REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE ***(DD-MM-YYYY)*** 2018-06-14	**2. REPORT TYPE** Technical		**3. DATES COVERED** ***(From - To)*** DEC2017 – APR2018
4. TITLE AND SUBTITLE Temperature Response Comparison of 3-Axis Gyroscopes			**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Ty Valascho, Tim Pietrzyk			**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397			**8. PERFORMING ORGANIZATION REPORT**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397			**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A. Approved for public release; distribution unlimited.			
13. SUPPLEMENTARY NOTES The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an Official Department of the Army position, policy, or decision, unless so designated by other documents.			
14. ABSTRACT Gyroscope sensor data was gathered from three different models of 9DOF IMU sensors: Kootek, Microstrain, and Phidget. Two sensors of each model were tested in a temperature chamber on each of the three axis X, Y, and Z and at each of three temperature points 0, 25, and 65 C. Three test runs were run at each test point. The data from this testing shows that the Microstrain sensed values were less accurate than the Kootek sensed values. Phidget sensed values were not able to be analyzed as the data was incomplete, despite rerunning the test using a third sensor.			
15. SUBJECT TERMS Robotics, gyroscope, testing			
16. SECURITY CLASSIFICATION OF: DISTR A			**17. LIMITATION OF ABSTRACT** / **18. NUMBER OF PAGES** 24 / **19a. NAME OF RESPONSIBLE PERSON** Ty Valascho
a. REPORT DISTR A	**b. ABSTRACT** DISTR A	**c. THIS PAGE** DISTR A	**19b. TELEPHONE NUMBER** ***(include area code)*** (586) 459-7485

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

US Army - TARDEC

Temperature Response Comparison of 3-Axis Gyroscopes

Tyrus Valascho, Tim Pietrzyk
6-14-2018

Executive Summary: Gyroscope sensor data was gathered from three different models of 9DOF IMU sensors: Kootek, Microstrain, and Phidget. Two sensors of each model were tested in a temperature chamber on each of the three axis X, Y, and Z and at three temperature points 0, 25, and 65^{o}C. Three test runs were conducted at each test point.

The data from this testing shows that the Microstrain sensed values were less accurate than the Kootek sensed values. The Phidget qyroscope values were not able to be analyzed as the data was incomplete, despite rerunning the test using a third sensor.

Contents

Introduction

IMU (Inertial Measurement Unit) sensors are used in a variety of applications to measure and monitor movement and orientation. Many IMUs have at least 6 sensors in a specific configuration, also called Degrees of Freedom (DOF). Linear acceleration in three dimensions (Ax / Ay / Az) is often measured by placing accelerometers in the three directions. Rotational rate (Gx / Gy / Gz) is usually measured by aligning 3 gyroscopic sensors in the three axes.

The sensed data provided by IMUs is useful but it is not perfect. Many IMUs are based on MEMS (Microelectromechanical systems) technology, which has some known shortcomings. Gyroscopic MEMS sensors are prone to drift and MEMS Accelerometers tend to have difficulties with signal stability. Both problems can be overcome or minimized with proper software design and filtering, but for purposes of this testing, raw data is most important.

Performance Test Procedure

The purpose of the testing was to compare operation of the 3-axis gyroscopes of IMUs at different temperatures. A rotational test fixtures was prepared to perform this work. Each IMU was then tested on each of three axes rotationally to measure the gyroscope signals.

Testing was performed by measuring the IMU response under repeatable rotational profiles at three temperature points: 0^{o}C, 25^{o}C, and 65^{o}C. The sensor and test chamber were soaked at the test temperature for at least 1 hour prior to executing each test run.

Rotational Test

To measure the gyroscope accuracy, the sensor was rotated for 0.9 seconds in the clockwise (CW) direction, then rotated 0.9 seconds in the counter clockwise (CCW) direction.

A typical plot of the voltage supplied to the motor used for the rotational testing is provided in Figure 1.

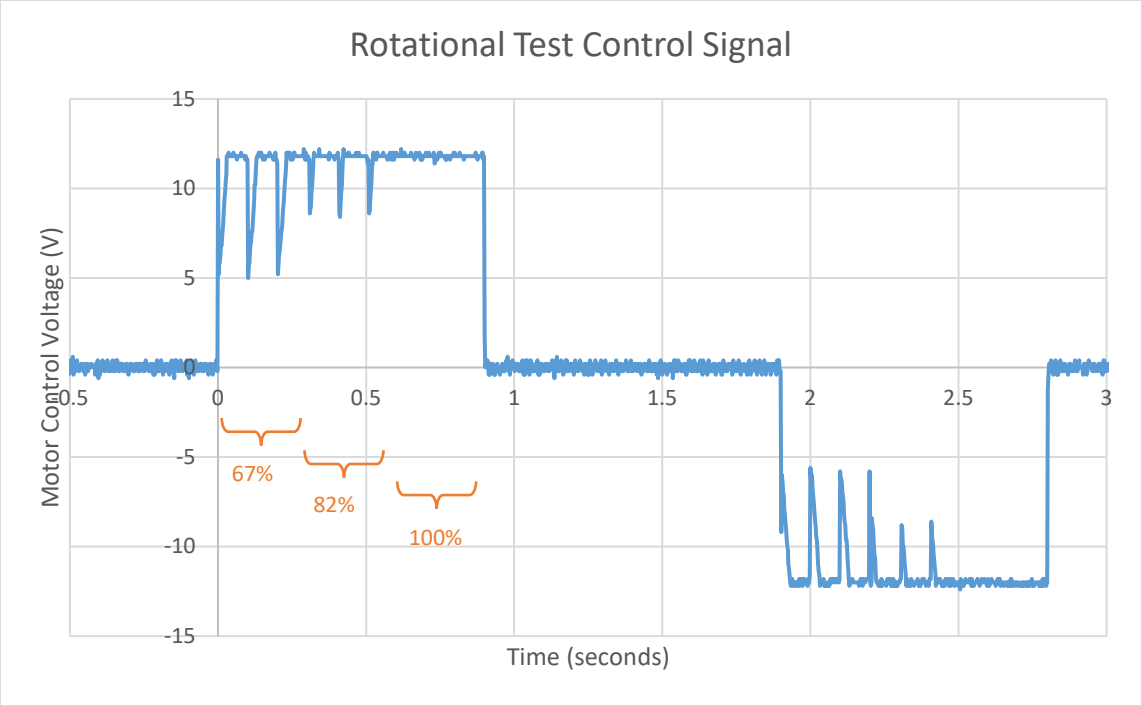

Figure 1 - Rotational Test Control Voltage Commanded

A PWM value was used to control the motor, and increasing PWM values of 67%, 82%, and 100% were commanded to the motor in 300 millisecond increments. Although Figure 1 only labels the command in the forward direction, the same pattern was used in the reverse direction.

Each test was performed three times on two sensors.

The motor used for rotational testing was an Anaheim Automation BDPG-36-57-24V-4200-R27 integrated DC motor and gearbox, controlled by an Arduino Mega 2560. The STMicroelectronics L298N H-bridge motor driver chip was used for the motor control.

There are three main parts of the rotational test setup, as illustrated in Figure 2 – the DV Motor, the Adapter, and the Sensor Cage.

