Change Log

Monday, January 20, 2014

6:28 PM

|  |  |
| --- | --- |
| 10-27-2013 | 1. Modified incorporate temperature correction for gyro and accelerometer  2. Corrected \_\_AVR\_\_ to \_\_AVR\_ATmega128\_\_  3. Added call to get temperature from MPU-6050 in getRawValues |
| 10-29-2013 | Corrected temp coefficient array for temp |
| 11-01-2013 | Added new inverse sqrt code obtained from articles:  <http://pizer.wordpress.com/2008/10/12/fast-inverse-square-root/> and  <http://www.diydrones.com/forum/topics/madgwick-imu-ahrs-and-fast-inverse-square-root> |
| 11-05-13 | 1. Add code to reset MPU-6050 on start-up, implemented in the Arduino sketch as well as option to manually reset in processing sketch.  2. Added LPF code line, currently set to MPU60X0\_DLPF\_BW\_42 |
| 11-23-13 | 1. Changed twoKidef from 0.1 to 0.0f -12-12 => twoKiDef changed from 0f to 0.0025f based on article in DIYDrones  2. Added bypass caps (0.1 and 0.01 ceramic caps on the power pins of the freeIMU  2a. Added a 10 uF tantalum on the power bus  3. Yaw drift stabilized very nicely  4. Updated processing sketch to calculate heading and display based on the freescale application note and as implemented:  <http://diydrones.com/profiles/blogs/tiltcompensated-heading>  5. update RESET function to zero out initial values |
| 12-06-13 | 1. added LSM303D Polulo board instead of calculating heading from FreeIMU  2. added option to turn on/off temp correction, temp\_corr\_on  3. added variable to set temperature break where calibration turns off, temp\_break, note: temp\_break in units directly from MPU-6050 |
| 01-02-14 to  01-05-14 | 1. Recalibrated 6050 using Madgewick automated calibration rig - see paper  2. Corrected temperature calibration coefficients to ensure that gyros are zeroed for whole temp range when at rest - deleted necessary to run zerogyro routine.  3. Acceleration coefficients adjusted so Ax = 0, Ay = 0 and Az = 1g when at rest  4. Had to tune Ki to reduce drift - now set to 0.165  5. Changed Kp to 1.0 - drift almost zero |
| 01-08-14 | 1. Completed coding ZARA ZUPT for Zero velocity detemination - added code to the processing sketch |
| 01-09-14 | 1. Added a Reset Quaternion matrix only - option 2 in the sketch - to allow reset of matrix after rapid movement. |
|  |  |
|  |  |

Dec 29, 2013

Completed assembly of the rig shown in the attached pdf by Seb Madgewick. Also, found a Arudino library (PMCtrl) which I tested last night but did not fully work reliably for some reason. After a little googling I found a python wrapper on the Pololu forum (<http://forum.pololu.com/viewtopic.php?f=16&t=5591>). Using the wrapper I added the code to auto rotate the rig like I would do if I did it by hand.

<<MaestroTest.py>>

Made a few modifications to the setup for my use over that from the attached paper.

Madgewick Paper and Software:

|  |  |
| --- | --- |
| <<automated\_calibration\_feasibility\_study.pdf>> | <<calibration\_feasibility\_study.zip>> |

Maestro Controler + Original PMCtrl Controller - see separate tabs

I also posted a YouTube video of the setup in operation:

<http://youtu.be/SNM6lB440Z8>

#!/usr/bin/python

###########################################################################################

# Filename:

# Device.py

# http://forum.pololu.com/viewtopic.php?f=16&t=5591

# Here is the code I'm running on both machines: (I'm adjusting for the

# different COM ports, and USB Dual Port mode is enabled)

###########################################################################################

# Project Authors:

# Juhapekka Piiroinen

# Brian Wu

#

# Changes:

# June 14, 2010 by Juhapekka Piiroinen - changes committed to svn

# - added comments for the device commands according to the manual from Pololu

# - added latest draft code for rotating base servo (Parallax Continuous Rotating Servo)

# - note! you should be able to clear error flags with .get\_errors function according to the manual

# - renamed CameraDriver to LegacyCameraDriver as Brian Wu has done better one

# - integrated batch of changes provided by Brian Wu

#

# June 11, 2010 by Brian Wu - Changes committed thru email

# - Decoupling the implementation from the program

#

# April 19, 2010 by Juhapekka Piiroinen

# - Initial Release

#

# Email:

# juhapekka.piiroinen@gmail.com

#

# License:

# GNU/GPLv3

#

# Description:

# A python-wrapper for Pololu Micro Maestro 6-Channel USB Servo Controller

#

############################################################################################

# /!\ Notes /!\

# You will have to enable \_USB Dual Port\_ mode from the \_Pololu Maestro Control Center\_.

#

############################################################################################

# Device Documentation is available @ http://www.pololu.com/docs/pdf/0J40/maestro.pdf

############################################################################################

# (C) 2010 Juhapekka Piiroinen

# Brian Wu

############################################################################################

import serial

import time

def log(\*msgline):

for msg in msgline:

print msg,

print

class Device(object):

def \_\_init\_\_(self,con\_port="COM12",ser\_port="COM13",timeout=1): #/dev/ttyACM0 and /dev/ttyACM1 for Linux

############################

# lets introduce and init the main variables

self.con = None

self.ser = None

self.isInitialized = False

############################

# lets connect the TTL Port

try:

self.con = serial.Serial(con\_port,timeout=timeout)

self.con.baudrate = 9600

self.con.close()

self.con.open()

log("Link to Command Port -", con\_port, "- successful")

except serial.serialutil.SerialException, e:

print e

log("Link to Command Port -", con\_port, "- failed")

if self.con:

#####################

#If your Maestro's serial mode is "UART, detect baud rate", you must first send it the baud rate indication byte 0xAA on

#the RX line before sending any commands. The 0xAA baud rate indication byte can be the first byte of a Pololu protocol

#command.

#http://www.pololu.com/docs/pdf/0J40/maestro.pdf - page 35

self.con.write(chr(0xAA))

self.con.flush()

log("Baud rate indication byte 0xAA sent!")

