AHRS IMU Sensor | WT31N

The Robust Acceleration & Angle Detector

The WT31N is a IMU sensor device, detecting acceleration , angle. The small outline makes it perfectly suitable for industrial applications such as condition monitoring and predictive maintenance. Configuring the device enables the customer to address a broad variety of application by interpreting the sensor data by smart algorithms and Kalman filtering.

BUILT-IN SENSORS

Accelerometer

Tutorial Link

Google Drive

Link to instructions DEMO:
WITMOTION Youtube Channel
WT31N Playlist

If you have technical problems or cannot find the information that you need in the provided documents, please contact our support team. Our engineering team is committed to providing the required support necessary to ensure that you are successful with the operation of our AHRS sensors.

Contact

Technical Support Contact Info

Application

- AGV Truck
- Platform Stability
- Auto Safety System
- 3D Virtual Reality
- Industrial Control
- Robot
- Car Navigation
- UAV
- Truck-mounted Satellite Antenna Equipment

Contents

1 Overview

WT31N's scientific name is AHRS IMU sensor. A sensor measures angle, acceleration. Its strength lies in the algorithm which can calculate 3-axis angle accurately.

WT31N is employed where the highest measurement accuracy is required. WT31N offers several advantages over competing sensor:

• Heated for best data availability: new WITMOTION patented zero-bias automatic detection calibration algorithm outperforms traditional accelerometer sensor

• High precision (X Y Z axis) Acceleration + (Roll Pitch) Angle output

• Low cost of ownership: remote diagnostics and lifetime technical support by WITMOTION service team

•Developed tutorial: providing manual, datasheet, Demo video, free software for Windows computer, APP for Android smartphones , and sample code for MCU integration including 51 serial, STM32, Arduino, Matlab, Raspberry Pi,communication protocol for project

• WITMOTION sensors have been praised by thousands of engineers as a recommended attitude measurement solution

2 Features

- The default baud rate of this device is 9600 and could be changed as 115200.

- The interface of this product only leads to a serial port

- The module consists of a high precision accelerometer sensor. The product can solve the current real-time motion posture of the module quickly by using the high-performance microprocessor, advanced dynamic solutions and Kalman filter algorithm.

- The advanced digital filtering technology of this product can effectively reduce the measurement noise and improve the measurement accuracy.

- Maximum 10Hz data output rate. Output content can be arbitrarily selected, the output speed 0.2Hz or 10HZ adjustable.

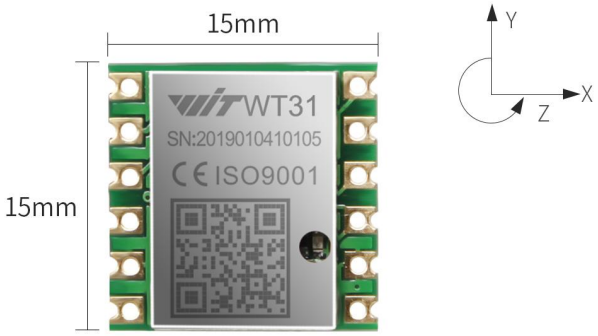
3 Specification

3.1 Parameter

Parameter	Specification
➢ Working Voltage	3.3V-5V
➢ Current	<10mA
➢ Size	15mm x 15mm X 2mm
➢ Data	Angle: X Y, 2-axis Acceleration: X Y Z, 3-axis Time
➢ Output frequency	0.2-10Hz
➢ Interface	Serial TTL level
➢ Baud rate	115200,9600(default)

Measurement Range & Accuracy		
Sensor	Measurement Range	Accuracy/ Remark
➢ Accelerometer	X, Y, Z, 3-axis ±2g	Accuracy: 0.01g Resolution: 16bit Stability: 0.005g
➢ Angle/ Inclinometer	X, Y, 2-axis X-axis: ±180° Y ±90° (Y-axis 90° is singular point)	Accuracy:X, Y-axis: 0.05°

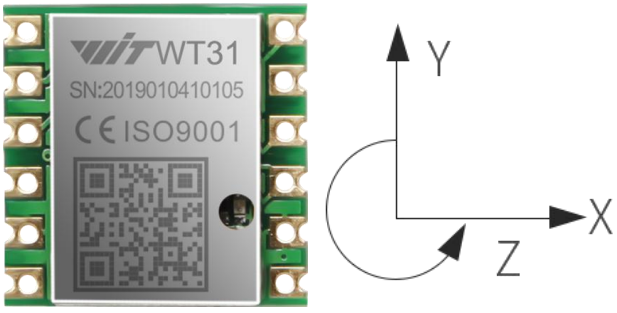
3.2 Size



Parameter	Specification	Tolerance	Comment
Length	15	±0.1	Unit: millimeter.
Width	15	±0.1	
Height	2	±0.1	
Weight	1	±0.1	Unit: gram

3.3 Axial Direction

The coordinate system used for attitude angle settlement is the northeast sky coordinate system. Place the module in the positive direction, as shown in the figure below, direction right is the X-axis, the direction forward is the Y-axis, and direction upward is the Z-axis. Euler angle represents the rotation order of the coordinate system when the attitude is defined as Z-Y-X, that is, first turn around the Z-axis, then turn around the Y-axis, and then turn around the X-axis.