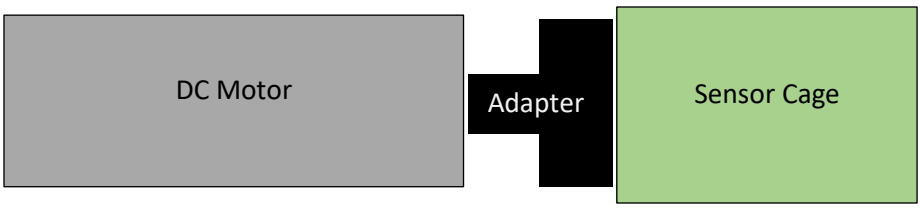

Figure 2 - Rotational Test Setup

In this test configuration, the motor is commanded to rotate in the CW direction, then CCW. This moves the adapter which in turn rotates the Sensor Cage.

The Sensor Cage houses the sensor that is being tested and is designed so that the cage can be mounted in each of the three axes such that the axis of rotation passes through the center of the sensor in each direction. The mounting pattern is 4 holes at the corners of a 20mm square on the Adapter and a combination of 2 holes and 2 posts on the Sensor Cage. The Sensor Cage is shown in Figure 3, with the Z-axis and X-axis mounting locations shown.

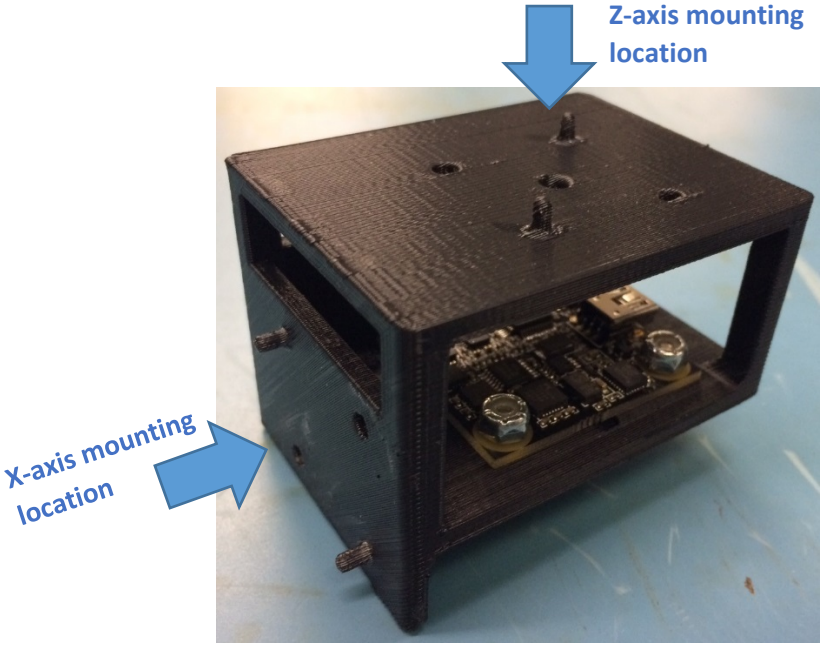

Figure 3 - Sensor Cage with a Phidget Sensor Installed

The Y-axis mounting location is on the back of the Sensor Cage and cannot be seen from this perspective. Figure 4 shows a CAD model of the sensor cage, which was fabricated and attached to the test fixture.

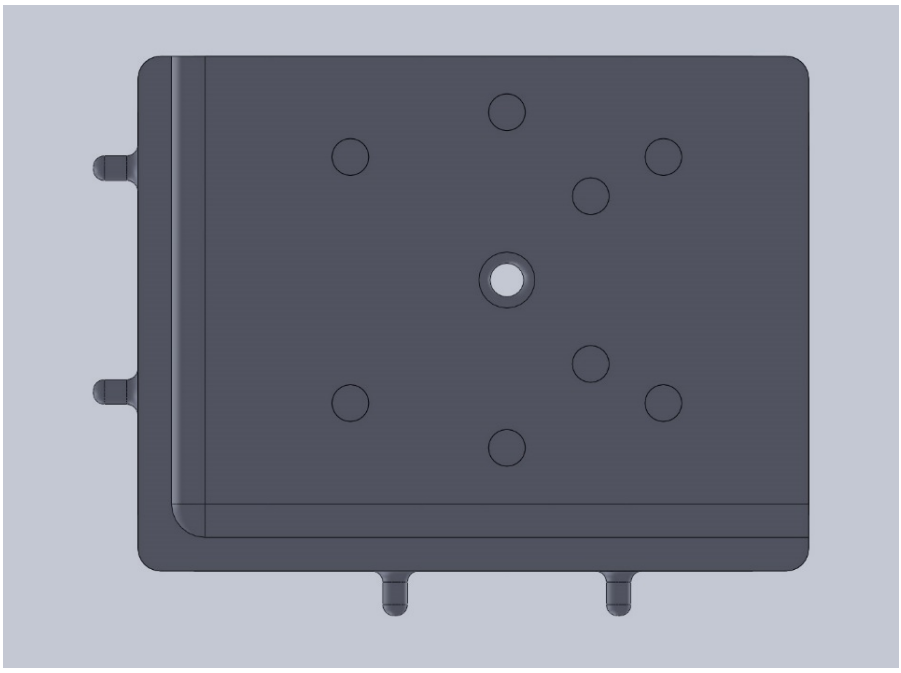

Figure 4 - Bottom View of Sensor Cage

A view of the entire test apparatus inside the temperature chamber is provided in Figure 5.