###################################

# lets connect the TTL Port

try:

self.ser = serial.Serial(ser\_port,timeout=timeout)

self.ser.close()

self.ser.open()

log("Link to TTL Port -", ser\_port, "- successful")

except serial.serialutil.SerialException, e:

print e

log("Link to TTL Port -", ser\_port, "- failed!")

self.isInitialized = (self.con!=None and self.ser!=None)

if (self.isInitialized):

err\_flags = self.get\_errors()

log("Device error flags read (",err\_flags,") and cleared")

log("Device initialized:",self.isInitialized)

###########################################################################################################################

## common write function for handling all write related tasks

def write(self,\*data):

if not self.isInitialized: log("Not initialized"); return

if not self.ser.writable():

log("Device not writable")

return

for d in data:

self.ser.write(chr(d))

self.ser.flush()

###########################################################################################################################

## Go Home

# Compact protocol: 0xA2

# --

# This command sends all servos and outputs to their home positions, just as if an error had occurred. For servos and

# outputs set to "Ignore", the position will be unchanged.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def go\_home(self):

if not self.isInitialized: log("Not initialized"); return

self.write(0xA2)

###########################################################################################################################

## Set Target

# Compact protocol: 0x84, channel number, target low bits, target high bits

# --

# The lower 7 bits of the third data byte represent bits 0-6 of the target (the lower 7 bits), while the lower 7 bits of the

# fourth data byte represent bits 7-13 of the target. The target is a non-negative integer.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def set\_target(self,servo,value):

if not self.isInitialized: log("Not initialized"); return

highbits,lowbits = divmod(value,32)

self.write(0x84,servo,lowbits << 2,highbits)

###########################################################################################################################

## Set Speed

# Compact protocol: 0x87, channel number, speed low bits, speed high bits

# --

# This command limits the speed at which a servo channel's output value changes. The speed limit is given in units of (0.25 us)/(10 ms)

# --

# For example, the command 0x87, 0x05, 0x0C, 0x01 sets

# the speed of servo channel 5 to a value of 140, which corresponds to a speed of 3.5 us/ms. What this means is that if

# you send a Set Target command to adjust the target from, say, 1000 us to 1350 us, it will take 100 ms to make that

# adjustment. A speed of 0 makes the speed unlimited, so that setting the target will immediately affect the position. Note

# that the actual speed at which your servo moves is also limited by the design of the servo itself, the supply voltage, and

# mechanical loads; this parameter will not help your servo go faster than what it is physically capable of.

# --

# At the minimum speed setting of 1, the servo output takes 40 seconds to move from 1 to 2 ms.

# The speed setting has no effect on channels configured as inputs or digital outputs.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def set\_speed(self,servo,speed):

if not self.isInitialized: log("Not initialized"); return

highbits,lowbits = divmod(speed,32)

#self.write(0x87,servo,lowbits << 2,highbits)

self.write(0x87,servo,lowbits,highbits)

###########################################################################################################################

## Set Acceleration

# Compact protocol: 0x89, channel number, acceleration low bits, acceleration high bits

# --

# This command limits the acceleration of a servo channel's output. The acceleration limit is a value from 0 to 255 in units of (0.25 us)/(10 ms)/(80 ms),

# --

# A value of 0 corresponds to no acceleration limit. An acceleration limit causes the speed of a servo to slowly ramp up until it reaches the maximum speed, then

# to ramp down again as position approaches target, resulting in a relatively smooth motion from one point to another.

# With acceleration and speed limits, only a few target settings are required to make natural-looking motions that would

# otherwise be quite complicated to produce.

# --

# At the minimum acceleration setting of 1, the servo output takes about 3 seconds to move smoothly from a target of 1 ms to a target of 2 ms.

# The acceleration setting has no effect on channels configured as inputs or digital outputs.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def set\_acceleration(self,servo,acceleration):

if not self.isInitialized: log("Not initialized"); return

highbits,lowbits = divmod(acceleration,32)

//self.write(0x89,servo,lowbits << 2,highbits)

self.write(0x89,servo,lowbits,highbits)

###########################################################################################################################

## Get Position

# Compact protocol: 0x90, channel number

# Response: position low 8 bits, position high 8 bits

# --

# This command allows the device communicating with the Maestro to get the position value of a channel. The position

# is sent as a two-byte response immediately after the command is received.

# --

# If the specified channel is configured as a servo, this position value represents the current pulse width that the Maestro

# is transmitting on the channel, reflecting the effects of any previous commands, speed and acceleration limits, or scripts

# running on the Maestro.

# --

# If the channel is configured as a digital output, a position value less than 6000 means the Maestro is driving the line low,

# while a position value of 6000 or greater means the Maestro is driving the line high.

# --

# If the channel is configured as an input, the position represents the voltage measured on the channel. The inputs on

# channels 0-11 are analog: their values range from 0 to 1023, representing voltages from 0 to 5 V. The inputs on channels

# 12-23 are digital: their values are either exactly 0 or exactly 1023.

# --

# Note that the formatting of the position in this command differs from the target/speed/acceleration formatting in the

# other commands. Since there is no restriction on the high bit, the position is formatted as a standard little-endian two-

# byte unsigned integer. For example, a position of 2567 corresponds to a response 0x07, 0x0A.

# --

# Note that the position value returned by this command is equal to four times the number displayed in the Position box

# in the Status tab of the Maestro Control Center.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def get\_position(self,servo):

if not self.isInitialized: log("Not initialized"); return None

self.write(0x90,servo)

data = self.ser.read(2)

if data:

return (ord(data[0])+(ord(data[1])<<8))/4

else:

return None

###########################################################################################################################

## Get Moving State

# Compact protocol: 0x93

# Response: 0x00 if no servos are moving, 0x01 if servos are moving

# --

# This command is used to determine whether the servo outputs have reached their targets or are still changing, limited

# by speed or acceleration settings. Using this command together with the Set Target command, you can initiate several

# servo movements and wait for all the movements to finish before moving on to the next step of your program.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def get\_moving\_state(self):

if not self.isInitialized: log("Not initialized"); return None

self.write(0x93)

data = self.ser.read(1)

if data:

return ord(data[0])

else:

return None

###########################################################################################################################

## Get Errors

# Compact protocol: 0xA1

# --

# Response: error bits 0-7, error bits 8-15

# --

# Use this command to examine the errors that the Maestro has detected.

# --

# The error register is sent as a two-byte response immediately after the command is received,

# then all the error bits are cleared. For most applications using serial control, it is a good idea to check errors continuously

