Movement measurement device for airplanes

Sigríður Árný Júlíusdóttir

Lokaverkefni í rafmagnstæknifræði BSc

2014

Höfundur: Sigríður Árný Júlíusdóttir
Kennitala: 020271-5079
Leiðbeinandi: Joseph Foley

Tækni- og verkfræðideild
School of Science and Engineering

Tækni- og verkfræðideild

Heiti verkefnis:
Movement measurement device for airplanes

Námsbraut:
Rafmagnstæknifræði BSc

Tegund verkefnis:
Lokaverkefni í tæknifræði BSc

Önn:
Vorönn 2014

Námskeið:
Lok 1012

Höfundur:
Sigríður Árný Júlíusdóttir

Umsjónarkennari:
Baldur Þorgilsson

Leiðbeinandi:
Joseph Foley

Fyrirtæki/stofnun:

Ágrip:

This thesis discusses the process of putting together a accelerometer and gyro measurement device for cabin-quality research.

The device uses MPU6050 to collect measurements. A Raspberry Pi is used and programmed in python to control the data collection saved to a file on a SD card.

The data is defined based on the slope of two measured points.

Tests measurements show that the unit is very sensitive and not suitable for measurements of small movements.

Dagsetning:
13-Maí 2014

Lykilorð íslensk:
Hröðunarmælir.

Lykilorð ensk:
MPU6050,
Raspberry Pi,
accelerometer, gyro

Dreifing:
opin ☐ lokuð ☐ til:

Abstract

This thesis is a part of a BS degree in electrical engineering from the University of Reykjavík. The project is to find suitable components and put together accelerometer gyro measurement device for cabin-quality research that is small and has low power supply. The device uses inertial measurement unit MPU6050 to collect movement measurements from six-axis. A raspberry Pi was programmed in python to control the data collected and store data of interest to a file on a SD card. Saved data is time of the measurement and measurements of the six-axis in the unit g and °/s.
The data is defined, based on the slope of two measured points.
Tests measurements show that the unit is very sensitive and not suitable for measurements of small movements. By ignoring the low slopes it gives a good idea of its movements.

Further calculations can be done for instance in Matlab to get speed or movements.

Útdráttur

Þessi ritgerð er hluti af BS gráðu í rafmagnstæknifræði frá Háskólanum í Reykjavík. Hún fjallar um verkefnið að finna íhluti og setja saman til að gera lítinn hröðunar og gyro mælir sem nýtir litla orku sem nota á fyrir gæðakönnun í farþegarými flugvéla. Notaður er MPU6050 hröðunar og gyro skynjari til að mæla hreyfingar á sex ásum. Raspberry Pi tölva er forrituð með Python forritunarmáli til að safna áhugaverðum mælingum og vista á texta skjali sem er geymt á minniskorti. Gögnin sem vistuð eru eru tími mælinga og mælingar ásanna í einingunni g og °/s.
Hversu áhugaverðar mælingarnar eru er skilgreint útfrá hallatölu tveggja mælipunkta. Tilrauna mælingar sýna að búnaðurinn er mjög viðkvæmur og hentar ekki til mælinga á fínhreyfingum. Hann gefur þó góða mynd á þeim hreyfingum sem hann verður fyrir og með því að hunsa lágar hallatölur má fá góða mynd af hreyfingum flugvélar.

Frekari útreikninga má gera í t.d. Matlab til að finna hraða eða færslu.

Abbreviations

A	**A**mper
ADC	**A**nalog to **D**igital **C**onverters
CSI	**C**amera **S**erial **I**nterface
DC	**D**irect **C**urrent
DSI	**D**isplay **S**erial **I**nterface
GPIO	**G**eneral **P**urpose **I**nput and **O**utput
HDMI	**H**igh **D**efinition **M**ultimedia **I**nterface
IMU	**I**nertial **M**easurement **U**nit
LED	**L**ight **E**mitting **D**iode
MB	**M**ega**b**yte
OS	**O**perating **S**ystem
RAM	**R**andom **A**ccess **M**emory
RCA	Connector used to carry signals
RJ45	Ethernet cables plug
SD	**S**ecure **D**igital
USB	**U**niversal **S**erial **B**us
V	**V**olt

Table of Contents

1. Introduction

This report is a part of a B.Sc. degree in electrical engineering at Reykjavík University. The idea for the project came after discussions with Joseph Foley and Þorgeir Pálsson. This report is a part of a bigger project aiming on defining passenger´s comfort and quality in Icelandair passenger flights.

The aim of this project is to find suitable components for accelerometer gyro measurement device for cabin-quality research and program it so the device will collects information about movements in the passenger compartment.
The devise must be small and with low power consumptions.
Testing the unit is needed to determine how accurate it measurements are and how suitable it is for its purpose.

An inertial measurement unit (IMU) is used to measure 3-axis acceleration and 3-axis gyroscope in the unit g and deg/sec. MPU-6050 6-axis motion processor is used for this task. It has I2C bus connection allowing the device to gather data using only two wires, full scale range can be configured from ±2g to ±16g for the accelerometer and ±250°/s to 2000 °/s for the gyroscope. The MPU-6050 also uses ADCs for digitizing the accelerometer outputs. The Raspberry Pi is a suitable computer for this project since it runs on only 5V and is small. After sorting the data and writing it along with timestamp on a .txt file on the SD card the data can be analyzed and further calculations can be made. The program language is Python, since Idle is included in the Raspberry Pi downloadable software NOOBS.
All the software writing was made on the Raspberry Pi.

No research was found on measuring the g force in commercial aircraft or how the g force affect the passengers experience but quite few on high performance fighter aircraft and the pilot high g toleration.

2. The technology

The MPU-60X0 is the first integrated 6-axis motion tracking device that combines 3-axis accelerometer, gyroscope and a digital motion processor. The 3-axis and 3-gyroscope technology is widely used today. Part of the devise popularity is how small it is, the low power consumption and at a low consumer price. Smartphone and tablet manufacturers use this technology ability to accurately track motions for example in gaming control, screen viewing and navigation.
(InvenSense Inc., 2013)

Since 2007 most phones and tablets contain accelerometer. Answering a phone call by shaking it is based on accelerometer detecting the motion as well as if the phone is turned from being portrait to landscape the display can change accordingly.
3-axis accelerometer provide information about acceleration in the basic x- y- z-axis directions, 3-axis gyroscope provides measurement of rotations Yaw, pitch and Roll along the basic 3-axis. Signals from gyroscopes and accelerometer can be integrated with a compass sensor to detect 9-degrees of freedom motion. (Steve Nasiri)
This measurement movement device sets the axis according to the plane and is not related to earth's directional force except gravity.

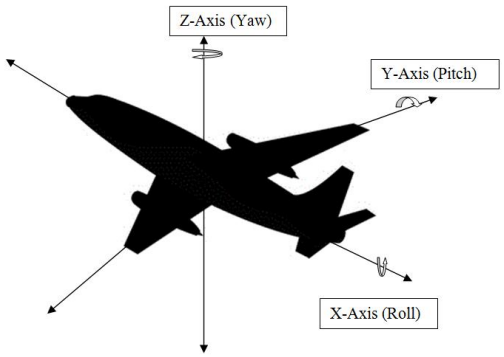

Figure 1 3-axis gyroscope detects roll, pitch and yaw

More progress in other fields is to be expected in this technology in the near future. In healthcare this technology can be used in monitoring the orientation of body limbs, in order to assess movement disorders, as fall sensor for the elderly or detect movements of any kind.