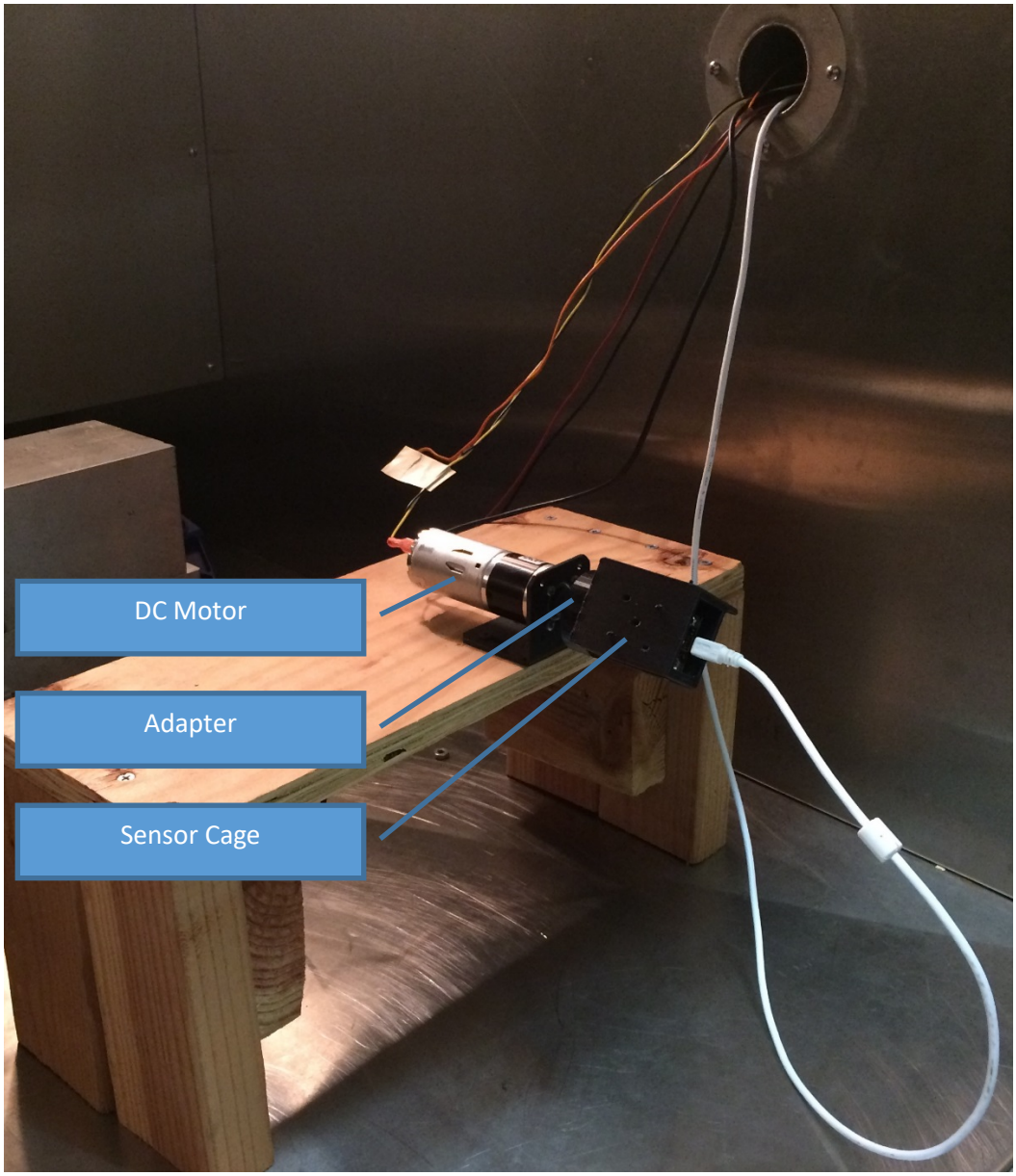

Figure 5 - Rotational Test Setup in Temperature Chamber

Performance Test Results

Test Articles

Two samples of each IMU product were tested three times in each of the 3 test cases: Gx, Gy, Gz. Technical details about the tested sensors are provided in Table 1.

Table 1 - Tested IMUs

Manufacturer	Model Number	Serial Number	Firmware	Notes
Phidget	1044_0	353985	402	
Phidget	1044_0	354963	402	
Phidget	1044_0	354002	401	
Microstrain	3DM-GX3 -35	6225-4220.39314	1.1.19	
Microstrain	3DM-GX3 -35	6225-4220.01431	1.1.19	
Kootek	GY-521	"A"	N/A	Uses the InvenSense MPU-6050
Kootek	GY-521	"B"	N/A	Uses the InvenSense MPU-6050

The Phidget and Microstrain products perform internal filtering of the raw values, but the Kootek does not.

Data Collection Methodology

The physical alignment of the sensors was based on how the sensor fit into the sensor cage with enough cabling to run the test. It did not necessarily correspond to the sensor's axes of orientation. To make the results consistent, the sensor cage axis was mapped to the sensor's understanding of axis. The mapping provided in Table 2 indicates what the sensor's axis was with respect to the corresponding Sensor Cage axis.

Table 2 - Sensor Axis Mapping

Sensor Cage Axis	Kootek	Microstrain	Phidget
Gx	-Gx	-Gx	Gy
Gy	-Gy	-Gy	-Gx
Gz	-Gz	-Gz	Gz

Measurement of the 3 sensors was through serial messages, but a different interface was employed for each.

To log data from the Phidget sensor, the example C# software "Spatial"[1] was modified to log the gyroscope and accelerometer values to a text file whenever they are displayed to the screen. This results in data being logged every 32 milliseconds. Every test run was saved as a separate txt file.

The Microstrain sensor logging was performed using Microstrain's "MIP Monitor" application, and the data stored as a csv file. Every test ran was saved as a separate csv file. The frequency of logging was 100 milliseconds.

The Kootek sensor communicates over I2C, which was read using an Arduino Mega 2560 board and logged every 30 milliseconds. The data was captured to the screen using a custom Arduino program, then copied into an excel spreadsheet with a file being recorded for each test run. The logging software for this is provided in the Appendix "Kootek Sensor Logging Software."

[1] The example software can be downloaded from the Phidget website: https://www.phidgets.com/?view=code_samples&lang=CSharp

The units from each logging software was configured as shown in Table 3.

Table 3 - Units of Logging Software

Units	Kootek	Microstrain	Phidget
Gyroscope	Degrees / second (65.5 degrees per bit)	Radians / second	Degrees / second

Results

Data was gathered from all sensors and test temperature conditions.

Post processing involved the following steps:

- Remap the data in accordance with Table 2 so all sensed values were with respect to the Sensor Cage
- Remove data that was not pertinent to what was being studying – specifically, the conditions before and after a test cycle
- Scale the data as needed so that all gyroscope units were Degrees / second
- The gyroscope data from the Kootek sensor had no filtering applied to it, so a 3 element median filter was applied to it to remove outlying sensed values and the sensor drift was removed

Idealized Response

To analyze the data, an idealized response for each plot was created. The idealized response was based on the maximum value measured during that test run and 2 additional steps of 67% and 82% of that maximum added which correspond to the PWM used to control the motors. An idealized value for a temperature was created by averaging all the idealized responses calculated at that temperature. A plot of the idealized temperature response at 25^{o}C versus a particular 25^{o}C test run is provided in Figure 6.

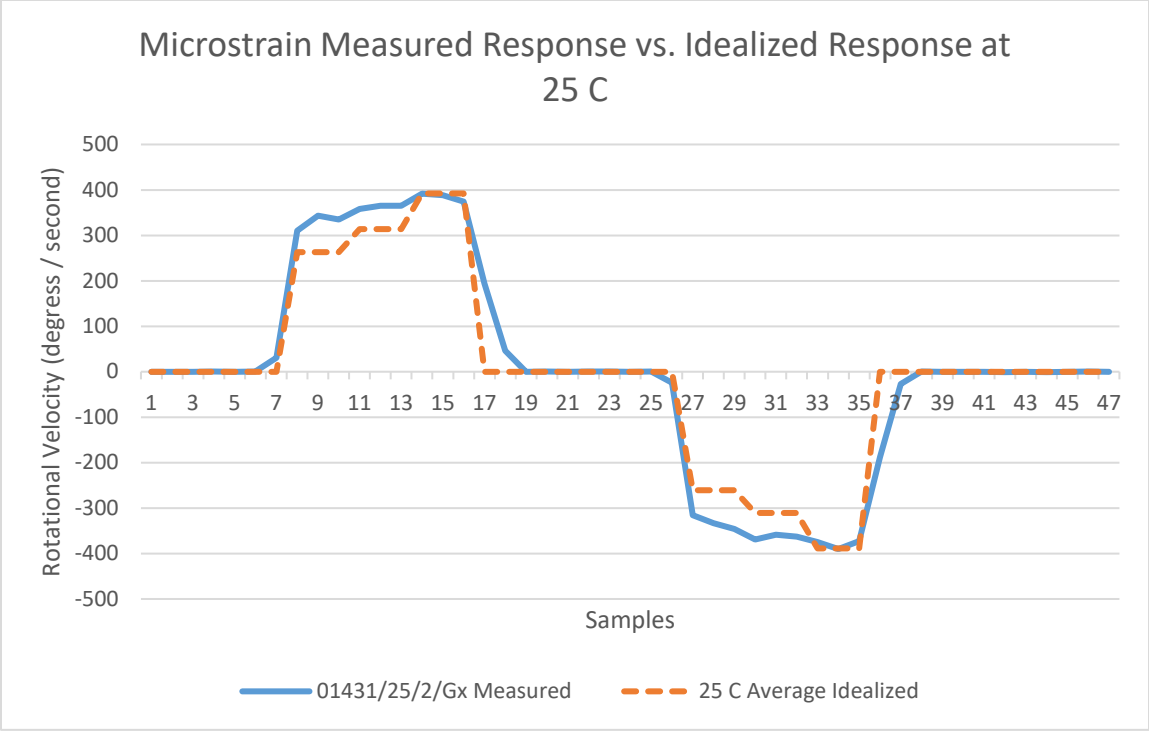

Figure 6 - Measured vs. Idealized Sensor Response

Because of the quantity of data that was captured, a shorthand was developed for identifying specific test run data using the following format, which is used in Figure 6:

<Sensor Serial Number>**/**<Temperature>**/**<Test Run>**/**<Measurement Axis>

In this example, a Microstrain test run:

01431/25/2/Gx

The values are:

Serial Number: 01431

Temperature: 25 C

Test run: 2

Measurement Axis: Gx

From the idealized response, the absolute value of the difference between each response measurement and the idealized response for that temperature was calculated. The non-movement parts of the response curves were ignored. Figure 7 shows the difference between the measured response and idealized response at 0^{o}C for the Kootek sensor.

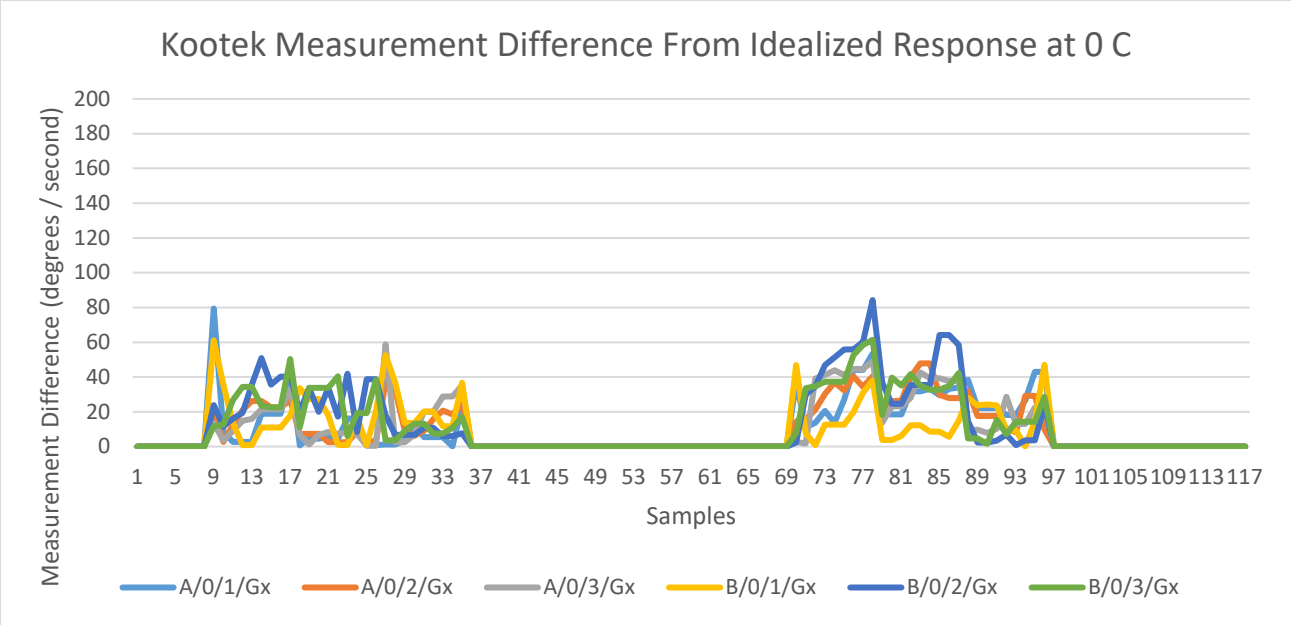

Figure 7 - Difference of Kootek Sensor Measurements at 0 C

These values were then averaged across all 3 test runs and each of the two sensors tested to generate a single number called “Average Difference” to compare the relative proximity of the values to the idealized response. A low value indicates a closer match to the idealized response than a higher value.

The complete set of the measured vs. idealized difference calculation data is provided in Table 4.

Table 4 - Summary of Measured vs. Idealized Difference Calculations

Temperature (C)	Axis	Sensor	Average Difference (deg/sec)	Std Deviation of Difference (deg/sec)	Maximum Difference (deg/sec)	Minimum Difference (deg/sec)
0	**Gx**	**Kootek**	19.7	16.5	84.3	0.0
0	**Gx**	**Microstrain**	30.0	26.6	107.6	0.0
0	**Gy**	**Kootek**	21.0	27.0	197.6	0.0
0	**Gy**	**Microstrain**	46.7	35.5	133.6	0.0
0	**Gz**	**Kootek**	25.1	37.2	250.2	0.0
0	**Gz**	**Microstrain**	43.1	34.0	127.9	0.0
25	**Gx**	**Kootek**	31.0	24.8	82.6	0.0
25	**Gx**	**Microstrain**	64.0	40.1	157.0	0.0
25	**Gy**	**Kootek**	27.6	28.6	237.6	0.0
25	**Gy**	**Microstrain**	81.9	46.2	181.0	0.0
25	**Gz**	**Kootek**	33.5	35.1	247.3	0.0
25	**Gz**	**Microstrain**	87.3	39.0	173.7	1.7
65	**Gx**	**Kootek**	55.6	32.9	119.3	0.0
65	**Gx**	**Microstrain**	97.0	39.9	192.0	26.9
65	**Gy**	**Kootek**	49.9	38.8	252.9	0.0
65	**Gy**	**Microstrain**	107.7	57.0	238.3	0.0
65	**Gz**	**Kootek**	50.8	43.7	370.3	0.0
65	**Gz**	**Microstrain**	118.5	47.3	221.3	3.3

Most of the Phidget sensed responses were incomplete, especially at the 25 and 65^{o}C temperature points. It was theorized that this was due to a poor USB connection from the data-logging computer to the sensor, so the test runs were repeated using a third sensor and with special care taken to observe the cabling and connections. These test runs showed similar results, so it was determined that the Phidget sensors or the logging program were not providing correct data and the results could not be used.

An example of this behavior is provided in Figure 8, where three response curves are provided that were taken using the Phidget sensor, serial number 353985.

