Overview

Usage of a Vibrating Gyroscope for Orientation Estimation

Dipl.-Inform. Gerhard Weiß
University of Kaiserslautern
Computer Science Department
Research Group Prof. Dr. E. v. Puttkamer

P.O.Box 3049 • D-67653 Kaiserslautern

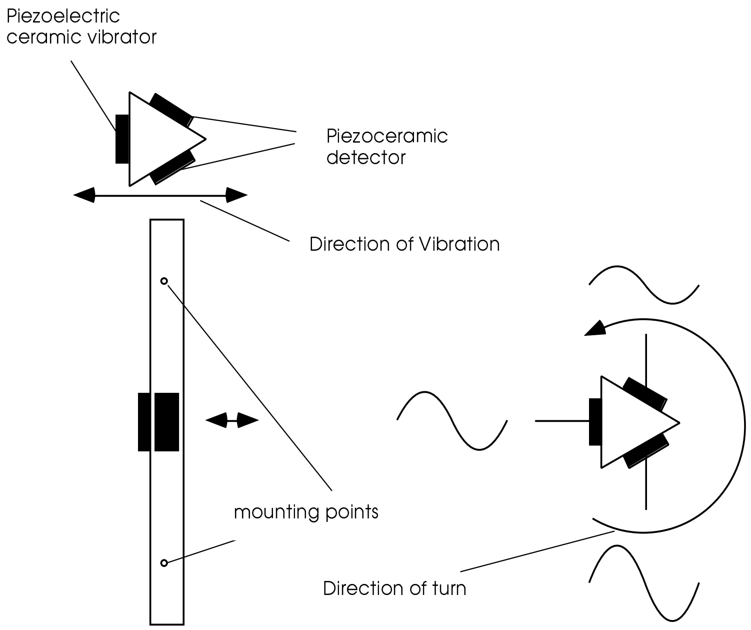
Phone +49-631-205-2656 • Fax +49-631-205-2803
e-Mail: weiss@informatik.uni-kl.de

The muRata Gyrostar

We tested the GYROSTAR ENV-05S. This device is a sensor for angular velocity. Therefore the orientation must be calculated by integration of the angular velocity over time. The devices output is a voltage proportional to the angular velocity and relative to a reference. The test where done to find out under which conditions it is possible to use this device for estimation of orientation.

Principle of the Sensor

Inside the device is a triangle metal prism, which is fixed at two points. The triangle prism is forced to vibrate by a piezoelectric ceramic at about 7 kHz. With no rotation around the high axis, the two other piezoelectric ceramics detect a equal large signal. When the prism is turned it gets twisted, so the detectors receive different signals. This signal difference is examined by the internal analogue circuits and brought out as a voltage proportional to the angular velocity.



Specs. of the muRata Gyrostar

Supply voltage	8-13.5 V DC
Supply current	13 mA
Max. angular velocity	+/- 90 °/sec
Scale factor	22 mV/°/sec.
Output	2.5 V DC @ 0 °/sec.
	+/-2.0 V DC at max. angular velocity
Linearity	Within 0.1% of max. angular velocity
Hysteresis	None
Temperature offset	Within 0.1% of max. angular velocity/°C
Drift	Within 0.1% of max. angular velocity/h
Work. Temperature	-20 - +60°C
Storage Temperature	-55 - +85°C
Output noise	Within 10 mV RMS
Dimensions	24 x 24 x 58 mm
Weight	41g

Investigations

1. Noise

The observed noise comes obviously from the vibration itself. As it has a frequency of 7 kHz, it can easily by suppressed.

2. Drift

Here seems to be ment the error in offset drift over time. The main source of the drift error seem to be self-heating.

3. Linearity

The device is highly linear. We observed only deviations of less than 0.05 °/s from the ideal line (at 21° - room temperature).

4. Scale factor and offset

Some sample measurements:

°C	scale factor
16	21.997 mV/°/s
18	22,007 mV/°/s
19,5	22,000 mV/°/s
22	22,069 mV/°/s
23	22.078 mV/°/s
45	21.991 mV/°/s

Note that the values are very close to 22 mV°/s (±0,15%).