2.1. Raspberry Pi

The Raspberry Pi is a small single board computer not much bigger than a credit card and therefore ideal for this movement device since it has to be small.

Figure 2 Raspberry Pi board.

This project uses model B with 512 MB RAM Raspberry Pi and all information in this report apply to the B model it does not have a built in hard disk but uses an SD memory card as permanent memory and for booting.
The Raspberry Pi needs at least 4GB Class 4 SD card to work.
There are two USB 2.0 ports on the circuit board that can be used to connect accessories to the computer such as keyboard, mouse or a wireless network adapter.
The Raspberry Pi has one Ethernet port with standard RJ45 allowing the computer to connect to the internet via Cat5 cable.

HDMI port is used to connect a monitor, providing both digital- video and audio output from the computer.
The board also has analog audio output for standard 3.5mm jack and a standard RCA-type jack for composite video out.
The Raspberry Pi has no build in power switch and a micro USB power connector is used for supplying power to the computer. 5V DC and minimum of 0,7A are needed to run the computer. Its low electric consumption is one of its greatest advantages excluding its size. This movement device uses a standard Nokia phone charger with 5V output and current of 1A.
NOOBS for Raspberry Pi is used in this movement device and the Raspberry Pi is running on Linux and Raspbian software.
NOOBS is a freeware and can be downloaded from the Raspberry Pi home page.
Five status LED´s provide visual indication on the computers status.
The board also has 26 GPIO pins that can be used for input or output, display serial interface (DSI) as well as camera serial interface (CSI). (Matt Richardson, 2012)

2.2. IMU

An inertial measurement unit (IMU) is a unit that combines information from two or more sensors, for example gyros, accelerometers magnetometer or GPS.
The accelerometer measure acceleration due to movement and gravity while the gyro measure the rate of angular rotation. The accelerometer can be used to calculate a tilt angle. A 3-axis accelerometer can tell the bearings of item relative to earth´s surface. If the item is in free fall it will show zero acceleration for z-axis.

As the gyro measures the rate of rotation around the axis it will measure a non zero value as long as the aircraft is rolling but zero if the roll stops.
Combining acceleration and gyro can give much more advanced information about the total movements.

(Anderson, 2008)

2.2.1. MPU-6050

The MPU-6050 has the ability to accurately track motions, using 6-axis motion device that combines of 3-axis gyroscope, 3-axis accelerometer and a digital motion processor. In additional to the accelerometer and gyroscope the MPU-6050 includes a temperature sensor. The movement device uses a breakout board from Sparkfun making this tiny item easy to work with.

Figure 3 Sparkfun breakout board with MPU-6050 [1]

I2C is used to communicate with the Raspberry Pi at 400 Hz. In addition to the SCL and SDA connection the MPU-6050 provides a VLOGIC reference pin to set the logic levels of its I2C interface, the VLOGIC pin is connected to 3,3V. The analog supply VDD also uses 3,3volts. The MPU-6050 features three 16-bit ADC´s for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. (InvenSense Inc., 2013)

[1] Figure 3 (https://www.sparkfun.com/products/11028)

2.2.2. Registers

Register 26 configures the pin sampling and the digital low pass filter settings for both the accelerometers and gyroscopes.

Register 27 is the gyroscope configuration and is used to set the gyroscopes full scale range. The available full scale range outputs are ± 250°/s, ± 500°/s, ± 1000°/s, and ± 2000°/s. The movement device uses ± 250°/s.

Register 28 is for the accelerometer configuration. Available full scale range outputs are ±2g, ±4g, ±8g, and ±16g. This project uses ±2g.

Registers 59 to 64 are the accelerometer measurements. These registers store the most recent measurements for the three axes. For this movement device register 59 is used for reading the high bits of x-axis and register 60 for the low bits.
Register 61 is used for reading the high bits of y-axis accelerometer and register 62 for the low bits. Register 63 is used for reading the high bits of z-axis accelerometer and register 64 for the low bits.

Registers 67 to 72 are the gyroscope measurements. These registers store the most recent measurements for the three axes. For this project register 67 is used for reading the high bits of gyro x-axis and register 68 for the low bits.
Register 69 is used for reading the high bits of gyro y-axis and register 70 for the low bits. Register 71 is used for reading the high bits of gyro z-axis and register 72 for the low bits.

Register 107 is the power management. It is activated with write command in the beginning of the program since the device powers up in sleep mode.
(InvenSense Inc., 2013)

2.2.3. Raw data reading

The MPU-6050 features ADC that converts analog voltage to a binary number and then to a digital number in base 10 for reading, while analog accelerometers gives voltage level within a predefined range as output that since has to be converted to a digital value using ADC.
The MPU6050 gives 8-bit low and 8-bit high output, combined those bits give 16-bit as output for each of the 6 axis.

A mathematical relationship shows how the numbers of bits an ADC handles. For example, 16-bit output gives a value in the range of $\mathbf{2^{16} = 65536}$ or numbers on the scale 0 to 65535.

A16-bit ADC resolution with maximum 3,3V input is:

$$Volts = \frac{Vref}{Digital\ output}$$

$$\frac{3{,}3V}{2^{16}} = 0{,}5\text{mV}$$

(Measurement Computing Corporation, 2012)

By dividing the raw accelerometer data by a sensitivity scale factor of 16384 LSB/g for the setting ±2g the data can be converted into multiplies of g. Dividing the raw gyro data by sensitivity scale factor of 131 LSB/°/s for the setting ± 250°/s gives angular velocity in °/s. (InvenSense Inc., 2013)

2.2.4. I2C bus

I2C bus is a multi master bus able to transfer data between two equipment using only two lines, a serial data line and a serial clock line. When I2C is used to carry information each device is recognized by a unique address. (Philips Semiconductors, 1995)

This movement device is using I2C digital output for the motions of the axis and gyro. The I2C ports on the Raspberry Pi are not enabled by default and have to be configured before receiving the signals from the MPU-6050.

Running the command: *sudo nano /etc /modules* provides information about what kernel modules should be loaded at boot time.
Two lines need to be added here:
i2c-bcm2708
i2c-dev

When activating the I2C port in Linux the files that disables the I2C port has to be edited. These settings are stored in:
sudo nano /etc /modprobe.d /raspi-blacklist.conf
Showing two files:
blacklist spi-bcm2708
blacklist i2c-bcm2708
These files are commented out and disable by putting # in front of them.
(Monk, 2013)

2.3. G force

G force is a unit measuring the inertial stress on a body undergoing acceleration of gravity. The human body is used to earth gravity, when stationary the force felt is 1g force pulling the body down. When the body undergoes a change in speed and direction the g force increases in proportion to the rate of change.
G force in a(x) (x-axis) is front to back, a(y) is side to side and a(z) is head to toe. Increased force along x- axis (a(x)) will pull a body forwards or pinned to the seat while increased force along z-axis (a(z)) will make the body feel heavier.
How much g force the body can tolerate depends on many factors such as where the forces are applied on the body and what direction they are coming from and the duration that they last. (Voshell, 2008)

Even though the human body can tolerate a significant amount of force in short amounts of time such as in crash situations this movement device focus is more on being able to measure the combined g force and gyro movements inside the airplane and compare with how the onboard passengers experience the flight and connection to motion sickness.