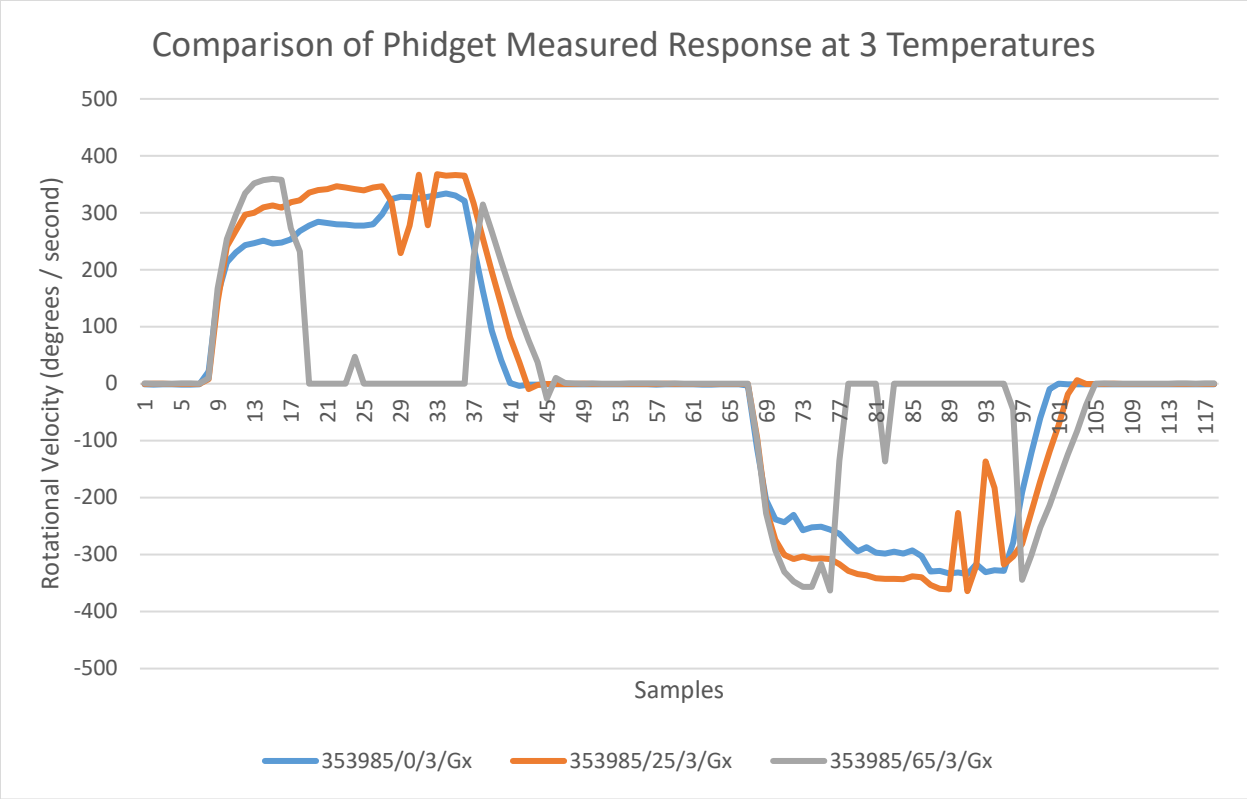

Figure 8 - Comparison of Phidget Response at 3 Temperatures

In this plot, the 65°C data plot can be seen experiencing the data loss issue.

Analysis

Product Variation

The average difference values are a measurement of how accurate the sensed values are when compared to the idealized response. Higher values mean less accurate sensor responses.

When compared over all the sensor values at all temperatures, the Microstrain sensor data has higher and therefore less accurate values, as seen in Figure 9. A second point that is illustrated by this plot is that both sensors see a loss in accuracy as the temperature is increased.

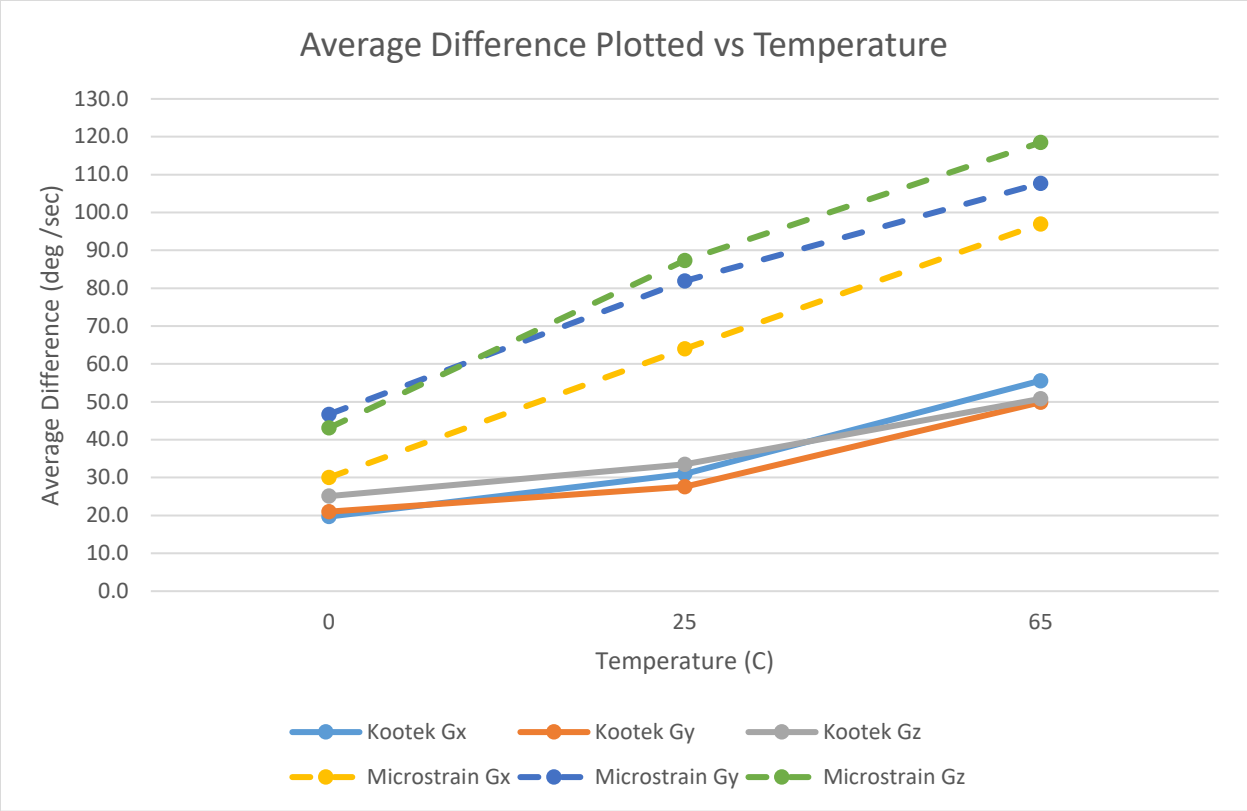

Figure 9 - Average Difference vs Temperature

Conclusions and Recommendations

Gyroscope data was gathered from three different models of 9DOF IMU sensors: Kootek, Microstrain, and Phidget. Two sensors of each model were tested in a temperature chamber on each of the three axis X, Y, and Z and at each of three temperature points 0, 25, and 65^{o}C.

The data from this testing shows that the Microstrain sensed values were less accurate than the Kootek sensed values. Phidget values were not able to be analyzed as the data was incomplete, despite rerunning the test using a third sensor.

Possible source of problems with the data:

- Inconsistent / imprecise attachment of motor to sensor cage: this resulted in some vibration during movement which could affect results
- Misalignments in each direction (X / Y / Z): the mechanical tolerances of the various parts of the system resulted in motion that was not purely in the rotational axis being measured
- Behavioral differences of the DC motors at different temperatures: the DC motor used for this testing was designed for use within the temperature range tested, but the behavior may not be consistent at each temperature tested
- Incorrect PWM Command: the PWM signal of the motor controller was not a proper PWM signal, which resulted in a higher commanded speed than the ideal response plot is modeling

Incorrect PWM Command

Of the possible problems with the data, the last issue was looked at more closely to determine how much of an effect it had on the results.