4 Pin Definition

PIN	Function
➢ VCC	3.3-5V input supply
➢ RX	Serial data input, TTL interface
➢ TX	Serial data output, TTL interface
➢ GND	Ground
➢ D2	X + alarm
➢ D3	X - alarm
➢ Y+	Y + alarm
➢ Y-	Y - alarm

5 Communication Protocol

Level : TTL level

Band rate : 9600(default), 115200, Stop bit 1, check bit 0.

Every frame data that the module sent to the PC software is divided into 3 packets, acceleration pocket, angular velocity packet and angle packet.

5.1 Output Data Format

5.1.1 Acceleration Output

Data Number	Data Content	Implication
0	0x55	Header of packet
1	0x51	Acceleration pack
2	AxL	X axis acceleration low type
3	AxH	X axis acceleration high type
4	AyL	Y axis acceleration low type
5	AyH	Y axis acceleration high type
6	AzL	Z axis acceleration low type
7	AzH	Z axis acceleration high type
8	TL	Temperature low type
9	TH	Temperature high type
10	Sum	Checksum

Formula for calculating acceleration：

ax=((AxH<<8)|AxL)/32768*16g(g is gravity acceleration, 9.8m/s2)
ay=((AyH<<8)|AyL)/32768*16g(g is gravity acceleration, 9.8m/s2)
az=((AzH<<8)|AzL)/32768*16g(g is gravity acceleration, 9.8m/s2)

Checksum：
Sum=0x55+0x51+AxH+AxL+AyH+AyL+AzH+AzL+TH+TL

Description:
1. The data is sent in hexadecimal not ASCII code.
2. Each data is transmitted in order of low byte and high byte, and the two are combined into a signed short type data. For example, the X-axis acceleration data Ax, where AxL is the low byte and AxH is the high byte. The conversion method is as follows:

Assuming that Data is actual data, DataH is the high byte part, and DataL is the low byte part, then: Data = ((short) DataH << 8) | DataL. It must be noted here that DataH needs to be converted to a signed short data first and then shifted, and the data type of Data is also a signed short type, so that it can represent negative numbers.

5.1.3 Angle Output

Data Number	Data Content	Implication
0	0x55	Header of packet
1	0x53	Angle packet
2	RollL	X axis angle low type
3	RollH	X axis angle high type
4	PitchL	Y axis angle low type
5	PitchH	Y axis angle high type
6	0x00	NO mean
7	0x00	NO mean
8	TL	Temperature low type
9	TH	Temperature high type
10	Sum	Checksum

Formula for calculating angle:

Roll angle(x axis) Roll=((RollH<<8)|RollL)/32768*180(°)
Pitching angle(y axis) Pitch=((PitchH<<8)|PitchL)/32768*180(°)

Formula for calculating temperature：
T=((TH<<8)|TL) /340+36.53℃

Checksum：
Sum=0x55+0x53+RollH+RollL+PitchH+PitchL+YawH+YawL+TH+TL

Attention：

The coordinate system used for attitude angle settlement is the northeast sky coordinate system. Place the module in the positive direction, as shown in the figure on Chapter 3.3, direction right is the X-axis, the direction forward is the Y-axis, and direction upward is the Z-axis. Euler angle represents the rotation order of the coordinate system when the attitude is defined as Z-Y-X, that is, first turn around the Z-axis, then turn around the Y-axis, and then turn around the X-axis.

5.2 Config Commands

5.2.1 Acceleration Calibration

Instruction content	Function	Remark
0xFF 0xAA 0x01 0x01 0x00	Accelerometer Calibration	X,Y axis to 0
0xFF 0xAA 0x04 0x06 0x00	Baud rate 115200, return rate 100HZ	Baud rate 115200
0xFF 0xAA 0x04 0x02 0x00	Baud rate 9600, return rate 20HZ	Baud rate 9600

Introductions :

1. When the module is on, it needs to be static first. Because the MCU inside the module will be automatically calibrated when it is static. After calibration, the angle of the Z-axis will be initialized to 0, Which can be considered as a signal that has been calibrated.

2. The factory default setting uses a serial port with a baud rate of 115200 and a frame rate of 100Hz. Configuration can be configured through the host computer software, because all configurations are saved after power-off, so only need to configure once.