Following table contains example of movements and g force.

G Force (z)	Example
1	Standing.
1.2	Normal elevator acceleration.
1.5	Commercial airliner during takeoff run.
2	Commercial airliner at rotation.
3.5	Maximum acceleration in amusement park rides
4+	Carrier based aircraft launch.
10	The NASA g-force simulator is limited to 10 g
20	Possibility of injury of organs moving.
30	Test dummies for air bag systems.

Figure 4 Example of movements and g force

(CNN/Sports Illustrated, 2001)

Studies have been made to investigate motion sickness caused by horizontal oscillation. It has been found that the time it took to reach moderate nausea was at minimum at 0, 2 Hz and it has also been concluded that there is a peak in motion sickness response in the region of 0, 2 Hz with vertical oscillation. (Barnaby E. Donohew, 2004)

2.4. Programing language

When running a Python program on Linux some utility can be setup to help the programming. Since the MPU-6050 uses I2C bus to connect to the Raspberry Pi, with a unique address that can be set by changing jumper settings on the module. It is very useful to see the address of the devices that is connected to the Raspberry Pi as a way of making sure everything is working.

2.4.1. Linux setup

To help with setup of the IMU and programming the python-smbus utility is installed by running the following commands in the Terminal:

sudo apt-get install python-smbus

sudo apt-get install i2c-tools

Once the python-smbus and I2C-tools are installed it is easy to see if there is anything connected by running the following commands:

sudo i2cdetect -y 1

(Monk, 2013)

The Sudo command lets the user act as a root having permission to do any changes needed on the system.
To make these changes available from Idle and avoid having to run the tools from the root the user Pi is added to the I2C access group by running the command:
sudo adduser pi i2c.

2.4.2. The software

The software written for this device is divided in two python files.
More detailed information about each function can be seen in the appendix in the complete files. MPU6050.py is mostly used for collecting the data from the IMU while measurement.py includes the loops that categories the data and write data of interest to the .txt file.

The MPU6050 file is setup to make connection to the MPU-6050 via I2C at the address 0x68 and send data to register 107 to wake up the device since it will come up in sleep mode when powered up.

Ten samples measurements for each axis are made in the beginning of the program to calculate the average.
The sum of each axis sample is divided by the number of the samples to get the Ztotal[i] array. This is done every time the program starts running to define the origin that following measurements are based upon. In practice the program is to be started in the aircraft when located on the ground and at minimum slope.
After setting the zero, measurements are based upon the movement of the plain from that time on.
Drawing of the axis is on the IMU and when the unit is located in the plain directions of the axis has to be taking into account for the installation.
Preferably the x-axis is set to pointing towards the front of the plain, y-axis to the sides and –axis up and down.

When collecting the raw data the average is subtracted from it to set the origin before it is divided with the sensitivity scale factor.

$$A[i] = \frac{data\ from\ register(higaddr, lowaddr) - \text{Ztotal}[i]}{sensitivity\ factor}$$

for i in range 0-5

Data from registers 0x3b to 0x48 are collected and Ztotal[i] is subtract before dividing it with the sensitivity factor 16384 for the accelerometers and 131 for the gyro.
Since A[2] is the acceleration in z axis one is subtracted to take into account the earth's 1g force after the calculations.

When the program gets run it opens a txt file named test and writes a header including time and in which order the measurements are written.

"Time ax = ay = az = gx = gy = gz = "

The measurement.py file imports data from the file MPU6050.
When the program is run it collects measurements through the MPU6050 file.
It uses Epoch time when writing down time in the txt file and it can then be converted into year month day seconds and milliseconds later.

The program also includes a function that calculates the slope H from measurements of the six axis from point A to B at the time T by using the absolute value of array B containing measurements from the MPU-6050 at the time TB and subtracting it from absolute value of array A containing measurements from the MPU-6050 at the time TA. The outcome is then dividing it with the time between measurements A and B.

$$H[i] = \frac{abs(B[i] - A[i])}{TB - TA} \text{ for i in range 0-5}$$

The main invocation in measurement.py calls for measurements from the MPU6050 file both array A and array B and then calculates the slope.

If the maximum of the slope array multiplied by ten is less than four no data will be written to the txt file. Each time the slope array multiplied by ten is less than four the program will slow down its request for new readings by one second or until it requests new data every ten seconds. This is done to save energy, prevent the devise to heat up and slow the unit down when the airplane is on a smooth run. If the maximum of the slope array multiplied by ten is equal or more than four the measurements will be written on the txt file along with a timestamp.
Since the plain will be in the air away for hours at the time and even days, writing only down measurements of interest will prevent the .txt file to be filled with zeros and also save space and energy.

3. Test signal

To evaluate the accuracy and activity of the unit tests were made for various movements of the unit as well as a still test. The breadboard was placed on the floor and moved by hand while the program collected 350 to 1000 data readings.
The Raspberry Pi was connected to a breadboard via Pi cobbler from Adafruit connecting all 26 GPIO pins to the breadboard. Breadboard wires with stiff ends were used to connect the MPU-6050 to the cobbler.
Read wire was used for the 3,3V green for the ground, white was used for the SDA connection and orange for SCL.

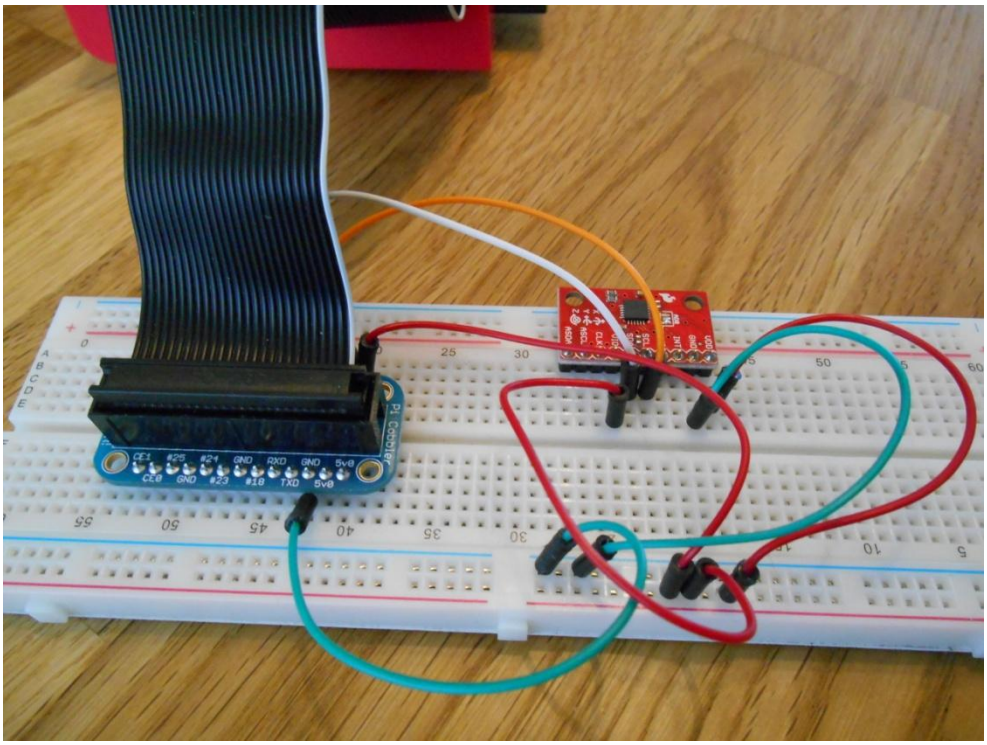

Figure 5 Test setup

3.1. Stationary test

As part of testing the movement measurement device the breadboard was placed stationary on the floor and 1000 measurements made. The measurements took just under 2 minutes, starting at 12:57:34.539 and ending at 12:59:17.288.

