Quadcopter
Teknologi og Forskningslre
Ullern VGS

Karsten S. Stadler
Martin Due Andersen
Thorvald Molthe Ballestad

Faglrer: Eivind Tjensvoll

May 11, 2015

Contents

Abstract

Vi har i skoleret 2014-15 designet, bygget og programmert et quadcopter. Denne teksten skal ta for seg prossessen, fra visjon til ferdig produkt, og gi en inngende forklaring i bde fastvaren og programvaren.

1 Visjon

Mlet med prosjektet var, fra starten av, konstruere et quadcopter med passelige flyegenskaper. Det var derimot ogs motsetninger innad i gruppen, Martin og Karsten var hovedsakelig opptatt av de cinematografiske mulighetene, mens Thorvald nsket automatisere s mye som mulig.

2 Fremgangsmte

Gruppen hadde mellom seg svrt lite kunnskap om Arduino, fastvare og quadcoptre, det ble derfor brukt mye tid p research. Forum, Arduino Playground og YouTube ble i stor grad brukt, da man kan lre mye av andres erfaringer.

3 Produkt

3.1 Programvare

3.2 Fastvare

4 Vedlegg

4.1 Quadcopter kildekode

```
#define SAFE
//#define DEBUG

//----------ESC's config--------------------

//PWM values
#define MOTOR_ZERO_LEVEL 125
#define MOTOR_ARM 140
#define MOTOR_MAX_LEVEL 254

//Motor pins
#ifdef _DUE_BOARD
#define MOTOR_FR 2
#define MOTOR_FL 5
#define MOTOR_BR 3
#define MOTOR_BL 4
#elif defined _UNO_BOARD
#define MOTOR_BR 10
#define MOTOR_BL 9
#define MOTOR_FR 3
#define MOTOR_FL 11
#endif

//----------Rx config----------------------
#define RX_TIMEOUT 2200
//------Pins------
#ifdef _DUE_BOARD
#define RX_PIN_THROTTLE 10
#define RX_PIN_PITCH 9
#define RX_PIN_ROLL 8
#define RX_PIN_YAW 11
//Auxillary - button and knobs
#define RX_PIN_AUX1 12
#define RX_PIN_AUX2 NULL

#elif defined _UNO_BOARD
#define RX_PIN_THROTTLE 2
#define RX_PIN_PITCH 1
#define RX_PIN_ROLL 4
#define RX_PIN_YAW 0

//Auxillary - button and knobs
```

```
#define RX_PIN_AUX1 5
#define RX_PIN_AUX2 NULL
#endif

//------KEY VALUES---
#define RX_THROTTLE_MIN 1028
#define RX_THROTTLE_MAX 1864
#define THROTTLE_MAX MOTOR_MAX_LEVEL - (PITCH_MAX + ROLL_MAX + YAW_MAX)
#define THROTTLE_MIN MOTOR_ZERO_LEVEL + 40

#define RX_PITCH_MIN 1196
#define RX_PITCH_MAX 1703
#define PITCH_MIN -15
#define PITCH_MAX 15
#define PITCH_MIN_DEG -30
#define PITCH_MAX_DEG 30

#define RX_ROLL_MIN 1194
#define RX_ROLL_MAX 1703
#define ROLL_MIN -15
#define ROLL_MAX 15

#define RX_YAW_MIN 1035
#define RX_YAW_MAX 1867
#define YAW_MIN -15
#define YAW_MAX 15

#define RX_AUX1_MIN 1040
#define RX_AUX1_MAX 1866
#define AUX1_MIN -15
#define AUX1_MAX 15

#define RX_AUX2_MIN
#define RX_AUX2_MAX
#define AUX2_MIN -15
#define AUX2_MAX 15
```

../Quadcopter/Quadcopter/config.h

```
void FlightController() {
  int motor[2][2]; //[F/B][R/L] motor[0][1]:motor front left

  float throttle, pitch, pitchSet, roll, rollSet, yaw, yawSet;
  //PID pidPitch, pidRoll; scope

  //motor values are computed by adding throttle, roll, yaw and pitch
  throttle = map(rxThrottle, RX_THROTTLE_MIN, RX_THROTTLE_MAX, THROTTLE_MIN,
    THROTTLE_MAX);

  //xxSet er  nsket verdi, setpoint. Her i antall grader. xx er det som skal sendes
    til motor
  //  pitch = map(rxPitch, RX_PITCH_MIN, RX_PITCH_MAX, PITCH_MIN, PITCH_MAX);
  pitchSet = map(rxPitch, RX_PITCH_MIN, RX_PITCH_MAX, PITCH_MIN_DEG, PITCH_MAX_DEG);
    //Regner ut  nsket hellning p  pitch
  rollSet  = map(rxRoll, RX_ROLL_MIN, RX_ROLL_MAX, ROLL_MIN, ROLL_MAX);
  yawSet   = map(rxYaw, RX_YAW_MIN, RX_YAW_MAX, YAW_MIN, YAW_MAX);


  pidPitch.update(0);
  pidRoll.update(0);

  pitch = pitchSet + pidPitch.evaluate(angles[1])*pidSensitivity;
  roll = rollSet + pidRoll.evaluate(angles[2])*pidSensitivity;
  yaw = yawSet;

  int i, j;
  for(i = 0; i < 2; i++) {
    for(j = 0; j < 2; j++) {
      motor[i][j] = (int) throttle\
  + pitch*zeroToMinus(i)\
  + roll*zeroToMinus(j)\
  + yaw*zeroToMinus(i xor j);
    }
  }


```

```
#ifdef SAFE
  if(rxAux1 < (RX_AUX1_MAX + RX_AUX1_MIN)/2 ) {
    for(i=0; i<2; i++) {
      for(j=0; j<2; j++) {
  motor[i][j] = MOTOR_ZERO_LEVEL;
      }
    }
  }
#endif

#ifdef STOP_MAX //Don't know if this will stay or not, if successfull, remove if
    for(i=0; i<2; i++) {
      for(j=0; j<2; j++) {
  if(motor[i][j] > MOTOR_MAX_LEVEL)
    motor[i][j] = MOTOR_MAX_LEVEL;

  if(motor[i][j] < MOTOR_ZERO_LEVEL)
    motor[i][j] = MOTOR_ZERO_LEVEL;
      }
    }
#endif

  Serial.println(motor[0][0]);
  /*
  Serial.print(motor[0][1]);
  Serial.print(motor[1][0]);
  Serial.println(motor[1][1]);
   */

  //PWM might damage motors
  analogWrite(MOTOR_FR, motor[0][0]);
  analogWrite(MOTOR_FL, motor[0][1]);
  analogWrite(MOTOR_BR, motor[1][0]);
  analogWrite(MOTOR_BL, motor[1][1]);
}

byte zeroToMinus(bool n) {
  return n ? 1 : -1; //1 is 1 and 0 is -1
}
```

../Quadcopter/Quadcopter/FlightController.ino

```
void Gyro() {
  sixDOF.getEuler(angles);
  //sixDOF.getYawPitchRoll(angles);
  //sixDOF.getAngles(angles);
}
```