The calculation of the idealized response plot was based on the software commanded PWM values of 67%, 82%, and 100%. However, a closer examination of the measured voltage in the PWM signal shown in Figure 1 shows that the actual voltage values were not consistent with the commanded values. PWM signals should be at zero during "off" periods and at the max voltage during the "on" cycles, with the ratio of "off" to "on" time being the same as the PWM. In reality, the voltages never were completely "off" which would result in the motor moving faster than the software commanded it. This phenomenon is depicted in Figure 10.

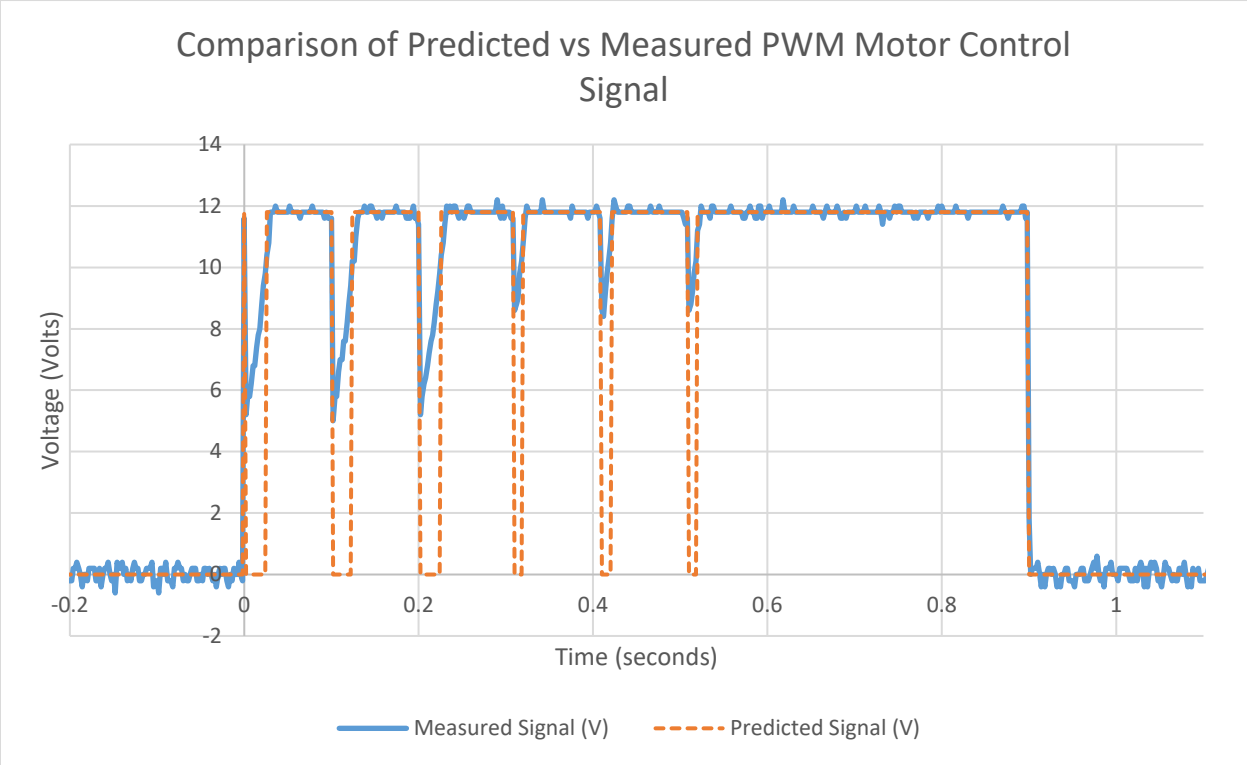

Figure 10 - Predicted PWM Voltages vs Measured PWM Voltages

By looking at the measured PWM voltages instead of the predicted voltages, it can be determined that the actual PWM duty cycles are 90% instead of 67% and 97% instead of 82%. These values cause recalculation of the idealized responses and measured vs. idealized difference calculations. After this recalculation, the results in Table 5 are provided.

Table 5 - Summary of Measured vs. Idealized Difference Recalculations

Temperature (C)	Axis	Sensor	Average Difference (deg/sec)	Std Deviation of Difference (deg/sec)	Maximum Difference (deg/sec)	Minimum Difference (deg/sec)
0	**Gx**	**Kootek**	36.9	29.0	163.5	0.0
0	**Gx**	**Microstrain**	26.6	19.2	64.7	0.0
0	**Gy**	**Kootek**	55.3	42.9	284.7	0.0
0	**Gy**	**Microstrain**	29.1	22.5	74.4	0.0
0	**Gz**	**Kootek**	58.1	55.7	336.0	0.0
0	**Gz**	**Microstrain**	24.2	20.2	66.9	0.0
25	**Gx**	**Kootek**	20.4	20.7	120.9	0.0
25	**Gx**	**Microstrain**	37.7	31.3	109.5	0.0
25	**Gy**	**Kootek**	33.9	40.6	324.7	0.0
25	**Gy**	**Microstrain**	61.6	30.1	111.5	0.0
25	**Gz**	**Kootek**	32.7	49.9	333.0	0.0
25	**Gz**	**Microstrain**	62.0	24.4	107.2	1.7
65	**Gx**	**Kootek**	19.3	15.3	70.9	0.0
65	**Gx**	**Microstrain**	68.9	32.4	151.8	5.0
65	**Gy**	**Kootek**	24.1	47.7	340.0	0.0
65	**Gy**	**Microstrain**	88.4	38.6	152.8	0.0
65	**Gz**	**Kootek**	28.4	54.3	370.3	0.0
65	**Gz**	**Microstrain**	92.6	32.2	141.6	3.3

Plots of the original data vs the recalculated data are provided in Figure 11 and Figure 12 for the Kootek and Microstrain sensors, respectively.