# and take appropriate action if errors occur.

# --

# Source: http://www.pololu.com/docs/pdf/0J40/maestro.pdf

def get\_errors(self):

if not self.isInitialized: log("Not initialized"); return None

self.write(0xA1)

data = self.ser.read(2)

if data:

return ord(data[0])+(ord(data[1])<<8)

else:

return None

###########################################################################################################################

## a helper function for Set Target

def wait\_until\_at\_target(self):

while (self.get\_moving\_state()):

time.sleep(0.1)

###########################################################################################################################

## Lets close and clean when we are done

def \_\_del\_\_(self):

if (self.ser):

self.ser.close()

if (self.con):

self.con.close()

del(self.ser)

del(self.con)

####################################################################

from array import \*

pitch = array("I", [1500, 960, 2064, 1500])

roll = array("I", [1508, 1393, 1620, 1508])

yaw = array("I", [1500, 1378, 1618, 1734, 1500])

servo = Device("COM12","COM13")

servo.set\_target(0,1500)

time.sleep(2)

servo.set\_target(1,1508)

time.sleep(2)

servo.set\_target(2,1500)

time.sleep(10)

for y in yaw:

servo.set\_speed(0, 1)

servo.set\_target(0, y)

servo.wait\_until\_at\_target()

time.sleep(2)

for r in roll:

servo.set\_speed(1, 1)

servo.set\_target(1, r)

servo.wait\_until\_at\_target()

time.sleep(2)

for p in pitch:

servo.set\_speed(2, 4)

servo.set\_target(2, p)

servo.wait\_until\_at\_target()

time.sleep(2)

servo.set\_target(2,1508)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1280)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1728)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_target(1,1280)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1728)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1508)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1728)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(2, 5)

servo.set\_target(2,960)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(2, 5)

servo.set\_target(2,2064)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(2, 5)

servo.set\_target(2,960)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(2, 5)

servo.set\_target(2,1500)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1500)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(2, 5)

servo.set\_target(2,2064)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1280)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 5)

servo.set\_target(1,1728)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1280)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_speed(1, 1)

servo.set\_target(1,1728)

servo.wait\_until\_at\_target()

time.sleep(3)

servo.set\_target(0,1500)

time.sleep(2)

servo.set\_target(1,1508)

time.sleep(2)

servo.set\_target(2,1500)

Implementation of temperature compensation

Sunday, January 19, 2014

10:55 AM

1/19/14

Temperature compensation seems to be built into every AHRS that I have looked at todate (Multwii, Ardupilot, IMU Brick, etc.) and was on Fabio Verasano's todo list for the ITG3200 (<https://code.google.com/p/itg-3200driver/issues/detail?id=9>). The code implemented in the reference apps all do a calibration when the unit is turned, however, the approach I took was slightly different but in the same vane. I did the calibration once over a wide temperature range, determined a break point where the accel & gyro values leveled out and then generated a calibration curve fit which I embedded into the FreeIMU library.

Also see the following references for further information on temperature compensation:

**Temperature compensation It does matter -** <http://slugsuav.soe.ucsc.edu/fllog/files/temperature_compensation.html>

**Application Note AN598 - Temperature Compensation Techniques for Low g iMEMS® Accelerometers** <http://www.analog.com/static/imported-files/application_notes/826890745AN598.pdf>

|  |  |  |  |
| --- | --- | --- | --- |
| <<Tempdata01-02-14.csv>> | <<TempCalibR2.txt>> | <<New Calib Corr 1-3-14>> | <<temp comp1.txt>> |

Changes to FreeIMU Code follows:

In the header added:

float c3[9] = { 0., 0., -1.618180e-09, 0., 0., 0., 0., 0., 0.};

float c2[9] = {4.798083e-07 ,-7.104300e-08 , -1.899410e-05, -4.387634e-08, -1.779335e-08, 4.216745e-09, 0., 0., 0. };

float c1[9] = {1.801522e-02 ,-5.200081e-03 , -1.462879e-01, -5.878346e-04, 1.172002e-03, -6.897733e-05, 0., 0., 0. };

float c0[9] = { -45.61 ,         -45.18, -305.58, 6.699801e+00, 8.341212e+00,        -2.171155e+01, 0., 0., 0. };

int nsamples = 75;

//original temp\_break = -4300;

int temp\_break = -1000;

int temp\_corr\_on = 1;

Updated the following function call:

/\*\*

\* Populates values with calibrated readings from the sensors

\*/

void FreeIMU::getValues(float \* values) {

float acgyro\_corr[9] = {0.,0.,0.,0.,0.,0.,0.,0.,0.};

int16\_t DTemp;

if(temp\_corr\_on == 1) {

DTemp = accgyro.getTemperature();

if(DTemp < temp\_break){

for(int i = 0; i < 9; i++) {

acgyro\_corr[i] = c3[i]\*(DTemp\*DTemp\*DTemp) + c2[i]\*(DTemp\*DTemp) + c1[i]\*DTemp + c0[i];

}

}

else {

float acgyro\_corr[9] = {0.,0.,0.,0.,0.,0.,0.,0.,0.};

}

}

#if HAS\_ITG3200()

int accval[3];

acc.readAccel(&accval[0], &accval[1], &accval[2]);

values[0] = (float) accval[0] - acgyro\_corr[0];

values[1] = (float) accval[1] - acgyro\_corr[1];

values[2] = (float) accval[2] - acgyro\_corr[2];

gyro.readGyro(&values[3]);

values[3] = values[3] - acgyro\_corr[3];

values[4] = values[4] - acgyro\_corr[4];

values[5] = values[5] - acgyro\_corr[5];

#else // MPU6050

int16\_t accgyroval[6];

accgyro.getMotion6(&accgyroval[0], &accgyroval[1], &accgyroval[2], &accgyroval[3], &accgyroval[4], &accgyroval[5]);

// remove offsets from the gyroscope

if(temp\_corr\_on == 1){

accgyroval[3] = accgyroval[3] - acgyro\_corr[3];

accgyroval[4] = accgyroval[4] - acgyro\_corr[4];

accgyroval[5] = accgyroval[5] - acgyro\_corr[5];

}

else {

accgyroval[3] = accgyroval[3] - gyro\_off\_x;

accgyroval[4] = accgyroval[4] - gyro\_off\_y;

accgyroval[5] = accgyroval[5] - gyro\_off\_z;

}

for(int i = 0; i<6; i++) {

if(i < 3) {

values[i] = (float) accgyroval[i] - acgyro\_corr[i];

}

else {

//values[i] = ((float) accgyroval[i] - acgyro\_corr[i])/ 16.4f; // NOTE: this depends on the sensitivity chosen

values[i] = ((float) accgyroval[i] )/ 16.4f; // NOTE: this depends on the sensitivity chosen

}

}

#endif

#warning Accelerometer calibration active: have you calibrated your device?

