AHRS-G micro, Attitude and Heading Reference System

Features

- 3-axis inertial measurement unit (Roll, pitch, yaw)
- Stand-alone, completely independent from GPS or Air-data computer
- Computes G-load, rate of turn and slip-angle
- Orientation independent electronic compass
- Low power consumption (60 mA @ 5V)
- Small, compact plug-in module
- 3.3-5V serial hardware interface
- Accepts 5V regulated
- Configurable for pitch installation error via software
- Optional on-board pressure and speed sensors available (on AW model)
- Accepts speed information from external sources
- No calibration required
- Great performance at multiple temperatures
- High vibration rejection over wide frequencies
- Complete operation using mini USB port

Applications

- Aviation
- Robotics, Industrial sensors
- UAVs

Description

The AHRS-G micro is a standalone, embedded attitude and heading reference system. With its small form factor and low power consumption, the AHRS-G micro is perfect for portable or panel-mounted devices such as navigation displays that integrate flight instrumentation (i.e attitude indicators, multifunction displays). It is also ideal as back-up instrument when embedded on portable hardware such as GPS and ADS-B receivers for mobile devices (i.e Tablets, Electronic Flight Bags). Every module undergoes a strict and precise calibration process before leaving the manufacturer for reliable operation at different temperature levels and vibration frequencies. In the simplest configuration, the hardware requires only three connections (PWR, TX, GND). Additionally, the module can be easily calibrated for pitch installation errors by sending a simple command via software. The module can also be powered, and data accessed through the micro-USB port for easy troubleshooting.

Air-data computer (AW model)

The AHRS-G micro AW model has integrated pressure transducers to calculate barometric altitude and indicated airspeed when connected to the static and dynamic lines of the aircraft.

OVERVIEW

- ASCII output (default) or binary
- Data IN/OUT at 115,200 Bauds on RX1/TX1 (Attitude and Heading data)
- 3-axis MEMs gyros, 3-axis accelerometer, 3-axis magnetometer
- Dimensions: 24mm x 41 x 12mm
- Weight: 68 grams (2.4 oz)

Table 1. Technical Specs

Technical Specifications	
Temperature Range (Operating)	-40 °C to +85 °C
Supply Voltage VDD	4.5 to 5.5V
Serial Port 1	3.3V to 5V (115,200 bauds)
Hose size required (AW model)	ID

Table 2. Output characteristics

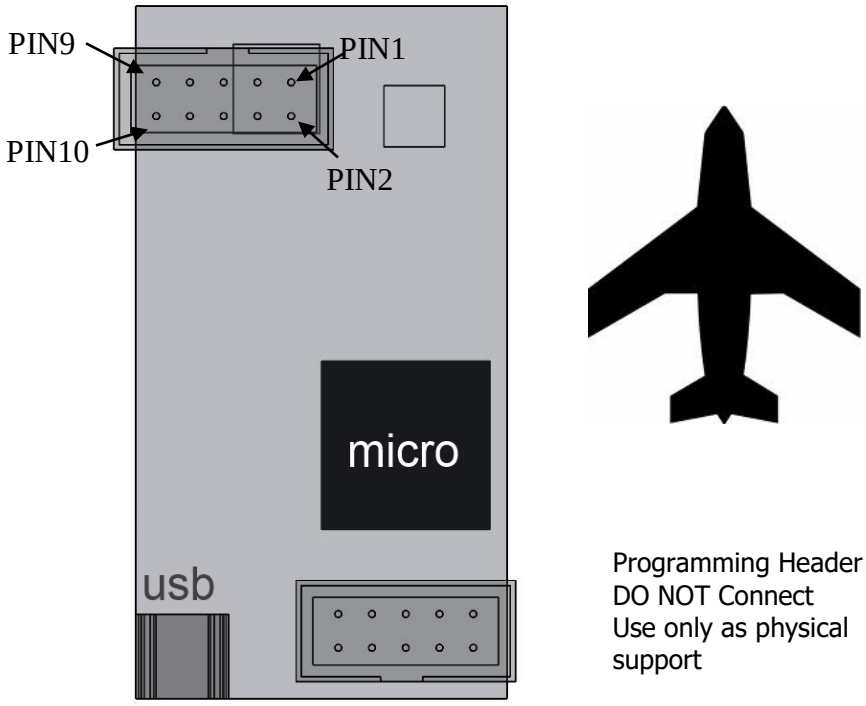
	Output Format	Resolution	Refresh rate	Range
Roll	Degrees	0.1 degrees	6 Hz (default) 12 Hz optional	-180 to 180 deg
Pitch	Degrees	0.1 degrees	6 Hz (default) 12 Hz optional	-90 to 90 deg
Magnetic heading	Degrees	0.1 degrees	6 Hz (default) 12 Hz optional	0 to 360 deg
Inclination (slip angle)	Degrees	0.1 degrees	6 Hz (default) 12 Hz optional	-90 to 90 deg
Rate of Turn (turn coordinator)	Degrees	0.1 deg per second	6 Hz (default) 12 Hz optional	-500 to 500 deg/sec
G forces	Magnitude in all directions (default) or Vectored for Z axis only (optional)	0.0001 Gs	6 Hz (default) 12 Hz optional	Up to 4 Gs
Altitude	Feet at 29.92 inHg	1 ft	6 Hz (default) 12 Hz optional	-1000 to 63,000ft