The data was transferred from the .txt file to Matlab and the following plots for each of the six axis where made.

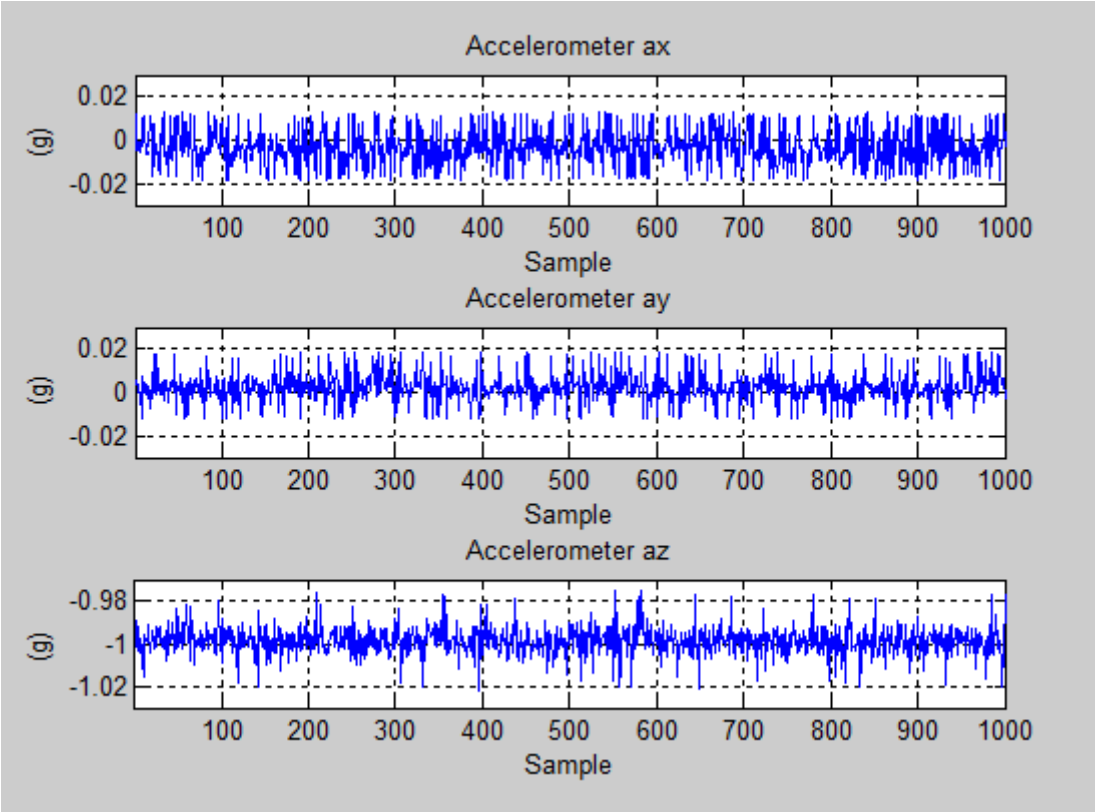

Figure 6 Accelerometer stationary

The accelerometer plots show that all accelerometers axis have noise around ±0,02g or 0,196 m/s^2. This shows that the MPU6050 has similar noise on all axes

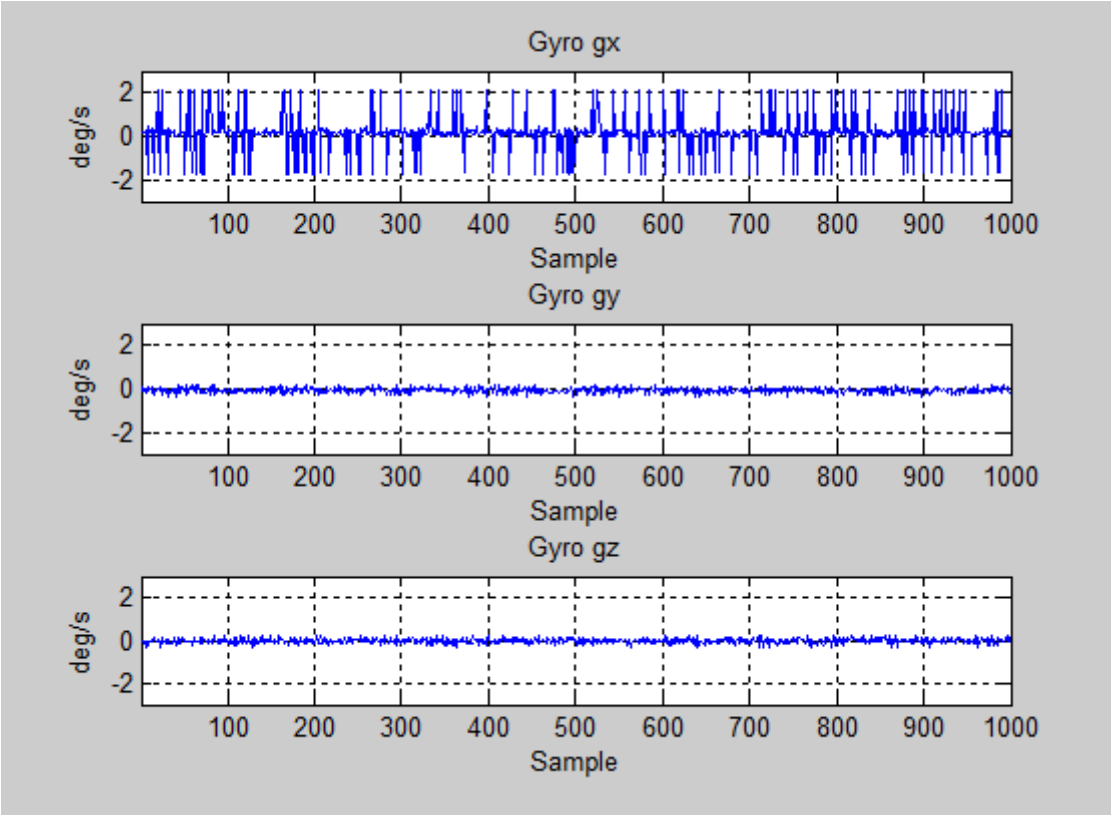

Figure 7 Gyro stationary

The gyro plots show that the gyro axis y and z have very little constant noise compare to the x- axis. The gyro x axis has noise around ± 2°/s while the y and z axis noise is around 0, 4°/s.

3.2. Tilt test

When testing tilt on the unit 1000 measurements were made.
The breadboard with the MPU6050 was placed stationary on the floor.
First it was tilt almost 90° along the x-axis and then put back as close to its original position as possible for few seconds. Then the breadboard was tilt almost 90° along the y-axis and put back as close to its original position as possible.