../Quadcopter/Quadcopter/Gyro.ino

```
/*
Karsten Sebastian Stadler, Martin Due Andersen and Thorvald Molthe Ballestad
 Ullern VGS - 2015

Quadcopter.ino is the main file.
*/

#define _DUE_BOARD
//#define _UNO_BOARD

#define STOP_MAX //stop motor values to exeeding extremals
#define DEBUG

#include "config.h"
#include <PID.h>
#include <Wire.h> //Library for    snakke med gyro/acc
#include <FreeSixIMU.h>
#include <FIMU_ADXL345.h>
#include <FIMU_ITG3200.h>

volatile unsigned int rxThrottle, rxPitch, rxRoll, rxYaw, rxAux1, rxAux2;
float angles[3];
PID pidPitch, pidRoll;
float pidSensitivity = 0.01;

FreeSixIMU sixDOF = FreeSixIMU(); //AccGyro
```

```
27
void setup() {
29   Wire.begin();
     rxInit();
31   #ifdef DEBUG
       Serial.begin(9600);
33   #endif
     delay(5);
35   sixDOF.init();
     delay(5); //delay for     v re sikker p  at gyroAcc starter opp. Kanskje ikke
       n dvendig med s  mye tid
37 }

39 void loop() {
     Gyro();
41   FlightController(); //writes appropriate values to motors using PID
   }
```

../Quadcopter/Quadcopter/Quadcopter.ino

```
   volatile int t[5];
2
   void rxInit() {
4    attachInterrupt(RX_PIN_THROTTLE, rxGoesUpThrottle, RISING);
     attachInterrupt(RX_PIN_PITCH, rxGoesUpPitch, RISING);
6    attachInterrupt(RX_PIN_ROLL, rxGoesUpRoll, RISING);
     attachInterrupt(RX_PIN_YAW, rxGoesUpYaw, RISING);
8    attachInterrupt(RX_PIN_AUX1, rxGoesUpAux1, RISING);
   }
10
   //Not the most elegant solution, but it works(hopefully)
12 void rxGoesUpThrottle() {
     attachInterrupt(RX_PIN_THROTTLE, rxGoesDownThrottle, FALLING);
14   t[0]=micros();
   }
16
   void rxGoesUpPitch() {
18   attachInterrupt(RX_PIN_PITCH, rxGoesDownPitch, FALLING);
     t[1]=micros();
20 }

22 void rxGoesUpRoll() {
     attachInterrupt(RX_PIN_ROLL, rxGoesDownRoll, FALLING);
24   t[2]=micros();
   }
26
   void rxGoesUpYaw() {
28   attachInterrupt(RX_PIN_YAW, rxGoesDownYaw, FALLING);
     t[3]=micros();
30 }

32 void rxGoesUpAux1() {
     attachInterrupt(RX_PIN_AUX1, rxGoesDownAux1, FALLING);
34   t[4]=micros();
   }
36
   void rxGoesDownThrottle() {
38   rxThrottle = micros() - t[0];
     attachInterrupt(RX_PIN_THROTTLE, rxGoesUpThrottle, RISING);
40 }

42 void rxGoesDownPitch() {
     rxPitch = micros() - t[1];
44   attachInterrupt(RX_PIN_PITCH, rxGoesUpPitch, RISING);
   }
46
   void rxGoesDownRoll() {
48   rxRoll = micros() - t[2];
     attachInterrupt(RX_PIN_ROLL, rxGoesUpRoll, RISING);
50 }

52 void rxGoesDownYaw() {
    rxYaw = micros() - t[3];
54   attachInterrupt(RX_PIN_YAW, rxGoesUpYaw, RISING);
   }
56
```

```
void rxGoesDownAux1() {
  rxAux1 = micros() - t[4];
  attachInterrupt(RX_PIN_AUX1, rxGoesUpAux1, RISING);
}
```

../Quadcopter/Quadcopter/RX.ino

```
#include "CommunicationUtils.h"

void serialPrintFloatArr(float * arr, int length) {
  for(int i=0; i<length; i++) {
    serialFloatPrint(arr[i]);
    Serial.print(",");
  }
}


void serialFloatPrint(float f) {
  byte * b = (byte *) &f;
  for(int i=0; i<4; i++) {

    byte b1 = (b[i] >> 4) & 0x0f;
    byte b2 = (b[i] & 0x0f);

    char c1 = (b1 < 10) ? ('0' + b1) : 'A' + b1 - 10;
    char c2 = (b2 < 10) ? ('0' + b2) : 'A' + b2 - 10;

    Serial.print(c1);
    Serial.print(c2);
  }
}


void writeArr(void * varr, uint8_t arr_length, uint8_t type_bytes) {
  byte * arr = (byte*) varr;
  for(uint8_t i=0; i<arr_length; i++) {
    writeVar(&arr[i * type_bytes], type_bytes);
  }
}


// thanks to Francesco Ferrara and the Simplo project for the following code!
void writeVar(void * val, uint8_t type_bytes) {
  byte * addr=(byte *)(val);
  for(uint8_t i=0; i<type_bytes; i++) {
    Serial.write(addr[i]);
  }
}
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/CommunicationUtils.cpp

```
#ifndef CommunitationUtils_h
#define CommunitationUtils_h

#include "Arduino.h"

void serialPrintFloatArr(float * arr, int length);
void serialFloatPrint(float f);
void writeArr(void * arr, uint8_t arr_length, uint8_t type_bytes);
void writeVar(void * val, uint8_t type_bytes);


#endif // CommunitationUtils_h
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/CommunicationUtils.h

```
/*
DebugUtils.h - Simple debugging utilities.
Copyright (C) 2011 Fabio Varesano <fabio at varesano dot net>

Ideas taken from:
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1271517197

This program is free software: you can redistribute it and/or modify
it under the terms of the version 3 GNU General Public License as
```

```
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

*/

#ifndef DEBUGUTILS_H
#define DEBUGUTILS_H

#ifdef DEBUG_V
  #include <WProgram.h>
  #define DEBUG_PRINT(str)    \
     Serial.print(millis());     \
     Serial.print(": ");    \
     Serial.print(__PRETTY_FUNCTION__); \
     Serial.print(' ');      \
     Serial.print(__FILE__);     \
     Serial.print(':');      \
     Serial.print(__LINE__);     \
     Serial.print(' ');      \
     Serial.println(str);
#endif

#ifdef DEBUG
  #define DEBUG_PRINT(str) \
     Serial.println(str);
#endif

#ifndef DEBUG_PRINT
  #define DEBUG_PRINT(str)
#endif


#endif //DEBUGUTILS_H
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/DebugUtils.h

```
/*****************************************************************************
 *                                                                           *
 * ADXL345 Driver for Arduino                                                *
 *                                                                           *
 *****************************************************************************
 *                                                                           *
 * This program is free software; you can redistribute it and/or modify      *
 * it under the terms of the GNU License.                                    *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU License V2 for more details.                                          *
 *                                                                           *
 *****************************************************************************/

#include "FIMU_ADXL345.h"
#include <Wire.h>

#define TO_READ (6)      // num of bytes we are going to read each time (two bytes
    for each axis)

ADXL345::ADXL345() {
  status = ADXL345_OK;
  error_code = ADXL345_NO_ERROR;

  gains[0] = 0.00376390;
  gains[1] = 0.00376009;
  gains[2] = 0.00349265;
}

void ADXL345::init(int address) {
  _dev_address = address;
  powerOn();
```

```
}

void ADXL345::powerOn() {
  //Turning on the ADXL345
  //writeTo(ADXL345_POWER_CTL, 0);
  //writeTo(ADXL345_POWER_CTL, 16);
  writeTo(ADXL345_POWER_CTL, 8);
}

// Reads the acceleration into an array of three places
void ADXL345::readAccel(int *xyz){
  readAccel(xyz, xyz + 1, xyz + 2);
}