// remove offsets and scale accelerometer (calibration)

values[0] = (values[0] - acc\_off\_x) / acc\_scale\_x;

values[1] = (values[1] - acc\_off\_y) / acc\_scale\_y;

values[2] = (values[2] - acc\_off\_z) / acc\_scale\_z;

#if HAS\_HMC5883L()

magn.getValues(&values[6]);

// calibration

if(DTemp < temp\_break) {

values[6] = (values[6] - acgyro\_corr[6] - magn\_off\_x) / magn\_scale\_x;

values[7] = (values[7] - acgyro\_corr[7] - magn\_off\_y) / magn\_scale\_y;

values[8] = (values[8] - acgyro\_corr[8] - magn\_off\_z) / magn\_scale\_z;

}

else {

#warning Magnetometer calibration active: have you calibrated your device?

values[6] = (values[6] - magn\_off\_x) / magn\_scale\_x;

values[7] = (values[7] - magn\_off\_y) / magn\_scale\_y;

values[8] = (values[8] - magn\_off\_z) / magn\_scale\_z;

}

#endif

}

MotionDetect() Implementation

Sunday, January 19, 2014

10:46 AM

1/19/14

Using the methodology in Personal Dead Reckoning System with Shoe Mounted Inertial Sensors, SUJATHA RAJAGOPAL, Master’s Degree Project, Stockholm, Sweden 2008, XR-EE-SB 2008:013 for ZARA/ZUPT the following MotionDetect was developed (Note - you will have to adjust the thresholds based on your application and noise in the system).

To draw the box with text above in processing I added the following code to the main sketch tab:

text(MotionDetect(),VIEW\_SIZE\_X-125,VIEW\_SIZE\_Y-125) ;

if(MotionDetect() > 0 ){

fill(#FF0000);

} else {

fill(#FFFFFF)

; }

rect(VIEW\_SIZE\_X-100,VIEW\_SIZE\_Y-100,50,50);

======================================================

public float MotionDetect() {

/\*###################################################################

acceleration squared euclidean norm analysis

################################################################### \*/

accnorm = (acc[0]\*acc[0]+acc[1]\*acc[1]+acc[2]\*acc[2]);

//if((accnorm >=0.96) && (accnorm <= 0.99)){

if((accnorm >=0.96) && (accnorm <= 0.995)){

accnorm\_test = 0;

} else {

accnorm\_test = 1; }

//take average of 5 to 10 points

accnorm\_avg.newNum(accnorm);

accnorm\_test\_avg.newNum(accnorm\_test);

/\* ####################################################################

#

# squared norm analysis to determine suddenly changes signal

#

##################################################################### \*/

accnorm\_var.newNum(sq(accnorm-accnorm\_avg.getAvg()));

if(accnorm\_var.getAvg() < 0.00025) {

accnorm\_var\_test = 0;

}else {

accnorm\_var\_test = 1; }

/\*####################################################################

#

# angular rate analysis in order to disregard linear acceleration

#

################################################################### \*/

if ((gyro[0] >=-0.0024) && (gyro[0] <= 0.0)) {

omegax = 0;

} else {

omegax = 1; }

if((gyro[1] >= -0.0012) && (gyro[1] <= 0.00215)) {

omegay = 0;

} else {

omegay = 1; }

if((gyro[2] >= -0.0012) && (gyro[2] <= 0.00215)) {

omegaz = 0;

} else {

omegaz = 1; }

if ((omegax+omegay+omegaz) > 0) {

omega\_test = 1;

} else {

omega\_test = 0; }

/\* ####################################################################

#

# combined movement detector

#

#################################################################### \*/

if ((accnorm\_test\_avg.getAvg() + omega\_test + accnorm\_var\_test) > 0) {

motionDetect = 1;

} else {

motionDetect = 0; }

//####################################################################

motion\_detect\_ma.newNum(motionDetect);

if(motion\_detect\_ma.getAvg() > 0.5) {

return 1;

} else {

return 0; }

//return omegaz;

}

Altitude-Altimeter Complementary Filter

Sunday, January 19, 2014

8:32 AM

1/19/14

Finished implementing Fabio's complimentary filter that was commented out in the FreeIMU library. The filter was extracted from the Multwii IMU.cpp code. Fabio used the same code in his PALLA software written in Python.

|  |  |  |
| --- | --- | --- |
| <<MultiWii\_2\_3.zip>> | <<MultiWii\_dev\_2014\_01\_14\_\_r1648.zip>> | <<PALLA\_software.zip>> |

I deleted the filter from the FreeIMU library and incorporated it into the Processing GUI for now as it was easier to test and reduces the size of the Arduino sketch and reduces the load as I am also planning to add addition code for GPS and RC commands. To fully implement you also had to translate his quaternion function library which I found difficult. However, instead of using his library I found a quaternion library already written for Processing:

**Proscene** (pronounced similar as the Czech word **"prosím"** which means **"please"**) is a java library package which provides classes to ease the creation of interactive 3D scenes in [Processing](http://processing.org/).

**Proscene** has been pretty much inspired in the [Qt's](http://qtsoftware.com/) [OpenGL](http://www.opengl.org/) [C++](http://en.wikipedia.org/wiki/C++) [libqglviewer](http://www.libqglviewer.com/) library from where it borrows the concept of an **interactive frame**, i.e., a coordinate system that can be controlled with the mouse. **Proscene** aims at broadening this idea by allowing the user to easily setup an [HID controlled scene](https://code.google.com/p/proscene/wiki/HIDevice). **Proscene** has a very similar functionality and API reference to that found in libqglviewer.