Offset at 0°/s

°C	Offset
16	2.428 V
18	2.452 V
19.5	2.477 V
22	2.501 V
23	2.525 V
45	2.549 V

=> approx. 0.5°/s /°C

This is the real problem: due to integration over time, a offset error leads to continuous drift.

Compensation of scale factor error

We observe the following:
A error in scale factor results in a proportional error in angle:

$$\alpha = \int (p + \Delta p)\bullet(\text{Measurement-Offset})dt = (p + \Delta p)\bullet\int(\text{Measurement-Offset})dt$$

The above described error of 0.15% in scale factor leads to a error of 0.15% in angle, so after 1000° (about 3 turns) the error should be less than 2°. In order to keep the angle low, there should be a many left as right turns to the vehicle.

Compensation of offset error

Errors in the offset come from drifts in temperature over time. Therefore the device should be kept at the same temperature. On land vehicles, we can assume, that from time to time the vehicle stands still. This is used to get the actual offset.

Sample code (in Pascal)

```
unit Gyroscope;
interface
      uses  GeneralTypesDecl;

      procedure GyroInit (time: integer);
      {initialising}
      {in par. time, the time between two calls of GyroUpdate is given}

      procedure GyroSetAngle (ExtOffset: radians);
      {To set a predefined offset}

      procedure GyroStartCalibration;
      {To tell, that the vehicle is stopped}

      procedure GyroStopCalibration;
      {Must be called before a move again}

      function GyroAngle: radians;
      {Gives the angle in radians,(all the angle, not modulo 2π)}
      {Positive: counterclockwise}

      procedure GyroUpdate;
      {Must be called exactly periodically}

implementation
      uses  ADCInterface; { This implements the procedure GetADC, gets the voltage}
                          { in 2-complement, 16 Bit wide, 0 near to 2.5V}
      const
            Weight = 40;
            SecToWait = 2;
            FixPoint = longint($40);
            Scalefactor = 61.4317E-9;


      var
            CurrentAngle, CurrentAngleVelocityAbs,
            CurrentAngleVelocityRel, Offset, OffsetFix: longint; {32 Bit}
            CycleTime, Count: integer; {16 Bit}
            GivenOffset: real;
            Calibrate: boolean;

      procedure GyroInit (time: integer);
            const n = 1000;
            var   i: integer;
      begin
            CurrentAngle := 0;
            OffsetFix := 0;
            GivenOffset := 0;
            CycleTime := time;
            Count := 0;
            { Do some smoothing}
            for i := 1 to n do
                  OffsetFix := (OffsetFix * (Weight - 1) + GetADC * FixPoint) div Weight;
            Offset := OffsetFix;
            Calibrate := false;
      end;


      procedure GyroUpdate;
      begin
            CurrentAngleVelocityAbs := GetADC * FixPoint;
            CurrentAngleVelocityRel := CurrentAngleVelocityAbs - Offset;
            CurrentAngle := CurrentAngle + CurrentAngleVelocityRel
            OffsetFix := (OffsetFix * (Weight - 1) + CurrentAngleVelocityAbs) div Weight;
            if Calibrate then
                  if count < SecToWait * CycleTime then
                        Count := Count + 1
                  else
                        Offset := OffsetFix
            else
                  count := 0;
      end;

      procedure GyroSetAngle (ExtOffset: radians);
      begin
            CurrentAngle := 0;
            GivenOffset := ExtOffset;
      end;

      procedure GyroStartCalibration;
      begin
            Calibrate := true;
```

```
    end;

    procedure GyroStopCalibration;
    begin
        Calibrate := false;
    end;

    function GyroAngle: radians;
    var   InternAngle: radians;
    begin
        InternAngle := (CurrentAngle div FixPoint) * Scalefactor * CycleTime +
                       GivenOffset;

        GyroAngle := InternAngle
    end;
end.
```

Results

With this sample code, after a warm up of 10 minutes, it was possible to keep below of a angle drift of 1°/min. . Periodic calibrations where necessary about every 2 minutes, but we used no form of shelter for the device to kept it out of cooling air.