// Reads the acceleration into three variable x, y and z
void ADXL345::readAccel(int *x, int *y, int *z) {
  readFrom(ADXL345_DATAX0, TO_READ, _buff); //read the acceleration data from the
    ADXL345

  // each axis reading comes in 10 bit resolution, ie 2 bytes.  Least Significat Byte
     first!!
  // thus we are converting both bytes in to one int
  *x = (((int)_buff[1]) << 8) | _buff[0];
  *y = (((int)_buff[3]) << 8) | _buff[2];
  *z = (((int)_buff[5]) << 8) | _buff[4];
}

void ADXL345::get_Gxyz(float *xyz){
  int i;
  int xyz_int[3];
  readAccel(xyz_int);
  for(i=0; i<3; i++){
    xyz[i] = xyz_int[i] * gains[i];
  }
}

// Writes val to address register on device
void ADXL345::writeTo(byte address, byte val) {
  Wire.beginTransmission(_dev_address); // start transmission to device
  Wire.write(address);              // send register address
  Wire.write(val);                  // send value to write
  Wire.endTransmission();          // end transmission
}

// Reads num bytes starting from address register on device in to _buff array
void ADXL345::readFrom(byte address, int num, byte _buff[]) {
  Wire.beginTransmission(_dev_address); // start transmission to device
  Wire.write(address);              // sends address to read from
  Wire.endTransmission();          // end transmission

  Wire.beginTransmission(_dev_address); // start transmission to device
  Wire.requestFrom(_dev_address, num);    // request 6 bytes from device

  int i = 0;
  while(Wire.available())          // device may send less than requested (abnormal)
  {
    _buff[i] = Wire.read();    // receive a byte
    i++;
  }
  if(i != num){
    status = ADXL345_ERROR;
    error_code = ADXL345_READ_ERROR;
  }
  Wire.endTransmission();          // end transmission
}

// Gets the range setting and return it into rangeSetting
// it can be 2, 4, 8 or 16
void ADXL345::getRangeSetting(byte* rangeSetting) {
  byte _b;
  readFrom(ADXL345_DATA_FORMAT, 1, &_b);
  *rangeSetting = _b & B00000011;
}

// Sets the range setting, possible values are: 2, 4, 8, 16
void ADXL345::setRangeSetting(int val) {
```

```
107   byte _s;
      byte _b;
109
      switch (val) {
111   case 2:
        _s = B00000000;
113     break;
      case 4:
115     _s = B00000001;
        break;
117   case 8:
        _s = B00000010;
119     break;
      case 16:
121     _s = B00000011;
        break;
123   default:
        _s = B00000000;
125   }
      readFrom(ADXL345_DATA_FORMAT, 1, &_b);
127   _s |= (_b & B11101100);
      writeTo(ADXL345_DATA_FORMAT, _s);
129 }
    // gets the state of the SELF_TEST bit
131 bool ADXL345::getSelfTestBit() {
      return getRegisterBit(ADXL345_DATA_FORMAT, 7);
133 }

135 // Sets the SELF-TEST bit
    // if set to 1 it applies a self-test force to the sensor causing a shift in the
        output data
137 // if set to 0 it disables the self-test force
    void ADXL345::setSelfTestBit(bool selfTestBit) {
139   setRegisterBit(ADXL345_DATA_FORMAT, 7, selfTestBit);
    }
141
    // Gets the state of the SPI bit
143 bool ADXL345::getSpiBit() {
      return getRegisterBit(ADXL345_DATA_FORMAT, 6);
145 }

147 // Sets the SPI bit
    // if set to 1 it sets the device to 3-wire mode
149 // if set to 0 it sets the device to 4-wire SPI mode
    void ADXL345::setSpiBit(bool spiBit) {
151   setRegisterBit(ADXL345_DATA_FORMAT, 6, spiBit);
    }
153
    // Gets the state of the INT_INVERT bit
155 bool ADXL345::getInterruptLevelBit() {
      return getRegisterBit(ADXL345_DATA_FORMAT, 5);
157 }

159 // Sets the INT_INVERT bit
    // if set to 0 sets the interrupts to active high
161 // if set to 1 sets the interrupts to active low
    void ADXL345::setInterruptLevelBit(bool interruptLevelBit) {
163   setRegisterBit(ADXL345_DATA_FORMAT, 5, interruptLevelBit);
    }
165
    // Gets the state of the FULL_RES bit
167 bool ADXL345::getFullResBit() {
      return getRegisterBit(ADXL345_DATA_FORMAT, 3);
169 }

171 // Sets the FULL_RES bit
    // if set to 1, the device is in full resolution mode, where the output resolution
        increases with the
173 //   g range set by the range bits to maintain a 4mg/LSB scal factor
    // if set to 0, the device is in 10-bit mode, and the range buts determine the
        maximum g range
175 //   and scale factor
    void ADXL345::setFullResBit(bool fullResBit) {
177   setRegisterBit(ADXL345_DATA_FORMAT, 3, fullResBit);
    }
179
```

```
// Gets the state of the justify bit
bool ADXL345::getJustifyBit() {
  return getRegisterBit(ADXL345_DATA_FORMAT, 2);
}

// Sets the JUSTIFY bit
// if sets to 1 selects the left justified mode
// if sets to 0 selects right justified mode with sign extension
void ADXL345::setJustifyBit(bool justifyBit) {
  setRegisterBit(ADXL345_DATA_FORMAT, 2, justifyBit);
}

// Sets the THRESH_TAP byte value
// it should be between 0 and 255
// the scale factor is 62.5 mg/LSB
// A value of 0 may result in undesirable behavior
void ADXL345::setTapThreshold(int tapThreshold) {
  tapThreshold = min(max(tapThreshold,0),255);
  byte _b = byte (tapThreshold);
  writeTo(ADXL345_THRESH_TAP, _b);
}

// Gets the THRESH_TAP byte value
// return value is comprised between 0 and 255
// the scale factor is 62.5 mg/LSB
int ADXL345::getTapThreshold() {
  byte _b;
  readFrom(ADXL345_THRESH_TAP, 1, &_b);
  return int (_b);
}

// set/get the gain for each axis in Gs / count
void ADXL345::setAxisGains(float *_gains){
  int i;
  for(i = 0; i < 3; i++){
    gains[i] = _gains[i];
  }
}
void ADXL345::getAxisGains(float *_gains){
  int i;
  for(i = 0; i < 3; i++){
    _gains[i] = gains[i];
  }
}


// Sets the OFSX, OFSY and OFSZ bytes
// OFSX, OFSY and OFSZ are user offset adjustments in twos complement format with
// a scale factor of 15,6mg/LSB
// OFSX, OFSY and OFSZ should be comprised between
void ADXL345::setAxisOffset(int x, int y, int z) {
  writeTo(ADXL345_OFSX, byte (x));
  writeTo(ADXL345_OFSY, byte (y));
  writeTo(ADXL345_OFSZ, byte (z));
}

// Gets the OFSX, OFSY and OFSZ bytes
void ADXL345::getAxisOffset(int* x, int* y, int*z) {
  byte _b;
  readFrom(ADXL345_OFSX, 1, &_b);
  *x = int (_b);
  readFrom(ADXL345_OFSY, 1, &_b);
  *y = int (_b);
  readFrom(ADXL345_OFSZ, 1, &_b);
  *z = int (_b);
}