**Proscene** provides seemless integration with **Processing**: its API has been designed to fit that of **Processing** and its implementation has been optimized to work along side with it. One of **Proscene** main implementation goals was to keep it independent of the underlying **Processing** 3D renderer. It has been **tested** with the OPENGL, P3D and GLGRAPHICS renderers and can properly work with any of them.

Pasted from <<https://code.google.com/p/proscene/>>

<<proscene-1.2.0.zip>>

I did not use the Proscene library in its entirety but extracted the Quaternion and math packages and inserted into the processing sketch. I also did notice that there was an error in the filter when comparing to the Multiwii code which fixed another issue. While the altitude stabilized it was still a little bit too jittery for me so I added the 1d kalman filter on the pressure prior to calculating the altitude and seemed to do the trick.

See <http://forum.arduino.cc/index.php?topic=72442.0> for the kalman filter that was incorporated.

======================================

Other available libraries:

1. <http://introcs.cs.princeton.edu/java/32class/Quaternion.java.html>
2. <http://willperone.net/Code/quaternion.php> (here's the entire math library -

<<willperone-master.zip>>



======================================

Following is the Processing Class:

// complementary filter from Palla Software

//

final float Kp1 = 0.55f; // PI observer velocity gain

final float Kp2 = 1.0f; // PI observer position gain

final float Ki = 0.001f; // PI observer integral gain (bias cancellation)

public class AltitudeComplementary {

//self.inited = 0

public float AltErrorI = 0;

//self.AccScale = 0.0

//public int startTime = millis();

public float AltError = 0;

public float InstAcc = 0.0;

public float Delta = 0;

public float EstVelocity = 0.0;

public float EstAlt = altitude;

//public float lastTime = millis();

// Computes the complementary estimation of Altitude based from the Barometer and Accelerometer

// @param dyn\_acc\_z the Z componente of the dynamic acceleration expressed in the Earth frame

// @BaroAlt the Baromether altitude reading as coming from the sensor expressed in meters with decimals

public float update(float dyn\_acc\_z, float BaroAlt, float dt ){

//int currentTime = millis();

//println(EstAlt);

//println(dyn\_acc\_z);

// Estimation Error

AltError = BaroAlt - EstAlt;

AltErrorI += AltError;

AltErrorI = constrain(AltErrorI,-150.00,+150.00);

InstAcc = dyn\_acc\_z \* 9.80665; //+ ((self.AltErrorI / 1000) if self.AltErrorI != 0.0 else 0)

if (AltErrorI != 0.0){

//self.InstAcc += self.AltErrorI / 10000.0

InstAcc = AltErrorI \* Ki;

}

if(Float.isNaN(InstAcc)){

InstAcc = 0;

}

float dt1 = dt/1000.;

// Integrators

Delta = InstAcc \* dt1 + (Kp1 \* dt1) \* AltError;

if (EstVelocity != 0.0) {

EstAlt += (EstVelocity/5.0 + Delta) \* (dt1 / 2) + (Kp2 \* dt1) \* AltError;

} else {

EstAlt = Delta \* (dt1 / 2) + (Kp2 \* dt1) \* AltError;

}

EstVelocity += Delta \* 10.0;

//print (self.InstAcc, self.AltErrorI, dt, self.Delta)

//lastTime = currentTime;

return EstAlt;

}

}

In the main:

Heading:

float[] dyn\_acc = new float[3];

float fused\_alt;

Quaternion dyn\_acc\_q;

Quaternion q1;

Quaternion multQ;

Quaternion dyn\_acc\_q\_earth;

Quaternion conQ;

AltitudeComplementary altitudeFilter = new AltitudeComplementary();

Draw:

//Kalman filter for the pressure

float press1 = pressK.update(press);

altitude = ((pow((sea\_press / press1), 1/5.257) - 1.0) \* (temp + 273.15)) / 0.0065;

altitude = altitude + 36/METERS2FT;

EstimatedAltitude();

Function:

void EstimatedAltitude() {

gravityCompensateDynAcc();

dyn\_acc\_q.x = dyn\_acc[0];

dyn\_acc\_q.y = dyn\_acc[1];

dyn\_acc\_q.z = dyn\_acc[2];

dyn\_acc\_q.w = 0;

q1.x = q[1]; q1.y = q[2]; q1.z = q[3]; q1.w = q[0];

multQ = Quaternion.multiply(q1, dyn\_acc\_q);

//conQ = Quaternion.conjugate(q1);

conQ.x = -q1.x; conQ.y = -q1.y; conQ.z = -q1.z; conQ.w = q1.w;

dyn\_acc\_q\_earth = Quaternion.multiply(multQ, conQ);

fused\_alt = altitudeFilter.update(dyn\_acc\_q\_earth.z, altitude, dt);

}

void gravityCompensateDynAcc() {

float[] g = new float[3];

// get expected direction of gravity in the sensor frame

g[0] = 2 \* (q[1] \* q[3] - q[0] \* q[2]);

g[1] = 2 \* (q[0] \* q[1] + q[2] \* q[3]);

g[2] = q[0] \* q[0] - q[1] \* q[1] - q[2] \* q[2] + q[3] \* q[3];

// compensate accelerometer readings with the expected direction of gravity

dyn\_acc[0] = acc[0] - g[0];

dyn\_acc[1] = acc[1] - g[1];

dyn\_acc[2] = acc[2] - g[2];

}

Implemented heading corrections

Monday, January 20, 2014

6:24 AM

1/19/14

Completed the modifications to the heading calculation to provide semi-stable readings down to the 1/10th degree otherwise the readings were gyrating +- 1degree:

Here is the commented code:

From the headed:

// These are needed for the moving average calculation

float[] data = new float[32];

float total = 0, average = 0;

int p = 0, n = 0;

From Draw:

//Moving average Heading

float corr\_heading;

float heading\_avg;

float heading = 0;

float oldHeading = 0.0;

int windSize = 96;

MovingAverage HeadingAvg = new MovingAverage(windSize);

//round the LSM303 heading to 1 decimal place

float head1 = iround(heading,1);

//add the local declination angle and ensure the angle stays between 0-360 degs

corr\_heading = clamp360(head1+declinationAngle);

//average the heading readings and apply a LPF (not used - oldHeading - 0) otherwise you get yaw from north

HeadingAvg.newNum(HeadingAvgCorr(corr\_heading, oldHeading));