5.2.2 Set alarm of module

1. Setting alarm value of X- angle

0xFF	0xAA	0x5A	XMINL	XMINH

For example: Set Alarm Angle of X- (Angel) to -10°, 0xFF 0xAA 0x5A 0xE4 0xF8
Angle conversion formula: Angel=((XMINH<<8)|XMINL)/32768*180(°)

2.Setting alarm value of X+ angle

0xFF	0xAA	0x5B	XMAXL	XMINH

For example: Set Alarm Angle of X+ (Angel) to +10° , 0xFF 0xAA 0x5B 0x1C 0x07
Angle conversion formula: Angel=((XMAXH<<8)|XMAXL)/32768*180(°)

3.Setting alarm value of Y- angle

0xFF	0xAA	0x5E	YMINL	XMINNH

For example: Set Alarm Angle of Y- (Angel) to +10° ,0xFF 0xAA 0x5A 0xE4 0xF8
Angle conversion formula: Angel=((YMINH<<8)|YMINL)/32768*180(°)

4.Setting alarm value of Y+ angle

0xFF	0xAA	0x5F	YMAXL	XMINH

For example: Set Alarm Angle of Y+ (Angel) to +10° , 0xFF 0xAA 0x5A 0x1C 0x07
Angle conversion formula: Angel=((YMAXH<<8)|YMAXL)/32768*180(°)

5.Setting of hold time setting

0xFF	0xAA	0x59	DELAYTL	DELAYTH

For Example: Set the save time to 100 ms Unit: ms
0xFF 0xAA 0x59 0x64 0x00

6. Setting of alarm level

Normally open (1): The port output is high level when the alarm value is included, and the port output is low level when the alarm value is exceeded.

0xFF	0xAA	0x62	0xFF	0xFF

Normally close (0): The port output is low level when the alarm value is included, and the port output is high level when the alarm value is exceeded.

0xFF	0xAA	0x62	0x00	0x00

5.3 Date Analysis and Sample Code

```
double a[3],w[3],Angle[3],T;
void DecodeIMUData(unsigned char chrTemp[])
{
    switch(chrTemp[1])
    {
    case 0x51:
        a[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*16;
        a[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*16;
        a[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*16;
        T = (short(chrTemp[9]<<8|chrTemp[8]))/340.0+36.25;
        break;
    case 0x52:
        w[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*2000;
        w[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*2000;
        w[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*2000;
        T = (short(chrTemp[9]<<8|chrTemp[8]))/340.0+36.25;
        break;
    case 0x53:
        Angle[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*180;
```

```
            Angle[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*180;
            Angle[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*180;
            T = (short(chrTemp[9]<<8|chrTemp[8]))/340.0+36.25;
            printf("a = %4.3f\t%4.3f\t%4.3f\t\r\n",a[0],a[1],a[2]);
            printf("w = %4.3f\t%4.3f\t%4.3f\t\r\n",w[0],w[1],w[2]);
printf("Angle
= %4.2f\t%4.2f\t%4.2f\tT=%4.2f\r\n",Angle[0],Angle[1],Angle[2],T);
            break;
        }
}
```

5.4 Data Analysis Sample Under Embedded Environment

The code is divided into two parts, one of them is interrupt reception. Find the header of the data and then put the data-packet into an array. The other one is data analysis, which is put into the main program section.

Interrupt unit(The following is AVR microcontroller code, different MCU reads the register a little different):

```
unsigned char Re_buf[11],counter=0;

unsigned char sign;

interrupt [USART_RXC] void usart_rx_isr(void) //USART serial receiving

interrupt

{

Re_buf[counter]=UDR; //Different micro-controller is slightly

Different

if(counter==0&&Re_buf[0]!=0x55) return; //NO.0 data is not a frame

header ,skip.

counter++;

if(counter==11) //11 data has been received

{ counter=0; // Reassigned,ready for the next frame data reception

sign=1;

}
```

```
}
Main program section：
float a[3],w[3],angle[3],T;
extern unsigned char Re_buf[11],counter;
extern unsigned char sign;
while(1)
{
if(sign)
{
sign=0;
if(Re_buf[0]==0x55) //Check the frame header
{
switch(Re_buf [1])
{c
ase 0x51:
a[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*16;
a[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*16;
a[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*16;
T = (short(Re_buf [9]<<8| Re_buf [8]))/340.0+36.25;
break;
```

```
case 0x52:
w[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*2000;
w[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*2000;
w[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*2000;
T = (short(Re_buf [9]<<8| Re_buf [8]))/340.0+36.25;
break;
case 0x53:
angle[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*180;
angle[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*180;
angle[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*180;
T = (short(Re_buf [9]<<8| Re_buf [8]))/340.0+36.25;
break;
}
}
      }
```