// Sets the DUR byte
// The DUR byte contains an unsigned time value representing the maximum time
// that an event must be above THRESH_TAP threshold to qualify as a tap event
// The scale factor is 625 s/LSB
// A value of 0 disables the tap/float tap funcitons. Max value is 255.
void ADXL345::setTapDuration(int tapDuration) {
  tapDuration = min(max(tapDuration,0),255);
  byte _b = byte (tapDuration);
  writeTo(ADXL345_DUR, _b);
```

```
}

// Gets the DUR byte
int ADXL345::getTapDuration() {
  byte _b;
  readFrom(ADXL345_DUR, 1, &_b);
  return int (_b);
}

// Sets the latency (latent register) which contains an unsigned time value
// representing the wait time from the detection of a tap event to the start
// of the time window, during which a possible second tap can be detected.
// The scale factor is 1.25ms/LSB. A value of 0 disables the float tap function.
// It accepts a maximum value of 255.
void ADXL345::setDoubleTapLatency(int floatTapLatency) {
  byte _b = byte (floatTapLatency);
  writeTo(ADXL345_LATENT, _b);
}

// Gets the Latent value
int ADXL345::getDoubleTapLatency() {
  byte _b;
  readFrom(ADXL345_LATENT, 1, &_b);
  return int (_b);
}

// Sets the Window register, which contains an unsigned time value representing
// the amount of time after the expiration of the latency time (Latent register)
// during which a second valud tap can begin. The scale factor is 1.25ms/LSB. A
// value of 0 disables the float tap function. The maximum value is 255.
void ADXL345::setDoubleTapWindow(int floatTapWindow) {
  floatTapWindow = min(max(floatTapWindow,0),255);
  byte _b = byte (floatTapWindow);
  writeTo(ADXL345_WINDOW, _b);
}

// Gets the Window register
int ADXL345::getDoubleTapWindow() {
  byte _b;
  readFrom(ADXL345_WINDOW, 1, &_b);
  return int (_b);
}

// Sets the THRESH_ACT byte which holds the threshold value for detecting activity.
// The data format is unsigned, so the magnitude of the activity event is compared
// with the value is compared with the value in the THRESH_ACT register. The scale
// factor is 62.5mg/LSB. A value of 0 may result in undesirable behavior if the
// activity interrupt is enabled. The maximum value is 255.
void ADXL345::setActivityThreshold(int activityThreshold) {
  activityThreshold = min(max(activityThreshold,0),255);
  byte _b = byte (activityThreshold);
  writeTo(ADXL345_THRESH_ACT, _b);
}

// Gets the THRESH_ACT byte
int ADXL345::getActivityThreshold() {
  byte _b;
  readFrom(ADXL345_THRESH_ACT, 1, &_b);
  return int (_b);
}

// Sets the THRESH_INACT byte which holds the threshold value for detecting
    inactivity.
// The data format is unsigned, so the magnitude of the inactivity event is compared
// with the value is compared with the value in the THRESH_INACT register. The scale
// factor is 62.5mg/LSB. A value of 0 may result in undesirable behavior if the
// inactivity interrupt is enabled. The maximum value is 255.
void ADXL345::setInactivityThreshold(int inactivityThreshold) {
  inactivityThreshold = min(max(inactivityThreshold,0),255);
  byte _b = byte (inactivityThreshold);
  writeTo(ADXL345_THRESH_INACT, _b);
}

// Gets the THRESH_INACT byte
int ADXL345::getInactivityThreshold() {
  byte _b;
```

```
331   readFrom(ADXL345_THRESH_INACT, 1, &_b);
      return int (_b);
333 }

335 // Sets the TIME_INACT register, which contains an unsigned time value representing
        the
    // amount of time that acceleration must be less thant the value in the THRESH_INACT
337 // register for inactivity to be declared. The scale factor is 1sec/LSB. The value
        must
    // be between 0 and 255.
339 void ADXL345::setTimeInactivity(int timeInactivity) {
      timeInactivity = min(max(timeInactivity,0),255);
341   byte _b = byte (timeInactivity);
      writeTo(ADXL345_TIME_INACT, _b);
343 }

345 // Gets the TIME_INACT register
    int ADXL345::getTimeInactivity() {
347   byte _b;
      readFrom(ADXL345_TIME_INACT, 1, &_b);
349   return int (_b);
    }
351
    // Sets the THRESH_FF register which holds the threshold value, in an unsigned format
        , for
353 // free-fall detection. The root-sum-square (RSS) value of all axes is calculated and
    // compared whith the value in THRESH_FF to determine if a free-fall event occured.
        The
355 // scale factor is 62.5mg/LSB. A value of 0 may result in undesirable behavior if the
         free-fall
    // interrupt is enabled. The maximum value is 255.
357 void ADXL345::setFreeFallThreshold(int freeFallThreshold) {
      freeFallThreshold = min(max(freeFallThreshold,0),255);
359   byte _b = byte (freeFallThreshold);
      writeTo(ADXL345_THRESH_FF, _b);
361 }

363 // Gets the THRESH_FF register.
    int ADXL345::getFreeFallThreshold() {
365   byte _b;
      readFrom(ADXL345_THRESH_FF, 1, &_b);
367   return int (_b);
    }
369
    // Sets the TIME_FF register, which holds an unsigned time value representing the
        minimum
371 // time that the RSS value of all axes must be less than THRESH_FF to generate a free
        -fall
    // interrupt. The scale factor is 5ms/LSB. A value of 0 may result in undesirable
        behavior if
373 // the free-fall interrupt is enabled. The maximum value is 255.
    void ADXL345::setFreeFallDuration(int freeFallDuration) {
375   freeFallDuration = min(max(freeFallDuration,0),255);
      byte _b = byte (freeFallDuration);
377   writeTo(ADXL345_TIME_FF, _b);
    }
379
    // Gets the TIME_FF register.
381 int ADXL345::getFreeFallDuration() {
      byte _b;
383   readFrom(ADXL345_TIME_FF, 1, &_b);
      return int (_b);
385 }

387 bool ADXL345::isActivityXEnabled() {
      return getRegisterBit(ADXL345_ACT_INACT_CTL, 6);
389 }
    bool ADXL345::isActivityYEnabled() {
391   return getRegisterBit(ADXL345_ACT_INACT_CTL, 5);
    }
393 bool ADXL345::isActivityZEnabled() {
      return getRegisterBit(ADXL345_ACT_INACT_CTL, 4);
395 }
    bool ADXL345::isInactivityXEnabled() {
397   return getRegisterBit(ADXL345_ACT_INACT_CTL, 2);
    }
```

```
399 bool ADXL345::isInactivityYEnabled() {
      return getRegisterBit(ADXL345_ACT_INACT_CTL, 1);
401 }
    bool ADXL345::isInactivityZEnabled() {
403   return getRegisterBit(ADXL345_ACT_INACT_CTL, 0);
    }
405
    void ADXL345::setActivityX(bool state) {
407   setRegisterBit(ADXL345_ACT_INACT_CTL, 6, state);
    }
409 void ADXL345::setActivityY(bool state) {
      setRegisterBit(ADXL345_ACT_INACT_CTL, 5, state);
411 }
    void ADXL345::setActivityZ(bool state) {
413   setRegisterBit(ADXL345_ACT_INACT_CTL, 4, state);
    }
415 void ADXL345::setInactivityX(bool state) {
      setRegisterBit(ADXL345_ACT_INACT_CTL, 2, state);
417 }
    void ADXL345::setInactivityY(bool state) {
419   setRegisterBit(ADXL345_ACT_INACT_CTL, 1, state);
    }
421 void ADXL345::setInactivityZ(bool state) {
      setRegisterBit(ADXL345_ACT_INACT_CTL, 0, state);
423 }