Airspeed	Knots	1 kt	6 Hz (default) 12 Hz optional	0kts to 210 kts (standard) Supports higher speeds with optional pressure transducer

Schematics

Figure 1. Pin Description

Bottom view (micro-processor is not visible)



Pin	Name	Description	Default
1	Serial TX1	AHRS data transmission (3.3V)	115,200 bauds
2	Serial RX1	INPUT (CRC based commands only) 3.3 to 5V	115,200 bauds
3	N/A	Not connected	Not connected
4	N/A	Not connected	Not connected
5	N/A	Not connected	Not connected.
6	N/A	Not connected	Not connected

7	5V	5V IN ←	-
8	5V	5V IN	-
9	GND	GND	-
10	GND	GND	-

Table 2. Pin description

Figure 2. Physical dimensions (mm)

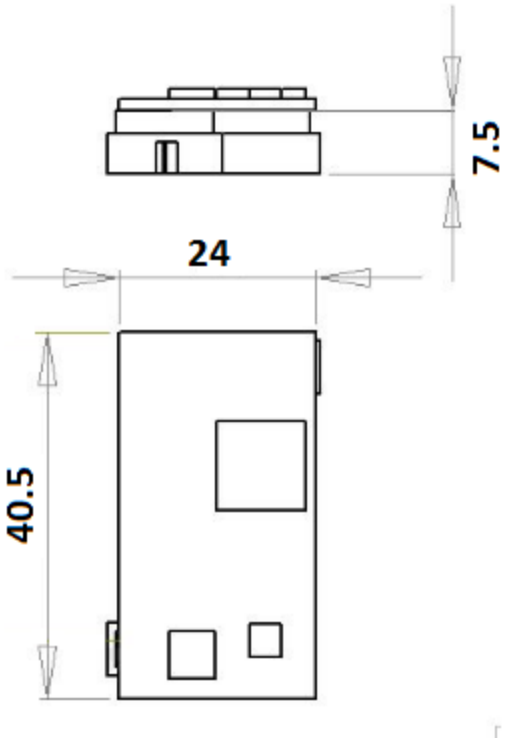
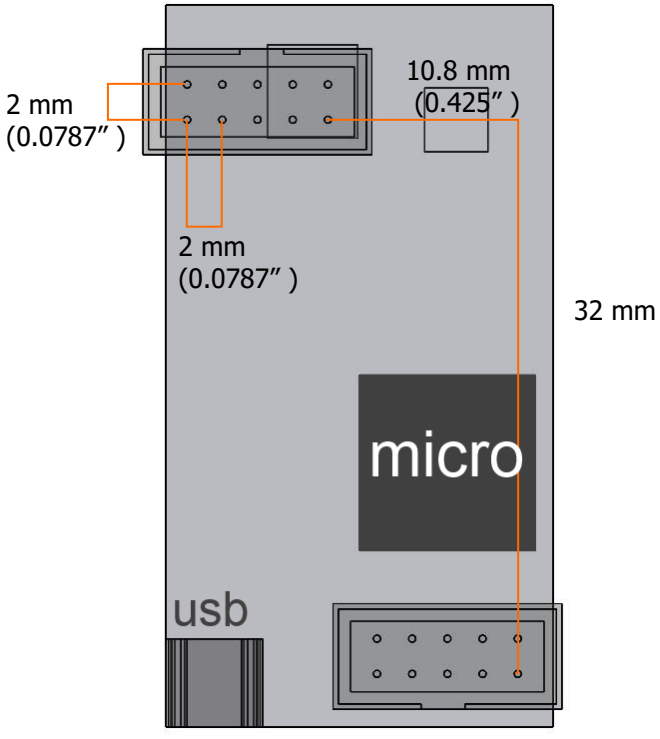