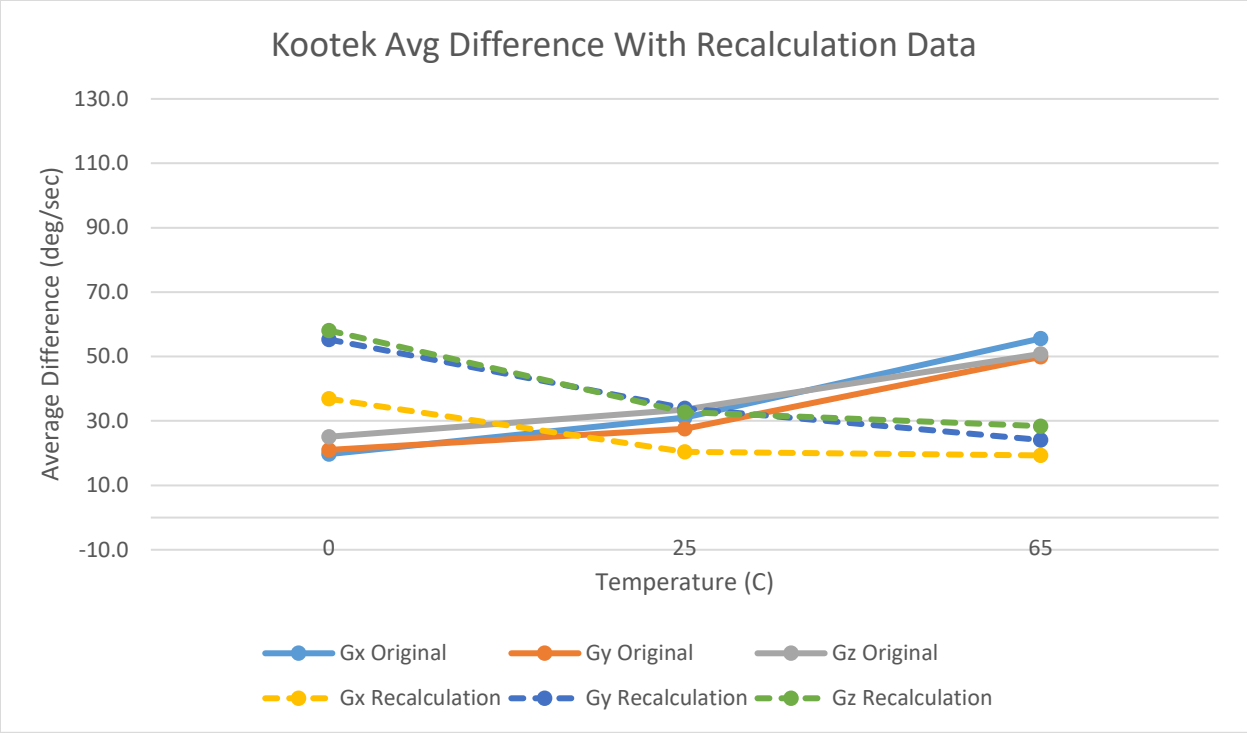

Figure 11 - Average Difference Recalculation Comparison, Kootek Sensor

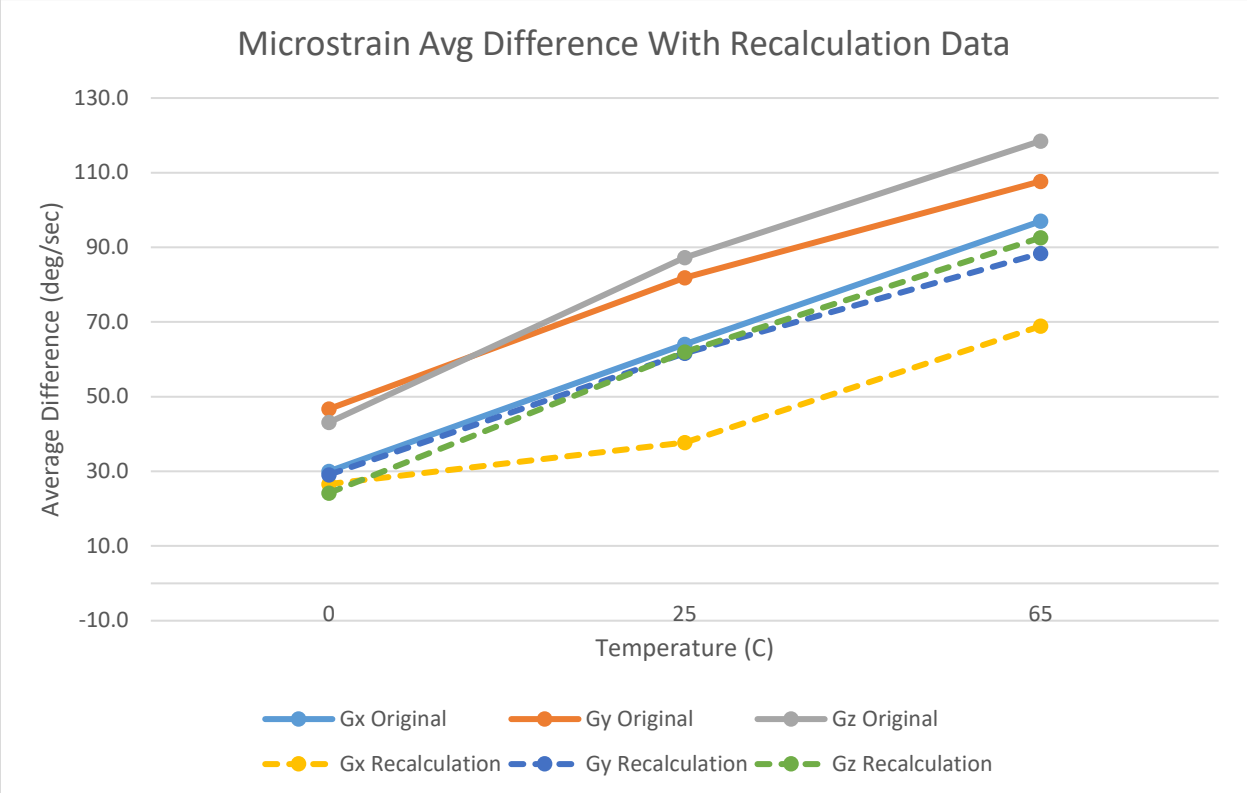

Figure 12 - Average Difference Recalculation Comparison, Microstrain Sensor

Recommendations

These recalculated results are more likely to represent the true response of the systems measured. In both the original data and the recalculated data, the Kootek sensor provides more accurate sensor responses in all cases except the 0 C recalculated data.

It is recommended that this testing be improved and performed again to verify results.

One improvement would be to increase the logging frequency of the Microstrain sensor. The other sensors were tested at approximately 3 times the frequency of the Microstrain sensor which may affect the accuracy of the data due to noise and also makes post-processing of the data more difficult, since the data curves cannot be accurately synchronized in time.

A second improvement would be a more sophisticated mechanical test fixture to reduce vibration and misalignment of the sensors. One way to achieve this would be to have the end opposite the motor be in a fixed location, possibly with a radial bearing to reduce friction.

A third improvement would be to have all sensors mounted together and tested simultaneously. This would reduce the test to test variation because all sensors would be experiencing the same movements. It would also reduce the amount of time spent testing and changing out the sensor cages.

The basic test plan should also be run on other gyroscope sensors for comparison.