425 bool ADXL345::isActivityAc() {
      return getRegisterBit(ADXL345_ACT_INACT_CTL, 7);
427 }
    bool ADXL345::isInactivityAc(){
429   return getRegisterBit(ADXL345_ACT_INACT_CTL, 3);
    }
431
    void ADXL345::setActivityAc(bool state) {
433   setRegisterBit(ADXL345_ACT_INACT_CTL, 7, state);
    }
435 void ADXL345::setInactivityAc(bool state) {
      setRegisterBit(ADXL345_ACT_INACT_CTL, 3, state);
437 }

439 bool ADXL345::getSuppressBit(){
      return getRegisterBit(ADXL345_TAP_AXES, 3);
441 }
    void ADXL345::setSuppressBit(bool state) {
443   setRegisterBit(ADXL345_TAP_AXES, 3, state);
    }
445
    bool ADXL345::isTapDetectionOnX(){
447   return getRegisterBit(ADXL345_TAP_AXES, 2);
    }
449 void ADXL345::setTapDetectionOnX(bool state) {
      setRegisterBit(ADXL345_TAP_AXES, 2, state);
451 }
    bool ADXL345::isTapDetectionOnY(){
453   return getRegisterBit(ADXL345_TAP_AXES, 1);
    }
455 void ADXL345::setTapDetectionOnY(bool state) {
      setRegisterBit(ADXL345_TAP_AXES, 1, state);
457 }
    bool ADXL345::isTapDetectionOnZ(){
459   return getRegisterBit(ADXL345_TAP_AXES, 0);
    }
461 void ADXL345::setTapDetectionOnZ(bool state) {
      setRegisterBit(ADXL345_TAP_AXES, 0, state);
463 }

465 bool ADXL345::isActivitySourceOnX(){
      return getRegisterBit(ADXL345_ACT_TAP_STATUS, 6);
467 }
    bool ADXL345::isActivitySourceOnY(){
469   return getRegisterBit(ADXL345_ACT_TAP_STATUS, 5);
    }
471 bool ADXL345::isActivitySourceOnZ(){
      return getRegisterBit(ADXL345_ACT_TAP_STATUS, 4);
473 }
```

```
475 bool ADXL345::isTapSourceOnX(){
      return getRegisterBit(ADXL345_ACT_TAP_STATUS, 2);
477 }
    bool ADXL345::isTapSourceOnY(){
479   return getRegisterBit(ADXL345_ACT_TAP_STATUS, 1);
    }
481 bool ADXL345::isTapSourceOnZ(){
      return getRegisterBit(ADXL345_ACT_TAP_STATUS, 0);
483 }

485 bool ADXL345::isAsleep(){
      return getRegisterBit(ADXL345_ACT_TAP_STATUS, 3);
487 }

489 bool ADXL345::isLowPower(){
      return getRegisterBit(ADXL345_BW_RATE, 4);
491 }
    void ADXL345::setLowPower(bool state) {
493   setRegisterBit(ADXL345_BW_RATE, 4, state);
    }
495
    float ADXL345::getRate(){
497   byte _b;
      readFrom(ADXL345_BW_RATE, 1, &_b);
499   _b &= B00001111;
      return (pow(2,((int) _b)-6)) * 6.25;
501 }

503 void ADXL345::setRate(float rate){
      byte _b,_s;
505   int v = (int) (rate / 6.25);
      int r = 0;
507   while (v >>= 1)
      {
509     r++;
      }
511   if (r <= 9) {
        readFrom(ADXL345_BW_RATE, 1, &_b);
513     _s = (byte) (r + 6) | (_b & B11110000);
        writeTo(ADXL345_BW_RATE, _s);
515   }
    }
517
    void ADXL345::set_bw(byte bw_code){
519   if((bw_code < ADXL345_BW_3) || (bw_code > ADXL345_BW_1600)){
        status = false;
521     error_code = ADXL345_BAD_ARG;
      }
523   else{
        writeTo(ADXL345_BW_RATE, bw_code);
525   }
    }
527
    byte ADXL345::get_bw_code(){
529   byte bw_code;
      readFrom(ADXL345_BW_RATE, 1, &bw_code);
531   return bw_code;
    }
533
    byte ADXL345::getInterruptSource() {
535   byte _b;
      readFrom(ADXL345_INT_SOURCE, 1, &_b);
537   return _b;
    }
539
    bool ADXL345::getInterruptSource(byte interruptBit) {
541   return getRegisterBit(ADXL345_INT_SOURCE,interruptBit);
    }
543
    bool ADXL345::getInterruptMapping(byte interruptBit) {
545   return getRegisterBit(ADXL345_INT_MAP,interruptBit);
    }
547
    // Set the mapping of an interrupt to pin1 or pin2
549 // eg: setInterruptMapping(ADXL345_INT_DOUBLE_TAP_BIT,ADXL345_INT2_PIN);
    void ADXL345::setInterruptMapping(byte interruptBit, bool interruptPin) {
```

```
551   setRegisterBit(ADXL345_INT_MAP, interruptBit, interruptPin);
}
553
bool ADXL345::isInterruptEnabled(byte interruptBit) {
555   return getRegisterBit(ADXL345_INT_ENABLE,interruptBit);
}
557
void ADXL345::setInterrupt(byte interruptBit, bool state) {
559   setRegisterBit(ADXL345_INT_ENABLE, interruptBit, state);
}
561
void ADXL345::setRegisterBit(byte regAdress, int bitPos, bool state) {
563   byte _b;
   readFrom(regAdress, 1, &_b);
565   if (state) {
     _b |= (1 << bitPos);  // forces nth bit of _b to be 1.  all other bits left alone
     .
567   }
   else {
569     _b &= ~(1 << bitPos); // forces nth bit of _b to be 0.  all other bits left alone
     .
   }
571   writeTo(regAdress, _b);
}
573
bool ADXL345::getRegisterBit(byte regAdress, int bitPos) {
575   byte _b;
   readFrom(regAdress, 1, &_b);
577   return ((_b >> bitPos) & 1);
}
579
// print all register value to the serial ouptut, which requires it to be setup
581 // this can be used to manually to check the current configuration of the device
void ADXL345::printAllRegister() {
583   byte _b;
   Serial.print("0x00: ");
585   readFrom(0x00, 1, &_b);
   print_byte(_b);
587   Serial.println("");
   int i;
589   for (i=29;i<=57;i++){
     Serial.print("0x");
591     Serial.print(i, HEX);
     Serial.print(": ");
593     readFrom(i, 1, &_b);
     print_byte(_b);
595     Serial.println("");
   }
597 }