//oldHeading = corr\_heading;

corr\_heading = HeadingAvg.getAvg();

====== additional functions and classes

//=======================================

public float iround(float number, float decimal) {

int ix;

ix = round(number\*pow(10, decimal));

return float(ix)/pow(10, decimal);

}

//==========================================

// converted from Michael Shimniok Data Bus code

// <http://mbed.org/users/shimniok/code/AVC_20110423/>

float clamp360(float x) {

while ((x) >= 360.0) (x) -= 360.0;

while ((x) < 0) (x) += 360.0;

return x;

}

// ============================================

//

float HeadingAvgCorr(float newx, float oldx) {

while ((newx + 180.0) < oldx) (newx) += 360.0;

while ((newx - 180.0) > oldx) (newx) -= 360.0;

while ((newx) == 360.0) (newx) = 0.0;

return newx;

}

From RosettaCode and Stackflow added the following two classes for averages:

import java.util.LinkedList;

import java.util.Queue;

public class MovingAverage {

private final Queue<Float> window = new LinkedList<Float>();

private final int period;

private float sum;

public MovingAverage(int period) {

assert period > 0 : "Period must be a positive integer";

this.period = period;

}

public void newNum(float num) {

sum += num;

window.add(num);

if (window.size() > period) {

sum -= window.remove();

}

}

public float getAvg() {

if (window.isEmpty()) return 0; // technically the average is undefined

return sum / window.size();

}

}

public class EMA {

private float alpha;

private float oldValue;

public EMA(float alpha) {

this.alpha = alpha;

}

public float getAvg(float value){

//if(oldValue == null) {

// oldValue = value;

// return value;

//}

float newValue = oldValue + alpha \* (value - oldValue);

oldValue = newValue;

return newValue;

}

}

Reset Implementation

Sunday, January 19, 2014

10:54 AM

In Pierre-Huggues Husson logbook he identified two similar problems that I also have run into:

* The MPU-6050 needs to have its clock source settled before doing anything, while the register address of this choice is high. That was a problem not obvious to find, since the infos available doesn’t seem to mention this problem. What happened was that the chip wasn’t working on first power-on, but after reset it’s working.
* The FreeIMU board doesn’t have a RESET pin, so it might happen that the chip is in an appropriate state on reboot. It’s especially true because the current code is polling informations from the board, so the I2C bus is always working. This means that a reset was most likely in a middle of a communication. A fix to that problem is to fake the end of the I2C transmission: put SDA up, and toggle SCL enough times (10 times should be enough.)

To address this problem as well as apparent issues with quaternion matrix destabilizing after rapid motion and increasing yaw drift two different resets were implemented (manually for now):

1. Hard reset of the MPU-6050 chip:
   1. FreeIMU.cpp

void FreeIMU::RESET\_Q() {

//reset matrix

q0 = 1.0f;

q1 = 0.0f;

q2 = 0.0f;

q3 = 0.0f;

exInt = 0.0;

eyInt = 0.0;

ezInt = 0.0;

twoKp = twoKpDef;

twoKi = twoKiDef;

integralFBx = 0.0f, integralFBy = 0.0f, integralFBz = 0.0f;

//lastUpdate = 0;

//now = 0;

}

1. FreeIMU.h

void RESET();

1. Soft Reset of the Q matrices
   1. FreeIMU.h

void RESET\_Q();

1. FreeIMU.cpp

void FreeIMU::RESET\_Q() {

//reset matrix

q0 = 1.0f;

q1 = 0.0f;

q2 = 0.0f;

q3 = 0.0f;

exInt = 0.0;

eyInt = 0.0;

ezInt = 0.0;

twoKp = twoKpDef;

twoKi = twoKiDef;

integralFBx = 0.0f, integralFBy = 0.0f, integralFBz = 0.0f;

//lastUpdate = 0;

//now = 0;

}

1. In the Processing GUI I set up a manual serial command to reset either option (See highlighted portion of code snippet below):

void keyPressed() {

if(key == 'h') {

println("pressed h");

// set hq the home quaternion as the quatnion conjugate coming from the sensor fusion

hq = quatConjugate(q);

sw.start();

}

else if(key == 'n') {

println("pressed n");

hq = null;

}

**else if(key == '1') {**

**myPort.clear();**

**myPort.write("1");**

**println("pressed 1");**

**}**

**else if(key == '2') {**

**myPort.clear();**

**myPort.write("2");**

**println("pressed 2");**

}

else if(key == 's') {

println("pressed s");

sw.start();

}

}

Several changes will be made to test key pressed after each data burst and auto option to zero q matrix when zero motion detected

Implementation of changes to Kp and Ki

Monday, January 20, 2014

4:04 PM

1/19/14

This is the last update to get yaw drift down to zero:

//twoKpDef changed to xx from 0.5 based on trial and error using new

//temperature correction method

#define twoKpDef (2.0f \* 0.75f) // 2 \* proportional gain

//twoKiDef changed from 0.1 to 0f to match the values in Sebastian Madgwicks

//twoKiDef changed from 0f to 0.0025f based on article in DIYDrones

//twoKiDef changed to .1625 as a result of corrected temp - 1/4/14

//updated code

#define twoKiDef (2.0f \* 0.1625f) // 2 \* integral gain

Heres the final youtube video showing results of changes:

<http://youtu.be/lwlOZzNJqJc>

**new inverse sqrt code**

11-01-2013

int instability\_fix = 1;

float invSqrt(float x) {

if (instability\_fix == 0)

{

union {

float f;

int32\_t i;

} y;

y.f = x;

y.i = 0x5f375a86 - (y.i >> 1);

y.f = y.f \* ( 1.5f - ( x \* 0.5f \* y.f \* y.f ) );

return y.f;

}

else if (instability\_fix == 1)

{

/\* close-to-optimal method with low cost from

http://pizer.wordpress.com/2008/10/12/fast-inverse-square-root \*/

uint32\_t i = 0x5F1F1412 - (\*(uint32\_t\*)&x >> 1);

float tmp = \*(float\*)&i;

return tmp \* (1.69000231f - 0.714158168f \* x \* tmp \* tmp);

}

else

{

/\* optimal but expensive method: \*/

return 1.0f / sqrt(x);

}

}

/\*\*

\* Fast inverse square root implementation. Compatible both for 32 and 8 bit microcontrollers.