Appendices

Abbreviations and Acronyms

CW: clockwise

CCW: counter clockwise

DOF: Degrees of Freedom

IMU: Inertial Measurement Units

MEMS: Microelectromechanical Systems

PWM: Pulse Width Modulation

Test Rig Motor Control Source Code

The source code is provided for a basic motor controller using Arduino Mega 2560. To control the motor, connect a laptop to the Arduino via USB and use a serial program such as Hyperterm. Connect at 9600 Baud with 8/1/none/HW flow control. Typing a “T” in the terminal will command the motor to run the rotational motion profile. Typing an “L” in the terminal will command the motor to run the linear motion profile (not used for this testing). The motor can also be advanced forward or backward by typing “F” or “B” respectively.

```
// Attach the L298 as follows
// Pin1 -> gnd
// Pin2 -> output of motor1
// Pin3 -> output of motor2fb
// Pin4 -> high power motor voltage supply (PS or battery) max 46 V
// Pin5 -> Pin6 of Arduino
// Pin6 -> Enable - controls motor speed (PWM) Pin2 of Arduino
// Pin7 -> Pin7 of Arduino
// Pin8 -> Attach to gnd of batt/PS and gnd of Arduino
// Pin9 -> 5V output of Arduino

const int motor_speed = 2; // This is the arduino pwm control pin for mtr speed
const int dir_pin_IN1 = 6; // Digital out to control motor direction
const int dir_pin_IN2 = 7; // Digital out to control motor direction

void setup() {
  // put your setup code here, to run once:
  pinMode(motor_speed, OUTPUT);
  pinMode(dir_pin_IN1, OUTPUT);
  pinMode(dir_pin_IN2, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  unpower_motor();
  delay(3000);
  int time_step = 300; // In milliseconds
  char rec_byte = 0;
  if (Serial.available() > 0) {
    // read the incoming byte:
    rec_byte = Serial.read();

    // If a T is rec'd complete a rotational test cycle: fwd then bwd
    if((rec_byte == 't') || (rec_byte == 'T')) {
      move_motor(true, 170);
      delay(time_step);
      move_motor(true, 210);
      delay(time_step);
      move_motor(true, 250);
      delay(time_step);
      brake_motor();
      delay(1000);
      move_motor(false, 170);
      delay(time_step);
      move_motor(false, 210);
      delay(time_step);
      move_motor(false, 250);
      delay(time_step);
    }

    // If a L is rec'd complete a linear test cycle: fwd then bwd
```

```
    if((rec_byte == 'l') || (rec_byte == 'L')) {
      move_motor(true, 1023);
      delay(time_step);
      delay(time_step);
      delay(time_step);
      delay(time_step);
      delay(time_step);
      brake_motor();
      delay(1000);
      move_motor(false, 1023);
      delay(time_step);
      delay(time_step);
      delay(time_step);
      delay(time_step);
      delay(time_step);
    }

    // If an F is rec'd move foreward
    if((rec_byte == 'f') || (rec_byte == 'F')) {
      move_motor(true, 170);
      delay(time_step);
      move_motor(true, 210);
      delay(time_step);
      move_motor(true, 250);
      delay(time_step);
      brake_motor();
      delay(2000);
    }

    // If a B is rec'd move backwards
    if((rec_byte == 'b') || (rec_byte == 'B')) {
      move_motor(false, 170);
      delay(time_step);
      move_motor(false, 210);
      delay(time_step);
      move_motor(false, 250);
      delay(time_step);
      brake_motor();
      delay(2000);
    }
  }
}

// Function to put the motor into unpowered mode
void unpower_motor(){
  Serial.println("unpowered");

  digitalWrite(dir_pin_IN1, LOW);
  digitalWrite(dir_pin_IN2, LOW);
  analogWrite(motor_speed, 0);
}
```

```
// Function to brake the motor
void brake_motor(){
  Serial.println("brake");
  digitalWrite(dir_pin_IN1, HIGH);
  digitalWrite(dir_pin_IN2, HIGH);
  analogWrite(motor_speed, 0);
}

// Function to move the motor
// direction = true means forward direction, false means backward
// If power supply is 12 VDC, min movement occurs around pwm = 60 or so.
void move_motor(bool direction, int speed){

  if(direction == true)
  {
    Serial.print("move fwd ");
    Serial.println(speed, DEC);
    digitalWrite(dir_pin_IN1, HIGH);
    digitalWrite(dir_pin_IN2, LOW);
  }
  else
  {
    Serial.print("move bwd ");
    Serial.println(speed, DEC);
    digitalWrite(dir_pin_IN1, LOW);
    digitalWrite(dir_pin_IN2, HIGH);
  }
  analogWrite(motor_speed, speed);
}
```

Kootek Sensor Logging Software Source Code

Source code for Kootek data collection using Arduino Mega 2560 is provided here.

```
// Based on code from:
// MPU-6050 Short Example Sketch
// By Arduino User JohnChi
// August 17, 2014
// Public Domain
#include<Wire.h>

const int MPU_addr=0x68;  // I2C address of the MPU-6050
float _AcX,_AcY,_AcZ,_Tmp,_GyX,_GyY,_GyZ; // Raw values read from sensor
uint8_t retval;
#define GYRO_CONFIG_ADDR 0x1B // Set range to 0x00 for 250, 0x01 for 500, and 0x02 for 1000, or
0x03 for 2000 (all are in deg/sec)
#define ACCEL_CONFIG_ADDR 0x1C // Set range to 0x00 for 2, 0x01 for 4, and 0x02 for 8, or 0x03 for
16 (all are in g)
```

```
void setup()
{
  Wire.begin();

  Serial.begin(38400);
  if (retval) {
    Serial.print(F("Wake up of MPU failed: "));
    Serial.println(retval);
  }

  // Configure the range of the sensor
  // Default range of sensor is only 250 deg/sec which causes saturation of the sensor for this test, we
move it faster than that.
 // Kootek (MPU-6050) issue – saturation of the values.  “The default setting in the I2Cdevlib class is +/-
2g for the accel and +/- 250 deg/sec for the gyro” (from https://www.i2cdevlib.com/forums/topic/4-
understanding-raw-values-of-accelerometer-and-gyrometer/).
  // Send the value of 0x00 for default 250 deg/s, 0x08 for 500 deg/s, 0x10 for 1000 deg/s, and 0x18 for
2000 deg/s
  Wire.beginTransmission(MPU_addr);
  Wire.write(GYRO_CONFIG_ADDR);
  Wire.write(0x08); // Set range to 500 deg/sec, default is only 250 which allows saturation of the sensor
for this test
  retval = Wire.endTransmission(true);

  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0x00);     // set to zero (wakes up the MPU-6050)
  retval = Wire.endTransmission(true);

  Serial.println();
  Serial.println("Ax\tAy\tAz\tGx\tGy\tGz");
}

int read_sensors()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  retval = Wire.endTransmission(false);
  if (retval)
  {
    Serial.print(F("Setting register number failed: "));
    Serial.println(retval);
  }
  retval = Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  if (retval != 14)
  {
    Serial.print(F(" Read registers not complete, read "));
    Serial.print(retval);
    Serial.println(F(" bytes."));
  }
  _AcX = Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
```

```
  _AcY = Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  _AcZ = Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  _Tmp = Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  _GyX = Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  _GyY = Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  _GyZ = Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
}

void loop()
{
    read_sensors();
    Serial.print(_AcX, DEC);
    Serial.print("\t");
    Serial.print(_AcY, DEC);
    Serial.print("\t");
    Serial.print(_AcZ, DEC);
    Serial.print("\t");
    Serial.print(_GyX, DEC);
    Serial.print("\t");
    Serial.print(_GyY, DEC);
    Serial.print("\t");
    Serial.println(_GyZ, DEC);
    delay(20);
}
```

Microstrain Data Processing Macro

Written for MS Excel 2013

```
Sub process_microstrain_data_00()
' process_microstrain_data_00 Macro
' Keyboard Shortcut: Ctrl+m
'
    Columns("M:M").Select
    ActiveWindow.LargeScroll ToRight:=2
    Columns("M:AZ").Select
    Selection.Delete Shift:=xlToLeft
    Columns("D:L").Select
    Selection.AutoFilter
    ActiveSheet.Range("$D$1:$L$137").AutoFilter Field:=9, Criteria1:="<>"
    Selection.Copy
    Sheets.Add After:=ActiveSheet
    Range("A3").Select
    ActiveSheet.Paste
End Sub
```