599 void print_byte(byte val){
   int i;
601   Serial.print("B");
   for(i=7; i>=0; i--){
603     Serial.print(val >> i & 1, BIN);
   }
605 }
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FIMU_ADXL345.cpp

```
/*****************************************************************************
2  *                                                                           *
 * ADXL345 Driver for Arduino                                                *
4  *                                                                           *
 *****************************************************************************
6  *                                                                           *
 * This program is free software; you can redistribute it and/or modify      *
8  * it under the terms of the GNU License.                                    *
 * This program is distributed in the hope that it will be useful,           *
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of           *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
12 * GNU License V2 for more details.                                          *
 *                                                                           *
14 *****************************************************************************/
#include "Arduino.h"
16
```

```
#ifndef FIMU_ADXL345_h
#define FIMU_ADXL345_h

/* -- ADXL345 addresses --*/
#define ADXL345_ADDR_ALT_HIGH 0x1D // ADXL345 address when ALT is connected to HIGH
#define ADXL345_ADDR_ALT_LOW  0x53 // ADXL345 address when ALT is connected to LOW

/* ------- Register names ------- */
#define ADXL345_DEVID 0x00
#define ADXL345_RESERVED1 0x01
#define ADXL345_THRESH_TAP 0x1d
#define ADXL345_OFSX 0x1e
#define ADXL345_OFSY 0x1f
#define ADXL345_OFSZ 0x20
#define ADXL345_DUR 0x21
#define ADXL345_LATENT 0x22
#define ADXL345_WINDOW 0x23
#define ADXL345_THRESH_ACT 0x24
#define ADXL345_THRESH_INACT 0x25
#define ADXL345_TIME_INACT 0x26
#define ADXL345_ACT_INACT_CTL 0x27
#define ADXL345_THRESH_FF 0x28
#define ADXL345_TIME_FF 0x29
#define ADXL345_TAP_AXES 0x2a
#define ADXL345_ACT_TAP_STATUS 0x2b
#define ADXL345_BW_RATE 0x2c
#define ADXL345_POWER_CTL 0x2d
#define ADXL345_INT_ENABLE 0x2e
#define ADXL345_INT_MAP 0x2f
#define ADXL345_INT_SOURCE 0x30
#define ADXL345_DATA_FORMAT 0x31
#define ADXL345_DATAX0 0x32
#define ADXL345_DATAX1 0x33
#define ADXL345_DATAY0 0x34
#define ADXL345_DATAY1 0x35
#define ADXL345_DATAZ0 0x36
#define ADXL345_DATAZ1 0x37
#define ADXL345_FIFO_CTL 0x38
#define ADXL345_FIFO_STATUS 0x39

#define ADXL345_BW_1600 0xF // 1111
#define ADXL345_BW_800  0xE // 1110
#define ADXL345_BW_400  0xD // 1101
#define ADXL345_BW_200  0xC // 1100
#define ADXL345_BW_100  0xB // 1011
#define ADXL345_BW_50   0xA // 1010
#define ADXL345_BW_25   0x9 // 1001
#define ADXL345_BW_12   0x8 // 1000
#define ADXL345_BW_6    0x7 // 0111
#define ADXL345_BW_3    0x6 // 0110


/*
 Interrupt PINs
 INT1: 0
 INT2: 1
 */
#define ADXL345_INT1_PIN 0x00
#define ADXL345_INT2_PIN 0x01

/*
 Interrupt bit position
 */
#define ADXL345_INT_DATA_READY_BIT 0x07
#define ADXL345_INT_SINGLE_TAP_BIT 0x06
#define ADXL345_INT_DOUBLE_TAP_BIT 0x05
#define ADXL345_INT_ACTIVITY_BIT   0x04
#define ADXL345_INT_INACTIVITY_BIT 0x03
#define ADXL345_INT_FREE_FALL_BIT  0x02
#define ADXL345_INT_WATERMARK_BIT  0x01
#define ADXL345_INT_OVERRUNY_BIT   0x00

#define ADXL345_OK    1 // no error
#define ADXL345_ERROR 0 // indicates error is predent

#define ADXL345_NO_ERROR   0 // initial state
```

```
#define ADXL345_READ_ERROR 1 // problem reading accel
#define ADXL345_BAD_ARG    2 // bad method argument

class ADXL345
{
public:
  bool status;            // set when error occurs
                          // see error code for details
  byte error_code;        // Initial state
  float gains[3];         // counts to Gs

  ADXL345();
  void init(int address);
  void powerOn();
  void readAccel(int* xyx);
  void readAccel(int* x, int* y, int* z);
  void get_Gxyz(float *xyz);

  void setTapThreshold(int tapThreshold);
  int getTapThreshold();
  void setAxisGains(float *_gains);
  void getAxisGains(float *_gains);
  void setAxisOffset(int x, int y, int z);
  void getAxisOffset(int* x, int* y, int*z);
  void setTapDuration(int tapDuration);
  int getTapDuration();
  void setDoubleTapLatency(int floatTapLatency);
  int getDoubleTapLatency();
  void setDoubleTapWindow(int floatTapWindow);
  int getDoubleTapWindow();
  void setActivityThreshold(int activityThreshold);
  int getActivityThreshold();
  void setInactivityThreshold(int inactivityThreshold);
  int getInactivityThreshold();
  void setTimeInactivity(int timeInactivity);
  int getTimeInactivity();
  void setFreeFallThreshold(int freeFallthreshold);
  int getFreeFallThreshold();
  void setFreeFallDuration(int freeFallDuration);
  int getFreeFallDuration();

  bool isActivityXEnabled();
  bool isActivityYEnabled();
  bool isActivityZEnabled();
  bool isInactivityXEnabled();
  bool isInactivityYEnabled();
  bool isInactivityZEnabled();
  bool isActivityAc();
  bool isInactivityAc();
  void setActivityAc(bool state);
  void setInactivityAc(bool state);

  bool getSuppressBit();
  void setSuppressBit(bool state);
  bool isTapDetectionOnX();
  void setTapDetectionOnX(bool state);
  bool isTapDetectionOnY();
  void setTapDetectionOnY(bool state);
  bool isTapDetectionOnZ();
  void setTapDetectionOnZ(bool state);

  void setActivityX(bool state);
  void setActivityY(bool state);
  void setActivityZ(bool state);
  void setInactivityX(bool state);
  void setInactivityY(bool state);
  void setInactivityZ(bool state);

  bool isActivitySourceOnX();
  bool isActivitySourceOnY();
  bool isActivitySourceOnZ();
  bool isTapSourceOnX();
  bool isTapSourceOnY();
  bool isTapSourceOnZ();
  bool isAsleep();

```

```
  bool isLowPower();
  void setLowPower(bool state);
  float getRate();
  void setRate(float rate);
  void set_bw(byte bw_code);
  byte get_bw_code();

  byte getInterruptSource();
  bool getInterruptSource(byte interruptBit);
  bool getInterruptMapping(byte interruptBit);
  void setInterruptMapping(byte interruptBit, bool interruptPin);
  bool isInterruptEnabled(byte interruptBit);
  void setInterrupt(byte interruptBit, bool state);

  void getRangeSetting(byte* rangeSetting);
  void setRangeSetting(int val);
  bool getSelfTestBit();
  void setSelfTestBit(bool selfTestBit);
  bool getSpiBit();
  void setSpiBit(bool spiBit);
  bool getInterruptLevelBit();
  void setInterruptLevelBit(bool interruptLevelBit);
  bool getFullResBit();
  void setFullResBit(bool fullResBit);
  bool getJustifyBit();
  void setJustifyBit(bool justifyBit);
  void printAllRegister();
  void writeTo(byte address, byte val);

private:
  void readFrom(byte address, int num, byte buff[]);
  void setRegisterBit(byte regAdress, int bitPos, bool state);
  bool getRegisterBit(byte regAdress, int bitPos);
  byte _buff[6] ;    //6 bytes buffer for saving data read from the device
  int _dev_address;
};
void print_byte(byte val);
#endif
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FIMU_ADXL345.h

