHRS**	
SYNOPSIS	**void nvtUpdateAHRS(uint8_t UPDATE)**	
Description	Start AHRS calculation and come up with Euler angle and Quaternion. Application can select sensors by active bits from input parameter ‘UPDATE’. Sensors with bit ‘1’ are going to be calculated by AHRS.	
Parameters	**uint8_t UPDATE**	Bit0: update with ACC Bit1: update with GYRO Bit2: update with MAG Bit3: update with BARO
Return Value	void	

NAME	**nvtGetEulerRPY**	
SYNOPSIS	**void nvtGetEulerRPY(float*)**	
Description	Get the Euler angle previously calculated by AHRS	
Parameters	**float* {3}**	[0]: Roll angle in degree [1]: Pitch angle in degree

		[2]: Yaw angle in degree
Return Value	void	

NAME	**nvtGetVelocity**	
SYNOPSIS	**void nvtGetVelocity(float* Velocity)**	
Description	Get the estimated velocity previously calculated by AHRS. The parameter returns the velocity (meter/second) along sensor ACC axis X, Y and Z. This needs application to input sensor ACC raw and turn on bit "update with ACC".	
Parameters	**float* Velocity {3}**	Velocity [0]: X velocity. Velocity [1]: Y velocity. Velocity [2]: Z velocity.
Return Value	void	

NAME	**nvtCalACCInit**	
SYNOPSIS	**void nvtCalACCInit(void)**	
Description	Application should call this function before processing ACC calibration.	
Parameters	**None**	
Return Value	void	

NAME	**nvtCalACCBufferFill**
SYNOPSIS	**signed char nvtCalACCBufferFill(int8_t Dir)**
Description	Input ACC raw data into AHRS buffer. For full calibration, application

	needs to set input parameter ‘Dir’ from 0 ~ 5, to sample all 6 faces ACC raw data into buffer.	
Parameters	**int8_t Dir**	0: Sample side 0 (“Fast Z”) 1: Sample side 1 2: Sample side 2 3: Sample side 3 4: Sample side 4 5: Sample side 5
Return Value	STATUS_BUFFER_FILLED：Side x buffer sample done STATUS_BUFFER_NOT_FILLED：Side x buffer not ready STATUS_CAL_DONE: All 6 buffers are full and ACC calibration done.	

NAME	**nvtGetAccOffset**	
SYNOPSIS	**void nvtGetAccOffset(float*)**	
Description	Get the calibration factor of ACC offset	
Parameters	**float* {3}**	[0]: X offset. [1]: Y offset. [2]: Z offset.
Return Value	void	

NAME	**nvtGetAccScale**
SYNOPSIS	**void nvtGetAccScale (float*)**
Description	Get the calibration factor of ACC scale

Parameters	**float* {3}**	[0]: X scale. [1]: Y scale. [2]: Z scale.
Return Value	void	

NAME	**nvtCalGyroInit**	
SYNOPSIS	**void nvtCalGyroInit(char axis)**	
Description	Application should call this function before processing GYRO scale calibration. Parameter ‘axis’ is the axis to be calibrated.	
Parameters	**char axis**	0: X 1: Y 2: Z
Return Value	void	

NAME	**nvtGyroScaleCalibrate**	
SYNOPSIS	**signed char nvtGyroScaleCalibrate(int8_t axis)**	
Description	Do the GYRO scale calibration by axis X,Y and Z. Parameter ‘axis’ can be 0 ~ 2.	
Parameters	**int8_t axis**	0: X 1: Y 2: Z
Return Value	STATUS_GYRO_CAL_RUNNING: Scale calibration to axis running	

	STATUS_GYRO_AXIS_CAL_DONE : Scale calibration to axis done

NAME	**nvtGyroCenterCalibrate**	
SYNOPSIS	**signed char nvtGyroCenterCalibrate(void)**	
Description	Do the GYRO center calibration. Just put sensor steady. Wait and check return value "STATUS_GYRO_CAL_DONE".	
Parameters	**int8_t axis**	0: X 1: Y 2: Z
Return Value	STATUS_GYRO_CAL_BEGINE: Center calibration begin STATUS_GYRO_CAL_RUNNING: Center calibration running STATUS_GYRO_CAL_DONE: Center calibration done	

NAME	**nvtGetCalibratedGYRO**	
SYNOPSIS	**void nvtGetCalibratedGYRO(float*)**	
Description	Get calibrated gyro from AHRS.	
Parameters	**float* {3}**	[0]: GYRO X (degree/second) [1]: GYRO Y (degree/second) [2]: GYRO Z (degree/second)
Return Value	void	

NAME	**nvtCalMAGBufferFill**
SYNOPSIS	**signed char nvtCalMAGBufferFill(void)**

Description	Direct the sampled MAG raw data into MAG calibration buffer in AHRS. When there are 120 samples in MAG calibration buffer, the sample is done and AHRS starts to calculate the calibration matrix for MAG.
Parameters	**None**
Return Value	STATUS_BUFFER_NOT_FILLED: Calibration buffer not filled yet. STATUS_CAL_DONE: Calibration buffer filled and calibration done.

NAME	**nvtGetMagCalQFactor**
SYNOPSIS	**uint8_t nvtGetMagCalQFactor(void)**
Description	When MAG calibration is done. Call this function to get a calibration quality factor (0 ~ 255). A factor less than 5 is good and less than 10 is fine. You may consider to redo calibration when factor greater than 10.
Parameters	**None**
Return Value	0 ~ 255 0~5: Good 6~10: Fine 11~255: Bad

NAME	**nvtGetAccZWithoutGravity**
SYNOPSIS	**void nvtGetAccZWithoutGravity(float *ZWithoutGravity, float *AccZMag)**
Description	Call this function to get the output Z without gravity effect and the magnitude of Z 、.

Parameters	**float *ZWithoutGravity,** **float *AccZMag**	ZWithoutGravity: (ACC_Z – Gravity_Z) AccZMag: (ACC_X* ACC_X + ACC_Y*ACC_Y+ACC_Z*ACC_Z)
Return Value	void	

NAME	**nvtGetMove**	
SYNOPSIS	**void nvtGetMove(float* Move)**	
Description	Call this function to get the move distance (cm) from AHRS ACC calculation.	
Parameters	**float * Move{3}**	Move[0]: X distance Move[1]: Y distance Move[2]: Z distance
Return Value	void	