The movements where kept as smooth and slow as possible while the test was done. The movements where done in the first half of the testing time, the second half of the test the IMU was located motionless on the floor so if any settling time was needed it would also show on this sample test.
This test started at 20:26:45.093 and ended at 20:28:27.645 taking just under 2 minutes.

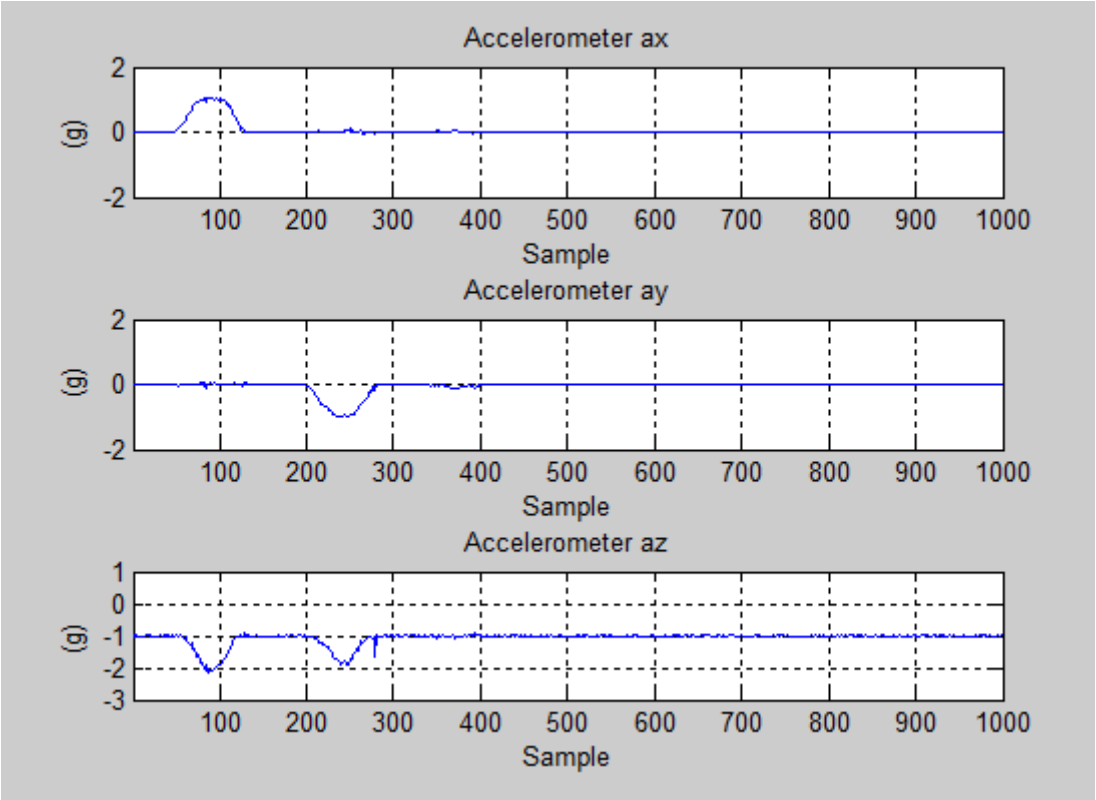

Figure 8 Accelerometer movements

The accelerometer plot shows increasing of g starting around 50 samples in x-axis and going up to 1g from sample number 80 to 100. Then the force decreases and reaches 0g again at sample number 130. At sample number 200 the movement on y axis starts and reaches -1g at sample 240 before returning to 0g at sample number 280.
The z-axis shows movements that are consequence to the movements on x- and y-axis. When leaning the MPU6050 along the x- and y-axis movements are also made on the z-axis. From sample 400 no movements are detected.
This is in consistent with the movements made on the MPU6050 for this test.

Between samples 50 and 130 a small movement is detected on the y-axis and between sample 200 and 280 a small movement is also detected on the x-axis this can be consequence of a small shaking of the IMU while moving the breadboard.

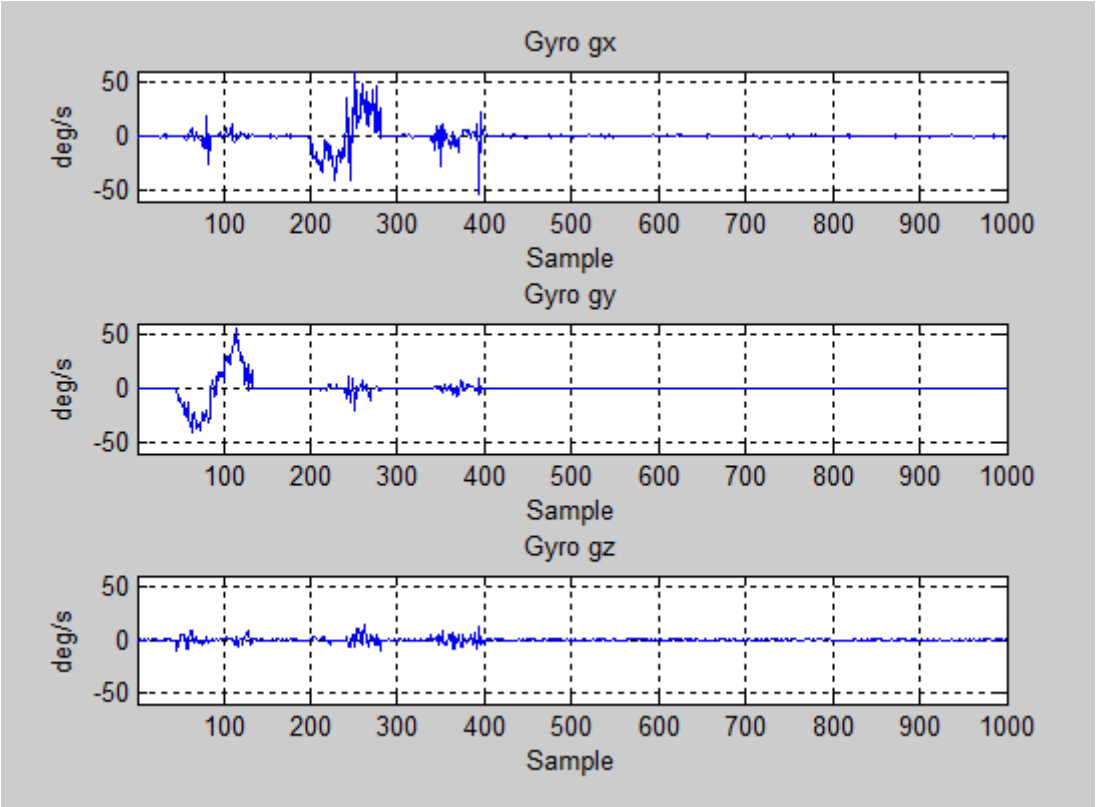

Figure 9 Gyro movements

The plot of gyro x shows the roll around x-axis. At sample 200 it starts to shows movements that consist with the tilt in gy-axis. The pitch gyro y starts at sample 50 and is consist with the tilt on ax-axis. Both the roll and the pitch go up to 50 °/s. Small movements at the yaw gyro z can be consequence of a small shaking in the test movement.