Figure3. Connector headers' orientation
Bottom View (micro not visible from bottom)



Design Concerns

The following sections provide information on designing with AHRS-G micro module, including module orientation, minimizing compass deviations etc.

Module orientation

When integrating the AHRS-G micro, developers have to take into account AHRS orientation in relation to the external plane field. **Figure 4** and **Table 3** below represent the axis convention used by the module. When using in aviation related applications, developers must keep in mind that the AHRS roll axis needs to be aligned with the aircraft's roll axis.

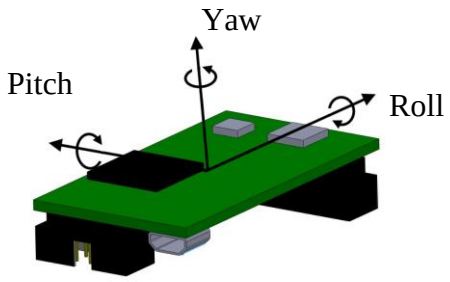
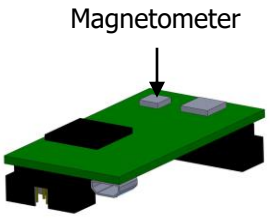


Figure 4. AHRS Calibration orientation

Minimizing Magnetic Interference

The 3-axis magnetometer located on the top left corner of the module is very susceptible to magnetic deviations and may affect the magnetic heading indication. In order to minimize magnetic deviations due to external components, developers should be aware of the location of the magnetometer and its surroundings on the development board. Position the magnetometer as far away from ferrous metals, high-power wires, etc as possible.



Minimizing Vibrations

When integrating the AHRS-G micro, it is recommended to use both connector headers even though one of them does not require any physical connections. Having both connector headers will minimize vibration and ensure the AHRS will stay in place. Take vibrations into account when designing the PCB and housing that will integrate the AHRS. Make sure the end-user will be able to securely locate it on a flat/stable surface and that it will remain properly aligned.

Table 3. AHRS output convention

Axis	Output
Roll	- Roll (positive clockwise)
Pitch	- Pitch (positive up)
Yaw	- Magnetic Heading (tied to the magnetometer) - Rate of turn (positive to the right)
Slip Angle (positive to the right)	
G forces (Vectored for Z axis only)	
Altitude	In feet at 29.92 inHg. Only available on AW model when connected to static source.
Airspeed	In Kts. Only available on AW model when connected to pitot-static source.

Adjusting Tilt Offset, and supplying Altitude and Speed information to AHRS G-micro

Unlike the AHRS-G micro AW, the SW version does not have integrated pressure transducers for Altitude and Speed calculations. Although not required, speed information is particularly helpful when estimating attitude indication during unusual maneuvers and aerobatics, thus the AHRS-G micro SW is capable of receiving this information from external hardware by serial TTYL communication using a simple command.

Command Specification

The iLevil can be configured using a 12-byte command. After the header, the command ID determines the action to be taken by the iLevil. Some commands require additional parameters (bytes 2 through 7). Use a zero to fill-in the unused bytes in the parameters section.

Header byte (0x24)	Command ID (Byte 1)	Parameters (Byte 2,3,4,5,6,7) (LSB)	CRC (Byte 8,9,10,11) (MSB)

Tilt Offset (Command ID 0x05)

The AHRS-G micro can be configured with a tilt offset (positive or negative) using the following command:

Byte #	Name	Size	Value
0	Header byte	1	0x24 ('$') start of message
1	Command ID	1	0x05 (save to memory)
2-3	Memory Address (LSB first)	2	Tilt offset address: 2 Byte2: 0x02 Byte3: 0x00
4-5	Tilt Offset (LSB first)	S2	Example: 5 degrees' offset Byte4: 0x05 Byte5: 0x00 Example -10 degrees' offset Byte4: 0xF6 Byte5: 0xFF
6-7	reserved	2	0x0000
8-11	CRC	4	Most significant byte first