\* @see http://en.wikipedia.org/wiki/Fast\_inverse\_square\_root

\*/

/\* original code from FreeIMU Library

float invSqrt(float number) {

union {

float f;

int32\_t i;

} y;

y.f = number;

y.i = 0x5f375a86 - (y.i >> 1);

y.f = y.f \* ( 1.5f - ( number \* 0.5f \* y.f \* y.f ) );

return y.f;

}

\*/

/\* Old 8bit version. Kept it here only for testing/debugging of the new code above.

float invSqrt(float number) {

volatile long i;

volatile float x, y;

volatile const float f = 1.5F;

x = number \* 0.5F;

y = number;

i = \* ( long \* ) &y;

i = 0x5f375a86 - ( i >> 1 );

y = \* ( float \* ) &i;

y = y \* ( f - ( x \* y \* y ) );

return y;

}

\*/

Other changes:

* 1. 1D Kalman Filter on q-values in sketch
  2. Additional fields added to raw value (‘r’) and ‘z’. Added ‘a’ for kalman filtered values

/\*\*

\* FreeIMU library serial communication protocol

\*/

#include <ADXL345.h>

#include <bma180.h>

#include <HMC58X3.h>

#include <ITG3200.h>

#include <MS561101BA.h>

#include <I2Cdev.h>

#include <MPU60X0.h>

#include <LSM303.h>

#include <EEPROM.h>

#include <Wire.h>

#include <SPI.h>

//#define DEBUG

#include "DebugUtils.h"

#include "CommunicationUtils.h"

#include "FreeIMU.h"

#define Has\_LSM303 1

#include "FilteringScheme.h"

KalmanFilter kFilters[4];

int k\_index = 4;

float q[4];

int raw\_values[11];

float ypr[3]; // yaw pitch roll

char str[256];

float val[9];

float val\_array[17];

// Set the FreeIMU object and LSM303 Compass

FreeIMU my3IMU = FreeIMU();

#if Has\_LSM303

//Set up tilt corrected LSM303D

LSM303 compass;

float declinationAngle = 0.229622;

float heading\_corr = -9999.;

#endif

//The command from the PC

char cmd;

void setup() {

Serial.begin(38400);

Wire.begin();

float qVal = 0.125; //Set Q Kalman Filter(process noise) value between 0 and 1

float rVal = 32.; //Set K Kalman Filter (sensor noise)

for(int i = 0; i <= k\_index; i++) { //Initialize Kalman Filters for 10 neighbors

//KalmanFilter(float q, float r, float p, float intial\_value);

kFilters[i].KalmanInit(qVal,rVal,5.0,0.5);

}

//#if HAS\_MPU6050()

// my3IMU.RESET();

//#endif

my3IMU.init(true);

#if Has\_LSM303

compass.init();

compass.enableDefault();

/\*

Calibration values; the default values of +/-32767 for each axis

lead to an assumed magnetometer bias of 0. Use the Calibrate example

program to determine appropriate values for your particular unit.

\*/

//compass.m\_min = (LSM303::vector<int16\_t>){-32767, -32767, -32767};

//compass.m\_max = (LSM303::vector<int16\_t>){+32767, +32767, +32767};

//compass.m\_min = (LSM303::vector<int16\_t>){-3317, -4057, -3161};

//compass.m\_max = (LSM303::vector<int16\_t>){+2494, +1743, +2584};

//compass.m\_min = (LSM303::vector<int16\_t>){-2653, -3008, -2907};

//compass.m\_max = (LSM303::vector<int16\_t>){+2624, +2677, +2687};

//compass.m\_min = (LSM303::vector<int16\_t>){-3033, -3509, -3162};

//compass.m\_max = (LSM303::vector<int16\_t>){+3370, +2903, +3074};

compass.m\_min = (LSM303::vector<int16\_t>){-2815, -3090, -2958};

compass.m\_max = (LSM303::vector<int16\_t>){+2946, +2654, +2734};

#endif

// LED

pinMode(13, OUTPUT);

}

void loop() {

if(Serial.available()) {

cmd = Serial.read();

if(cmd=='v') {

sprintf(str, "FreeIMU library by %s, FREQ:%s, LIB\_VERSION: %s, IMU: %s", FREEIMU\_DEVELOPER, FREEIMU\_FREQ, FREEIMU\_LIB\_VERSION, FREEIMU\_ID);

Serial.print(str);

Serial.print('\n');

}

else if(cmd=='1'){

#if HAS\_MPU6050()

my3IMU.RESET();

#endif

my3IMU.init(true);

}

else if(cmd=='2'){

my3IMU.RESET\_Q();

}

else if(cmd=='r') {

uint8\_t count = serial\_busy\_wait();

for(uint8\_t i=0; i<count; i++) {

my3IMU.getRawValues(raw\_values);

sprintf(str, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,", raw\_values[0], raw\_values[1], raw\_values[2], raw\_values[3], raw\_values[4], raw\_values[5], raw\_values[6], raw\_values[7], raw\_values[8], raw\_values[9]);

Serial.print(str);

#if HAS\_MS5611()

Serial.print(my3IMU.getBaroTemperature()); Serial.print(",");

Serial.print(my3IMU.getBaroPressure()); Serial.print(",");

#endif

#if Has\_LSM303

compass.read();

Serial.print(compass.heading());Serial.print(",");

#endif

Serial.print(millis()); Serial.print(",");

Serial.println("\r\n");

}

}

else if(cmd=='b') {

uint8\_t count = serial\_busy\_wait();

for(uint8\_t i=0; i<count; i++) {

#if HAS\_ITG3200()

my3IMU.acc.readAccel(&raw\_values[0], &raw\_values[1], &raw\_values[2]);

my3IMU.gyro.readGyroRaw(&raw\_values[3], &raw\_values[4], &raw\_values[5]);

#else // MPU6050

my3IMU.accgyro.getMotion6(&raw\_values[0], &raw\_values[1], &raw\_values[2], &raw\_values[3], &raw\_values[4], &raw\_values[5]);

#endif

writeArr(raw\_values, 6, sizeof(int)); // writes accelerometer and gyro values

#if IS\_9DOM()

my3IMU.magn.getValues(&raw\_values[0], &raw\_values[1], &raw\_values[2]);

writeArr(raw\_values, 3, sizeof(int));