3.3. Rotation test

The device was placed on an office chair as it was rotating. First a full circle clockwise then set stationary for few seconds before rotating a full circle counterclockwise a bit faster. After a short hold the device was then rotated again in a clockwise circle.
350 measurements were collected during this test.

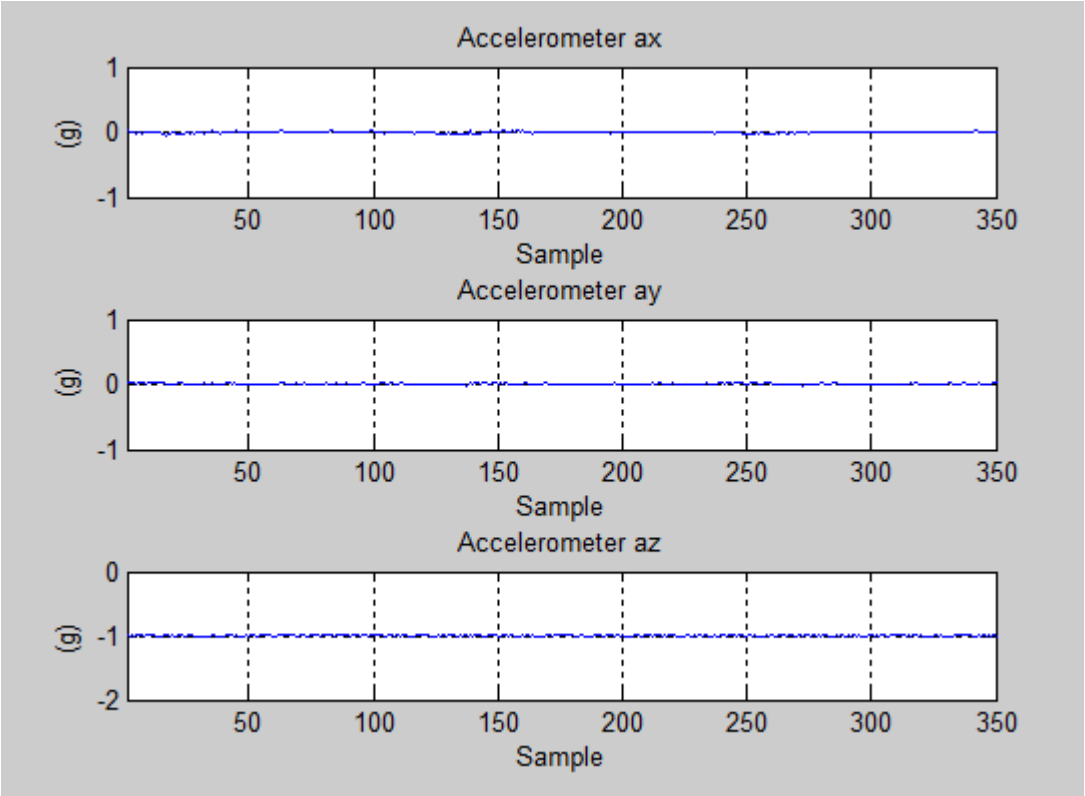

Figure 10 accelerometer rotating

Since the devise is only moving in circles around Yaw (z-axis) no change is to be expected for the accelerometer axis.

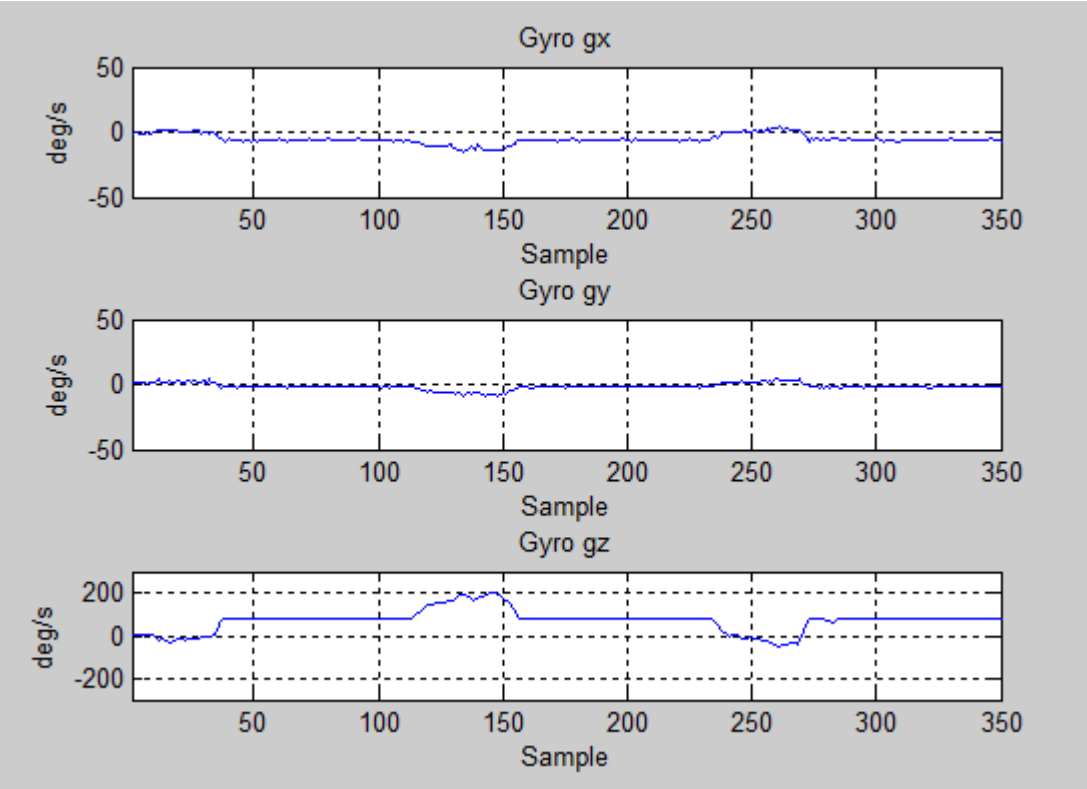

Figure 11 Rotating gyro

The chart for gyro gx and gy shows a small rotation in roll and pitch that is the consequence of unavoidable shaking of the chair holding the unit.

The chart for gyro gz shows a slow clockwise circle in the beginning then a faster movement in counterclockwise circle before showing another slower clockwise circle in the end. The line does not go back to zero in between circles because of offsett.

3.4. Suggested ways of processing the data

Further process can be done depending on what information's are needed from the measurements.
The Movement measurement device is designed to write timestamp and data in the unit g on a .txt file to be further process in Matlab or by other methods.
Matlab can be used to convert epoch milliseconds written on the txt file into date.

For example the following Matlab code converts the epoch time 1.3983e+12:

```
time = [1398257854539.00;]
time_reference = datenum('1970', 'yyyy');
 time_matlab = time_reference + time / 8.64e7;
 date = datestr(time_matlab, 'yyyy dd mm HH:MM.SS.FFF');

date  = 2014 23 04 12:57.34.539
```

Returning date in the form: year day month hours: minutes. seconds. milliseconds

The rate of change in the velocity with respect to time is acceleration.
Acceleration is therefore time derivative of velocity and velocity is the time derivative of position. It is possible to multiply the g unit with 9,8 to get m/s^2 and ten integrate the acceleration to get velocity in the unit m/s. Integrating the velocity gives position in the unit m.
If acceleration = 0 during a period of time, then the velocity is unchanging and constant over that period.

$$Position:\ p = p(t)$$

$$velocity:\ v(t) = \frac{d}{dt}p(t)$$

$$Acceleration:\ a(t) = \frac{dv}{dt} = \frac{d^2p}{dt^2}$$

The object is speeding up only when the velocity and acceleration have the same sign.