CRC-calculation

The following function (calculate_crc32) must be called twice in order to obtain the final crc value. The first call of the function calculates crc against the array containing the first 8 bytes of the command, and the second call against an array of zeros.

```
//Global variables
#define CRC32_POLYNOMIAL      0x4C11DB7  // Polinomial used for CRC32 calculation
/*****************************************************************************
* Function Name  : calculate_crc32 ( initial value, address to initial byte, number of bytes)
* Description    : calculate the CRC32 for the EEPROM
* Input          : sum: previous checksum value as initial value
*                  p  : address to start point of checksum
```

```
*                len: number of bytes to check
* Output         : new checksum value
*******************************************************************************/
u32 calculate_crc32 ( u32 sum, u16 p, u16 len )
{
  while ( len )
   {
      int i = 0 ;
      unsigned char byte = 0 ;
      p++;

      len -- ;
      for (i = 0; i < 8; ++i)    // 8 bits per byte
      {
         unsigned long osum = sum ;
         sum <<= 1 ;

         if ( byte & 0x80 )   // check for 1 in the MSB of "byte"
         {
            sum |= 1 ;
         }

         if ( osum & 0x80000000 ) // check for 1 in the MSB of the previous CRC32
         {
            sum ^= CRC32_POLYNOMIAL ;
         }
         byte <<= 1 ;
      }
   }
   return sum ;
}
```

Altitude, Speed and Acceleration command ID

The above command protocol is used to send Altitude and Speed information. Because the AHRS does not know the input rate of the speed and altitude information from different external devices, a third component is necessary, acceleration. The external device must calculate the acceleration in G's based on the last speed sent to the AHRS and the current speed to be sent.
If any of the values are invalid, send 0xFFFF for that value. This command does not have to be sent at a particular rate, as long as the acceleration is calculated correctly. If no acceleration is able to be computed, use 0, or 0xFF and the AHRS-G micro will not use the speed for compensation, only for display.

Byte #	Name	Size	Value
0	Header byte	1	0x24 ('$') start of message

1	Command ID	1	0x0D (carriage return)
2-3	Airspeed	2	Speed in meters per second (multiplied by 100) Resolution 0.01 mts/sec **Least significant byte first** Use 0xFFFF if value is invalid
4-5	Altitude	2	Altitude in feet + 5000 Resolution 1 ft **Least significant byte first** Use 0xFFFF if value is invalid
6-7	Acceleration	2	Acceleration in G (multiplied by 100) Resolution 0.01 g **Least significant byte first** Use 0xFFFF if value is invalid
8-11	CRC	4	Most significant byte first

Sample code on how to construct Altitude,Speed, Acc command

```
//Send Altitude=10,500 ft, Speed = 75 kts, and Acceleration = 0.13g
u16 theSpeed = (75*0.514444)*(100); //converting to mts/sec *100
u16 theAltitude = 10500 + 5000; //altitude +5000
u16 theAcceleration = 0.13*100;  //multiply by 100
Out_to_AHRS(0x0D,(theSpeed&0xFF),(theSpeed>>8),(theAltitude&0xFF),(theAltitude>>8),
(theAcceleration&0xFF),(theAcceleration>>8));

/*****************************************************************************
* Function Name  : Out_to_AHRS
* Description    : Sends commands to AHRS micro
*****************************************************************************/
void Out_to_AHRS(u8 CMD, u8 a1,u8 a2,u8 a3,u8 a4,u8 a5,u8 a6)
{
  const u32 zero = 0 ;
  u32 Test_checksum;
  char   TxData[20];
  char  *T;


 u8 i;


   T = &TxData[0];
        TxData[0] = '$'; // Header byte
        TxData[1] = CMD; // command ID
        TxData[2] = a1; // less significant byte
        TxData[3] = a2; // most significant byte
        TxData[4] = a3; // less significant byte
        TxData[5] = a4; // most significant byte
        TxData[6] = a5; // less significant byte
```

```
        TxData[7] = a6; // most significant byte

Test_checksum = calculate_crc32 ( 0, (u8*)T , 8 ); // first CRC call
Test_checksum = calculate_crc32 ( Test_checksum, (unsigned char *)&zero, 4); //second CRC call

        TxData[11] = (Test_checksum & 0xFF000000) / 0x1000000;
        TxData[10] = (Test_checksum & 0xFF0000) / 0x10000;
        TxData[9] = (Test_checksum & 0xFF00) / 0x100;
        TxData[8] = (Test_checksum & 0xFF);

        //send data out
        for(i=0; i<12; i++)
        {
          USART_SendData(UART5, TxData[i]);
          while(USART_GetFlagStatus(UART5, USART_FLAG_TXE) == RESET);
        }

}
```