#endif

Serial.println();

}

}

else if(cmd == 'q') {

uint8\_t count = serial\_busy\_wait();

for(uint8\_t i=0; i<count; i++) {

my3IMU.getQ(q);

serialPrintFloatArr(q, 4);

Serial.println("");

}

}

else if(cmd == 'z') {

float val\_array[17] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

uint8\_t count = serial\_busy\_wait();

for(uint8\_t i=0; i<count; i++) {

my3IMU.getQ(q);

my3IMU.getValues(val);

val\_array[7] = (val[3] \* M\_PI/180);

val\_array[8] = (val[4] \* M\_PI/180);

val\_array[9] = (val[5] \* M\_PI/180);

val\_array[4] = (val[0]);

val\_array[5] = (val[1]);

val\_array[6] = (val[2]);

val\_array[10] = (val[6]);

val\_array[11] = (val[7]);

val\_array[12] = (val[8]);

val\_array[0] = (q[0]);

val\_array[1] = (q[1]);

val\_array[2] = (q[2]);

val\_array[3] = (q[3]);

val\_array[15] = millis();

#if Has\_LSM303

compass.read();

val\_array[16] = compass.heading();

#endif

#if HAS\_MS5611()

// with baro

val\_array[13] = (my3IMU.getBaroTemperature());

val\_array[14] = (my3IMU.getBaroPressure());

#endif

serialPrintFloatArr(val\_array,17);

Serial.print('\n');

}

}

else if(cmd == 'a') {

float val\_array[17] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

uint8\_t count = serial\_busy\_wait();

for(uint8\_t i=0; i<count; i++) {

my3IMU.getQ(q);

my3IMU.getValues(val);

val\_array[7] = (val[3] \* M\_PI/180);

val\_array[8] = (val[4] \* M\_PI/180);

val\_array[9] = (val[5] \* M\_PI/180);

val\_array[4] = (val[0]);

val\_array[5] = (val[1]);

val\_array[6] = (val[2]);

val\_array[10] = (val[6]);

val\_array[11] = (val[7]);

val\_array[12] = (val[8]);

val\_array[0] = kFilters[0].measureRSSI(q[0]);

val\_array[1] = kFilters[1].measureRSSI(q[1]);

val\_array[2] = kFilters[2].measureRSSI(q[2]);

val\_array[3] = kFilters[3].measureRSSI(q[3]);

val\_array[15] = millis();

#if Has\_LSM303

compass.read();

val\_array[16] = compass.heading();

#endif

#if HAS\_MS5611()

// with baro

val\_array[13] = (my3IMU.getBaroTemperature());

val\_array[14] = (my3IMU.getBaroPressure());

#endif

serialPrintFloatArr(val\_array, 17);

Serial.println("");

}

}

#ifndef CALIBRATION\_H

else if(cmd == 'c') {

const uint8\_t eepromsize = sizeof(float) \* 6 + sizeof(int) \* 6;

while(Serial.available() < eepromsize) ; // wait until all calibration data are received

EEPROM.write(FREEIMU\_EEPROM\_BASE, FREEIMU\_EEPROM\_SIGNATURE);

for(uint8\_t i = 1; i<(eepromsize + 1); i++) {

EEPROM.write(FREEIMU\_EEPROM\_BASE + i, (char) Serial.read());

}

my3IMU.calLoad(); // reload calibration

// toggle LED after calibration store.

digitalWrite(13, HIGH);

delay(1000);

digitalWrite(13, LOW);

}

else if(cmd == 'x') {

EEPROM.write(FREEIMU\_EEPROM\_BASE, 0); // reset signature

my3IMU.calLoad(); // reload calibration

}

#endif

else if(cmd == 'C') { // check calibration values

Serial.print("acc offset: ");

Serial.print(my3IMU.acc\_off\_x);

Serial.print(",");

Serial.print(my3IMU.acc\_off\_y);

Serial.print(",");

Serial.print(my3IMU.acc\_off\_z);

Serial.print("\n");

Serial.print("magn offset: ");

Serial.print(my3IMU.magn\_off\_x);

Serial.print(",");

Serial.print(my3IMU.magn\_off\_y);

Serial.print(",");

Serial.print(my3IMU.magn\_off\_z);

Serial.print("\n");

Serial.print("acc scale: ");

Serial.print(my3IMU.acc\_scale\_x);

Serial.print(",");

Serial.print(my3IMU.acc\_scale\_y);

Serial.print(",");

Serial.print(my3IMU.acc\_scale\_z);

Serial.print("\n");

Serial.print("magn scale: ");

Serial.print(my3IMU.magn\_scale\_x);

Serial.print(",");

Serial.print(my3IMU.magn\_scale\_y);

Serial.print(",");

Serial.print(my3IMU.magn\_scale\_z);

Serial.print("\n");

}

else if(cmd == 'd') { // debugging outputs

while(1) {

my3IMU.getRawValues(raw\_values);

sprintf(str, "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,", raw\_values[0], raw\_values[1], raw\_values[2], raw\_values[3], raw\_values[4], raw\_values[5], raw\_values[6], raw\_values[7], raw\_values[8], raw\_values[9], raw\_values[10]);

Serial.print(str);

Serial.print('\n');

my3IMU.getQ(q);

serialPrintFloatArr(q, 4);

Serial.println("");

my3IMU.getYawPitchRoll(ypr);

Serial.print("Yaw: ");

Serial.print(ypr[0]);

Serial.print(" Pitch: ");

Serial.print(ypr[1]);

Serial.print(" Roll: ");

Serial.print(ypr[2]);

Serial.println("");

}

}

}

}

char serial\_busy\_wait() {

while(!Serial.available()) {

; // do nothing until ready

}

return Serial.read();

}

const int EEPROM\_MIN\_ADDR = 0;

const int EEPROM\_MAX\_ADDR = 511;

void eeprom\_serial\_dump\_column() {

// counter

int i;

// byte read from eeprom

byte b;

// buffer used by sprintf

char buf[10];

for (i = EEPROM\_MIN\_ADDR; i <= EEPROM\_MAX\_ADDR; i++) {

b = EEPROM.read(i);

sprintf(buf, "%03X: %02X", i, b);

Serial.println(buf);

}

}

1/20/14

See processing sketch for all changes to the sketch as describe in change log.