If velocity is	and acceleration is	then object is	and its speed is
positive	positive	moving right	Incresing
positive	negative	moving right	decreasing
negative	positive	moving left	decreasing
negative	negative	moving left	Incresing

Figure 12 Connection between velocity and acceleration

(Adams, 2006)

3.4.1. Calculating tilt angles

To define the angles of the acceleromerer all three axis outputs are used.

ṕ is defined as the angle of the x-axis relative to the ground. ϕ is defined as the angle of the y-axis relative to the ground. Θ is the angle of the z axis relative to the gravity.

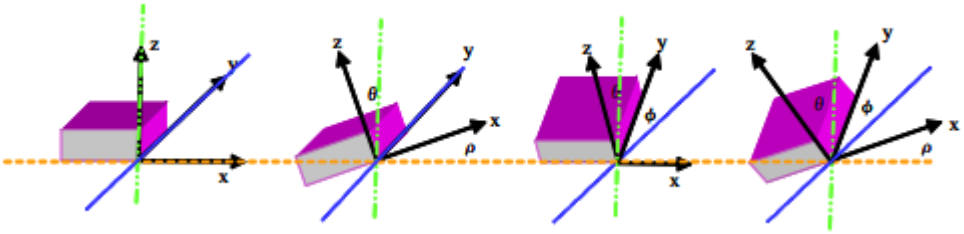

Figure 13 Tilt angles

$$\acute{\rho} = \arctan\left(\frac{ax}{\sqrt{ay^2 + az^2}}\right)$$

$$\phi = \arctan\left(\frac{ay}{\sqrt{ax^2 + az^2}}\right)$$

$$\Theta = \arctan\left(\frac{\sqrt{ax^2 + ay^2}}{az}\right)$$

(Tuck, 2007)

4. Summary

4.1. Discussion

When selecting a computer to run the unit some of the most important things to consider were the size and low power supply. The Raspberry Pi is suitable not only because of those things it is also very silent and offers various features such as wireless connection.

Accelerometers and gyro can be purchased separately and connected together later in the process. The MPU6050 includes both accelerometers and gyro making the process easier. Several combined IMU are available but the MPU6050 has an advantage as it includes I2C bus for connection and can access other sensors. It gives a digital output making calculations based on the volts input unnecessary. It also includes a low pass filter for both accelerometers and gyro. The I2C works well on Linux and the python. To save space the program Idle is used for programming since it is included in on the Raspberry Pi downloadable software NOOBS.

When testing the unit the Raspberry Pi was connected to a breadboard via Pi cobbler, the cobbler is rather short limiting the movements of the breadboard a bit.
Further tests should be done in aircraft to determine data of interest and the frequency of collected data further.

The program is intended to be used in aircrafts and is not related to earth´s directional forces except gravity. It is designed to be set of before each round trip and the data to be collected from the txt file on return.

4.2. Conclusion

The needs for a small and low power supply unit measuring both acceleration and gyro is met by combining the Raspberry Pi and MPU6050.

The program collects data from the MPU6050 as array A and array B then calculates the slope. If the slope is equal or bigger than four timestamp and data is written on the txt file that is saved on a SD card.

The program can be adapted to more detailed needs of the user, such as further separation of the data based on the slope and temperature measurements can easily be added to the txt file.

Three individual tests were made. The stationary test shows some noise around ±0,02g for the accelerometer and up to ÷0, 4°/s for the gyro.
When tilting the MPU6050 of almost 90° the accelerometers shows increasing of 1g and the peaks of the gyro go up to 50°/s.

The noise in the measurements are not major compared to the tilt test but it must be taking into account that they can have effect on later calculations and when integrating the data.

The measurements of the tests are consistent with the movements of the MPU6050 and among of the six axes.
By ignoring the low slopes the unit gives a good idea of its movements.

4.3. Future Work

The measurement unit returns data on a txt file saved on a SD card, for easier access to the data a wireless connection can be set up for the Raspberry Pi.

A user-friendly environment has to be made preferably combined with other programs used on the Raspberry Pi by the airline.

If needed a Kalmann filter could be added and used to filter the data before further calculations are made.

The program is intended to be used in aircrafts and needs to be tested in flight as sensitivity and separation of the data needs to be adapted by the average movements in the passenger compartment,

Bibliography

Adams, R. A. (2006). *Calculus a complete course.* Toronto: Pearson.

Anderson, C. (2008, July 6). *What's the difference between a gyro and an accelerometer? Do I need both?* Retrieved April 3, 2014, from Diydrones: http://www.diydrones.com/profiles/blog/show?id=705844%3ABlogPost%3A41145

Barnaby E. Donohew, M. J. (2004, August 1). *Motion Sickness: Effect of the Frequency of Lateral.* Retrieved April 3, 2014, from Ingentaconnect: http://www.ingentaconnect.com/content/asma/asem/2004/00000075/00000008/art00001

CNN/Sports Illustrated. (2001, April 30). *No need for speed*. Retrieved Mars 27, 2014, from Sportsillustrated CNN: http://sportsillustrated.cnn.com/motorsports/news/2001/04/29/cart_cancel_ap/

InvenSense Inc. (2013, August 19). *Product Specification Revision 3.4.* Retrieved Mars 3, 2014, from Invensense: http://invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf

InvenSense Inc. (2013, August 19). *Register Map and Descriptions Revision 4.2.* Retrieved Mars 3, 2014, from Invensense: http://www.invensense.com/mems/gyro/documents/RM-MPU-6000A-00v4.2.pdf

Matt Richardson, S. W. (2012). *Getting started with Raspberry Pi.* Sebastopol: O´Reilly.

Measurement Computing Corporation. (2012). *Data Acquisition Handbook Third Edition.* Retrieved Mars 25, 2014, from mccdaq: http://www.mccdaq.com/support/data-acquisition-handbook.aspx

Monk, S. (2013, April 20). *Configuring I2C*. Retrieved Februar 17, 2014, from Adafruit Learning System: http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c

Philips Semiconductors. (1995, April). *The I2C-bus and how to use it*. Retrieved February 17, 2014, from i2c-bus: http://www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf

Steve Nasiri, S.-H. L. (n.d.). *Motion Processing.* Retrieved Mars 25, 2014, from Invensense: http://www.invensense.com/mems/gyro/documents/whitepapers/InvenSense_MotionProcessing_MPUApps_WhitePaper.pdf

Tuck, K. (2007, Juní). *Tilt Sensing Using Linear Accelerometers.* Retrieved April 20, 2014, from Medical University of Hannover: https://www.mh-hannover.de/fileadmin/zentrale_einrichtungen/zentrale_forschungswerkst/downloads/AN3461.pdf

Voshell, M. (2008, November 28). *High Acceleration and the Human Body.* Retrieved Mars 27, 2014, from Cognitive Systems Engineering Laboratory The Ohio State University.: http://csel.eng.ohio-state.edu/voshell/gforce.pdf