```
/*****************************************************************************
* ITG3200.cpp - ITG-3200/I2C library v0.5 for Arduino                        *
* Copyright 2010-2011 Filipe Vieira & various contributors                   *
* http://code.google.com/p/itg-3200driver                                    *
* This file is part of ITG-3200 Arduino library.                             *
*                                                                            *
* This library is free software: you can redistribute it and/or modify       *
* it under the terms of the GNU Lesser General Public License as published   *
* by the Free Software Foundation, either version 3 of the License, or       *
* (at your option) any later version.                                        *
*                                                                            *
* This program is distributed in the hope that it will be useful,            *
* but WITHOUT ANY WARRANTY; without even the implied warranty of             *
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
* GNU Lesser General Public License for more details.                        *
*                                                                            *
* You should have received a copy of the GNU Lesser General Public License   *
* along with this program.  If not, see <http://www.gnu.org/licenses/>.      *
*****************************************************************************/
/*****************************************************************************
* Tested on Arduino Mega with ITG-3200 Breakout                              *
* SCL     -> pin 21     (no pull up resistors)                               *
* SDA     -> pin 20     (no pull up resistors)                               *
* CLK & GND -> pin GND                                                       *
* INT       -> not connected  (but can be used)                              *
* VIO & VDD -> pin 3.3V                                                      *
*****************************************************************************/
#include "FIMU_ITG3200.h"


ITG3200::ITG3200() {
  setGains(1.0,1.0,1.0);
  setOffsets(0.0,0.0,0.0);
  setRevPolarity(0,0,0);
  //Wire.begin();        //Normally this code is called from setup() at user code
```

```
                        //but some people reported that joining I2C bus earlier
                        //apparently solved problems with master/slave conditions.
                        //Uncomment if needed.
}

void ITG3200::init(unsigned int  address) {
  // Uncomment or change your default ITG3200 initialization

  // fast sample rate - divisor = 0 filter = 0 clocksrc = 0, 1, 2, or 3  (raw values)
  init(address, NOSRDIVIDER, RANGE2000, BW256_SR8, PLL_XGYRO_REF, true, true);

  // slow sample rate - divisor = 0  filter = 1,2,3,4,5, or 6  clocksrc = 0, 1, 2, or 3  (raw values)
  //init(NOSRDIVIDER, RANGE2000, BW010_SR1, INTERNALOSC, true, true);

  // fast sample rate 32Khz external clock - divisor = 0  filter = 0  clocksrc = 4  (raw values)
  //init(NOSRDIVIDER, RANGE2000, BW256_SR8, PLL_EXTERNAL32, true, true);

  // slow sample rate 32Khz external clock - divisor = 0  filter = 1,2,3,4,5, or 6  clocksrc = 4  (raw values)
  //init(NOSRDIVIDER, RANGE2000, BW010_SR1, PLL_EXTERNAL32, true, true);
}

void ITG3200::init(unsigned int address, byte _SRateDiv, byte _Range, byte _filterBW, byte _ClockSrc, bool _ITGReady, bool _INTRawDataReady) {
  _dev_address = address;
  setSampleRateDiv(_SRateDiv);
  setFSRange(_Range);
  setFilterBW(_filterBW);
  setClockSource(_ClockSrc);
  setITGReady(_ITGReady);
  setRawDataReady(_INTRawDataReady);
  delay(GYROSTART_UP_DELAY);  // startup
}

byte ITG3200::getDevAddr() {
  /*readmem(WHO_AM_I, 1, &_buff[0]);
  return _buff[0];  */
  return _dev_address;
}

void ITG3200::setDevAddr(unsigned int  _addr) {
  writemem(WHO_AM_I, _addr);
  _dev_address = _addr;
}

byte ITG3200::getSampleRateDiv() {
  readmem(SMPLRT_DIV, 1, &_buff[0]);
  return _buff[0];
}

void ITG3200::setSampleRateDiv(byte _SampleRate) {
  writemem(SMPLRT_DIV, _SampleRate);
}

byte ITG3200::getFSRange() {
  readmem(DLPF_FS, 1, &_buff[0]);
  return ((_buff[0] & DLPFFS_FS_SEL) >> 3);
}

void ITG3200::setFSRange(byte _Range) {
  readmem(DLPF_FS, 1, &_buff[0]);
  writemem(DLPF_FS, ((_buff[0] & ~DLPFFS_FS_SEL) | (_Range << 3)) );
}

byte ITG3200::getFilterBW() {
  readmem(DLPF_FS, 1, &_buff[0]);
  return (_buff[0] & DLPFFS_DLPF_CFG);
}

void ITG3200::setFilterBW(byte _BW) {
  readmem(DLPF_FS, 1, &_buff[0]);
  writemem(DLPF_FS, ((_buff[0] & ~DLPFFS_DLPF_CFG) | _BW));
}
```

```
108 bool ITG3200::isINTActiveOnLow() {
    readmem(INT_CFG, 1, &_buff[0]);
110   return ((_buff[0] & INTCFG_ACTL) >> 7);
}
112
void ITG3200::setINTLogiclvl(bool _State) {
114   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_ACTL) | (_State << 7)));
116 }

118 bool ITG3200::isINTOpenDrain() {
    readmem(INT_CFG, 1, &_buff[0]);
120   return ((_buff[0] & INTCFG_OPEN) >> 6);
}
122
void ITG3200::setINTDriveType(bool _State) {
124   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_OPEN) | _State << 6));
126 }

128 bool ITG3200::isLatchUntilCleared() {
    readmem(INT_CFG, 1, &_buff[0]);
130   return ((_buff[0] & INTCFG_LATCH_INT_EN) >> 5);
}
132
void ITG3200::setLatchMode(bool _State) {
134   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_LATCH_INT_EN) | _State << 5));
136 }

138 bool ITG3200::isAnyRegClrMode() {
    readmem(INT_CFG, 1, &_buff[0]);
140   return ((_buff[0] & INTCFG_INT_ANYRD_2CLEAR) >> 4);
}
142
void ITG3200::setLatchClearMode(bool _State) {
144   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_INT_ANYRD_2CLEAR) | _State << 4));
146 }

148 bool ITG3200::isITGReadyOn() {
    readmem(INT_CFG, 1, &_buff[0]);
150   return ((_buff[0] & INTCFG_ITG_RDY_EN) >> 2);
}
152
void ITG3200::setITGReady(bool _State) {
154   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_ITG_RDY_EN) | _State << 2));
156 }

158 bool ITG3200::isRawDataReadyOn() {
    readmem(INT_CFG, 1, &_buff[0]);
160   return (_buff[0] & INTCFG_RAW_RDY_EN);
}
162
void ITG3200::setRawDataReady(bool _State) {
164   readmem(INT_CFG, 1, &_buff[0]);
    writememem(INT_CFG, ((_buff[0] & ~INTCFG_RAW_RDY_EN) | _State));
166 }

168 bool ITG3200::isITGReady() {
    readmem(INT_STATUS, 1, &_buff[0]);
170   return ((_buff[0] & INTSTATUS_ITG_RDY) >> 2);
}
172
bool ITG3200::isRawDataReady() {
174   readmem(INT_STATUS, 1, &_buff[0]);
    return (_buff[0] & INTSTATUS_RAW_DATA_RDY);
176 }

178 void ITG3200::readTemp(float *_Temp) {
    readmem(TEMP_OUT,2,_buff);
180   *_Temp = 35 + (((_buff[0] << 8) | _buff[1]) + 13200) / 280.0;     // F=C*9/5+32
}
182
void ITG3200::readGyroRaw(int *_GyroX, int *_GyroY, int *_GyroZ){
```