RESULTING COMMAND SENT TO AHRS (speed 75kts, altitude 10,500ft, acc 0.13g)
0x24 0x0D 0x12 0x0F 0x8C 0x3C 0x0D 0x00 0xEE 0xAA 0x6B 0x1B

Sample code on how to calculate Acceleration (1 Hz)

```
//Global variable
u8 GPS_fix; //or validity of Pressure data if using instead of GPS (0-invalid)
float New_speed = 0;
float Last_speed = 0;
float GPSGroundSpeed = 0; //or Indicated airspeed if using instead of GPS in Kts
float GPSAcceleration = 0; // or acceleration based on indicated airspeed over time
u8 Acc_counter = 1; //time between last known speed and new speed calculation

/*******************************************************************************
* Function Name  : TimerInterruption_1Hz
* Description    : Calculates Acceleration based on speed change over 1 second
*******************************************************************************/
void TimerInterruption_1Hz(void)
{
u16 theSpeed =  0xFFFF;
u16 theAltitude = 0xFFFF;




u16 theAcceleration = 0xFFFF;

GPSAcceleration = 0;
```

```
if (GPSFix>0) //or Pressure data valid
  {
    New_speed = GPSGroundSpeed*0.514444; // in m/s

    if (GPSGroundSpeed>5) //to prevent noise
       {
        if (Acc_counter<6)//allows GPS loosing fix for  max 6 sec
                    {
            GPSAcceleration = (New_speed - Last_speed)/(Acc_counter*9.8);
           }
        if (GPSAcceleration>0.5)
                    {
                     GPSAcceleration = 0.5;
                    }
        if (GPSAcceleration<-0.5)
                    {
                     GPSAcceleration = -0.5;
                    }
        Acc_counter = 1;

       }
     theSpeed = (u16)(New_speed*100.0);
     theAcceleration = (s16)(GPSAcceleration*100.0);
     theAltitude = GPSAltitude + 5000;
     Last_speed = New_speed;
   }
   else // lost GPSfix or no pressure data valid during this interruption
   {
      if (Acc_counter<200) Acc_counter++;
   }

Out_to_AHRS(0x0D,(theSpeed&0xFF),(theSpeed>>8),(theAltitude&0xFF),(theAltitude>>8),
(theAcceleration&0xFF),(theAcceleration>>8));
}
```

Sample code on the receiving end of the AHRS-micro

```
//At this point the message has been copied to RxData[] and crc has been verified...

if (RxData[1] == 0x0D)   //Read airspeed from external source
{
   if ((Pressure_installed==0) || (Pressure_installed==4)) //
     {
       if ((RxData[2] !=0xFF) || (RxData[3] !=0xFF))
             {
           True_vel = (float)((s16)(RxData[3]*0x100 + RxData[2])/100.);
             }
           }
   if (Pressure_installed==0) //use ST altimeter if Pressure_installed = 4
     {
          if ((RxData[4] !=0xFF) || (RxData[5] !=0xFF))
             {
```

```
                Altitude  = (RxData[5]*0x100 + RxData[4]) - 5000;
        }
         }
  if ((Pressure_installed==0) || (Pressure_installed==4))
   {
          if ((RxData[6] !=0xFF) || (RxData[7] !=0xFF))
            {
                Acc_vel = 0.8*Acc_vel + 0.2*(float)((s16)(RxData[7]*0x100 +
                      RxData[6])/100.); //comes in G's div 100
          Press_inp_flag = 1;
        }
     }

 }
```

Notes

Ordering Information

Levil Aviation
1704 Kennedy Point, Ste 1124
Oviedo, FL USA 32765

Phone: 407-542-3971
info@levil.com