Table of Figures

Appendix

MPU6050.py

```
# Python API for the MPU6050 IMU

# $Id$

# Written by Arny Juliusdottir <sigridur06@ru.is>

import smbus

import struct

import time


class mpu6050(object):

    A=[0]* 6

    B=[0]* 6

   Ztotal = [0]* 6

    i=0

    m=0

    T=0

    cmd = {

        "WAKEUP" : 0x6b,

        }

    defaultaddr = 0x68
```

```
    """I2C operations for the MPU 6050 IMU"""
    def __init__(self, smbid=1, address=0x68):
        """Setup the MPU 6050
        smbid identifies which I2C bus (default 1)
        address is the target address of the I2C device (default 0x68)"""
        self.bus = smbus.SMBus(smbid)
        self.address = address
        self.bus.write_byte_data(self.address,
                                 self.cmd["WAKEUP"],
                                 0)
        self.Zero()


    def Zero(self):
        def busbytetoint(highaddr, lowaddr):
            highbyte = self.bus.read_byte_data(self.address,highaddr)
            lowbyte = self.bus.read_byte_data(self.address,lowaddr)
            byteval = "%c%c" % (highbyte, lowbyte)
            return struct.unpack(">h", byteval)[0]


        def sample_i2c(highaddr, lowaddr, n=10):
            """Read in n samples from the I2C addresses given, and put into the zarray.
Default is 10 samples"""
```

```
            z=[0]* n
            for i in range(n):
                z[i]= busbytetoint(highaddr,lowaddr)
            return sum(z) / float(len(z))

        self.Ztotal[0] = sample_i2c(0x3b,0x3c) #ax
        self.Ztotal[1] = sample_i2c(0x3d,0x3e) #ay
        self.Ztotal[2] = sample_i2c(0x3f,0x40) #az
        self.Ztotal[3] = sample_i2c(0x43,0x44) #gx
        self.Ztotal[4] = sample_i2c(0x45,0x46) #gy
        self.Ztotal[5] = sample_i2c(0x47,0x48) #gz

# Called to print information on screen
    def printinfo(self, t):
        print('A=',self.A)
        print('t=',t)
        return

 # Save new data in array A
    def getdataA(self):
        def busbytetoint(highaddr, lowaddr):
            highbyte = self.bus.read_byte_data(self.address,highaddr)
            lowbyte = self.bus.read_byte_data(self.address,lowaddr)
```

```
            byteval = "%c%c" % (highbyte, lowbyte)
            return struct.unpack(">h", byteval)[0]
        Ztotal = self.Ztotal
        A = self.A
            # collected data minus – Zero / sensitivity scale factor
        A[0]= (busbytetoint(0x3b,0x3c)-Ztotal[0])/16384.0
        A[1]= (busbytetoint(0x3d,0x3e)-Ztotal[1])/16384.0
        A[2]= ((busbytetoint(0x3f,0x40)-Ztotal[2])/16384.0)-1
        A[3]= (busbytetoint(0x43,0x44)-Ztotal[3])/131.0
        A[4]= (busbytetoint(0x45,0x46)-Ztotal[4])/131.0
        A[5]= (busbytetoint(0x47,0x48)-Ztotal[5])/131.0

# Save new data in array B
    def getdataB(self):
      def busbytetoint(highaddr, lowaddr):
          highbyte = self.bus.read_byte_data(self.address,highaddr)
          lowbyte = self.bus.read_byte_data(self.address,lowaddr)
          byteval = "%c%c" % (highbyte, lowbyte)
          return struct.unpack(">h", byteval)[0]

    Ztotal = self.Ztotal
        B = self.B
          # collected data minus – Zero / sensitivity scale factor
```

```
        B[0]= busbytetoint(0x3b,0x3c)/16384.0
        B[1]= busbytetoint(0x3d,0x3e)/16384.0
        B[2]= (busbytetoint(0x3f,0x40)/16384.0)-1
        B[3]= busbytetoint(0x43,0x44)/131.0
        B[4]= busbytetoint(0x45,0x46)/131.0
        B[5]= busbytetoint(0x47,0x48)/131.0
        # main invocation
# Only gets run if this is run as the program
# Otherwise only setup the object as above
if __name__ == "__main__":
    import time
    U = mpu6050()
# opens test.txt and writes header
    f= open("test.txt","a")
    f.write(time.strftime("%d:%B:%Y"))
    f.write("\n\n")
    f.write (" time        ax=      ay=      az=      gx=      gy=      gz=        " )
    f.write("\n")
    f.close()
```

Measurement.py

```
 # -*- coding: utf-8 -*-
# Python API for the MPU6050 IMU
# $Id$
# Written by Arny Juliusdottir <sigridur06@ru.is>
import mpu6050
import datetime
import time
H=[0]* 6
i = 0
m = 0
def TimestampMillisec64():
    return int((datetime.datetime.utcnow() - datetime.datetime(1970, 1, 1)).total_seconds() * 1000)


class measurements(object):
    def __init__(self):
        self.mpu = mpu6050.mpu6050()
        self.f = open("test.txt","a")
    def getdataA(self):
        self.dataA = self.mpu.getdataA()
    def getdataB(self):
```

```
        self.dataB = self.mpu.getdataB()
# Write array A to the file test.txt
    def writedata(self):
         t = TimestampMillisec64()
        f = self.f
        f=open("test.txt","a")
         f.write("%d" %t)
        A = self.mpu.A
        f.write (" %04.6f %04.6f %04.6f %04.6f %04.6f %04.6f" \
                %(A[0],A[1],A[2],A[3]\
                    ,A[4],A[5]))
        f.write("\n")
        return

# calculating the slope between array A and B
    def slope(self):
        T= tB-tA
        H[0]= abs((self.mpu.B[0]-self.mpu.A[0])/T)
        H[1]= abs((self.mpu.B[1]-self.mpu.A[1])/T)
        H[2]= abs((self.mpu.B[2]-self.mpu.A[2])/T)
        H[3]= abs((self.mpu.B[3]-self.mpu.A[3])/T)
        H[4]= abs((self.mpu.B[4]-self.mpu.A[4])/T)
        H[5]= abs((self.mpu.B[5]-self.mpu.A[5])/T)
```

```
        return[H]


# Print data on screen, used while testing
   # def screen(self):
       # print('H=',H)
      #  print('A=',self.mpu.A)
      #  print('B=',self.mpu.B)
      #return


# main invocation
# Only gets run if this is run as the program
# Otherwise only setup the object as above
if __name__ == "__main__":
    M = measurements()
    # Calls for new data for array A and B
    M.getdataA();
    tA= float(round(time.time()*1000))
    M.getdataB();
    tB= float(round(time.time()*1000))
    M.slope();
    #M.screen()
```

```
    while max(H)>0:
        M.getdataA();
        tA= float(round(time.time()*1000))
        M.getdataB();
        tB= float(round(time.time()*1000))
        M.slope();
        #M.screen()
            # if max slope is less than 4 no information is written on the txt.file
        if (max(H)*10 < 4):
            time.sleep(m)
            m=m+1
            if m==10:
                m=m-1
       # if max slope is more than 4 then information is printed on txt file
        else:
           # M.screen();
            M.writedata();
            m=0

else:
    print ('no data from accelgyro')
```