```
184   readmem(GYRO_XOUT, 6, _buff);
      *_GyroX = ((_buff[0] << 8) | _buff[1]);
186   *_GyroY = ((_buff[2] << 8) | _buff[3]);
      *_GyroZ = ((_buff[4] << 8) | _buff[5]);
188 }

190 void ITG3200::readGyroRaw(int *_GyroXYZ){
      readGyroRaw(_GyroXYZ, _GyroXYZ+1, _GyroXYZ+2);
192 }

194 void ITG3200::setRevPolarity(bool _Xpol, bool _Ypol, bool _Zpol) {
      polarities[0] = _Xpol ? -1 : 1;
196   polarities[1] = _Ypol ? -1 : 1;
      polarities[2] = _Zpol ? -1 : 1;
198 }

200 void ITG3200::setGains(float _Xgain, float _Ygain, float _Zgain) {
      gains[0] = _Xgain;
202   gains[1] = _Ygain;
      gains[2] = _Zgain;
204 }

206 void ITG3200::setOffsets(int _Xoffset, int _Yoffset, int _Zoffset) {
      offsets[0] = _Xoffset;
208   offsets[1] = _Yoffset;
      offsets[2] = _Zoffset;
210 }

212 void ITG3200::zeroCalibrate(unsigned int totSamples, unsigned int sampleDelayMS) {
      int xyz[3];
214   float tmpOffsets[] = {0,0,0};

216   for (int i = 0;i < totSamples;i++){
        delay(sampleDelayMS);
218     readGyroRaw(xyz);
        tmpOffsets[0] += xyz[0];
220     tmpOffsets[1] += xyz[1];
        tmpOffsets[2] += xyz[2];
222   }
      setOffsets(-tmpOffsets[0] / totSamples, -tmpOffsets[1] / totSamples, -tmpOffsets[2]
        / totSamples);
224 }

226 void ITG3200::readGyroRawCal(int *_GyroX, int *_GyroY, int *_GyroZ) {
      readGyroRaw(_GyroX, _GyroY, _GyroZ);
228   *_GyroX += offsets[0];
      *_GyroY += offsets[1];
230   *_GyroZ += offsets[2];
232 }

234 void ITG3200::readGyroRawCal(int *_GyroXYZ) {
      readGyroRawCal(_GyroXYZ, _GyroXYZ+1, _GyroXYZ+2);
236 }

238 void ITG3200::readGyro(float *_GyroX, float *_GyroY, float *_GyroZ){
      int x, y, z;

240   readGyroRawCal(&x, &y, &z); // x,y,z will contain calibrated integer values from
        the sensor
      *_GyroX =  x / 14.375 * polarities[0] * gains[0];
242   *_GyroY =  y / 14.375 * polarities[1] * gains[1];
      *_GyroZ =  z / 14.375 * polarities[2] * gains[2];
244 }

246 void ITG3200::readGyro(float *_GyroXYZ){
      readGyro(_GyroXYZ, _GyroXYZ+1, _GyroXYZ+2);
248 }

250 void ITG3200::reset() {
      writemem(PWR_MGM, PWRMGM_HRESET);
252   delay(GYROSTART_UP_DELAY); //gyro startup
254 }

256 bool ITG3200::isLowPower() {
      readmem(PWR_MGM, 1, &_buff[0]);
      return (_buff[0] & PWRMGM_SLEEP) >> 6;
```

```
}

void ITG3200::setPowerMode(bool _State) {
  readmem(PWR_MGM, 1, &_buff[0]);
  writemem(PWR_MGM, ((_buff[0] & ~PWRMGM_SLEEP) | _State << 6));
}

bool ITG3200::isXgyroStandby() {
  readmem(PWR_MGM, 1, &_buff[0]);
  return (_buff[0] & PWRMGM_STBY_XG) >> 5;
}

bool ITG3200::isYgyroStandby() {
  readmem(PWR_MGM, 1, &_buff[0]);
  return (_buff[0] & PWRMGM_STBY_YG) >> 4;
}

bool ITG3200::isZgyroStandby() {
  readmem(PWR_MGM, 1, &_buff[0]);
  return (_buff[0] & PWRMGM_STBY_ZG) >> 3;
}

void ITG3200::setXgyroStandby(bool _Status) {
  readmem(PWR_MGM, 1, &_buff[0]);
  writemem(PWR_MGM, ((_buff[0] & PWRMGM_STBY_XG) | _Status << 5));
}

void ITG3200::setYgyroStandby(bool _Status) {
  readmem(PWR_MGM, 1, &_buff[0]);
  writemem(PWR_MGM, ((_buff[0] & PWRMGM_STBY_YG) | _Status << 4));
}

void ITG3200::setZgyroStandby(bool _Status) {
  readmem(PWR_MGM, 1, &_buff[0]);
  writemem(PWR_MGM, ((_buff[0] & PWRMGM_STBY_ZG) | _Status << 3));
}

byte ITG3200::getClockSource() {
  readmem(PWR_MGM, 1, &_buff[0]);
  return (_buff[0] & PWRMGM_CLK_SEL);
}

void ITG3200::setClockSource(byte _CLKsource) {
  readmem(PWR_MGM, 1, &_buff[0]);
  writemem(PWR_MGM, ((_buff[0] & ~PWRMGM_CLK_SEL) | _CLKsource));
}

void ITG3200::writemem(uint8_t _addr, uint8_t _val) {
  Wire.beginTransmission(_dev_address);   // start transmission to device
  Wire.write(_addr); // send register address
  Wire.write(_val); // send value to write
  Wire.endTransmission(); // end transmission
}

void ITG3200::readmem(uint8_t _addr, uint8_t _nbytes, uint8_t __buff[]) {
  Wire.beginTransmission(_dev_address); // start transmission to device
  Wire.write(_addr); // sends register address to read from
  Wire.endTransmission(); // end transmission

  Wire.beginTransmission(_dev_address); // start transmission to device
  Wire.requestFrom(_dev_address, _nbytes);// send data n-bytes read
  uint8_t i = 0;
  while (Wire.available()) {
    __buff[i] = Wire.read(); // receive DATA
    i++;
  }
  Wire.endTransmission(); // end transmission
}
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FIMU_ITG3200.cpp

```
/*****************************************************************************
* ITG3200.h - ITG-3200/I2C library v0.5 for Arduino                          *
* Copyright 2010-2011 Filipe Vieira & various contributors                   *
* http://code.google.com/p/itg-3200driver                                    *
* This file is part of ITG-3200 Arduino library.                             *
```

```
6  *                                                                       *
   * This library is free software: you can redistribute it and/or modify  *
8  * it under the terms of the GNU Lesser General Public License as published *
   * by the Free Software Foundation, either version 3 of the License, or  *
10 * (at your option) any later version.                                   *
   *                                                                       *
12 * This program is distributed in the hope that it will be useful,       *
   * but WITHOUT ANY WARRANTY; without even the implied warranty of        *
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
   * GNU Lesser General Public License for more details.                   *
16 *                                                                       *
   * You should have received a copy of the GNU Lesser General Public License *
18 * along with this program.  If not, see <http://www.gnu.org/licenses/>. *
   *************************************************************************/
20 /*************************************************************************
   * Tested on Arduino Mega with ITG-3200 Breakout                         *
22 * SCL     -> pin 21     (no pull up resistors)                          *
   * SDA     -> pin 20     (no pull up resistors)                          *
24 * CLK & GND -> pin GND                                                 *
   * INT       -> not connected  (but can be used)                         *
26 * VIO & VDD -> pin 3.3V                                                *
   *************************************************************************/
28 #ifndef FIMU_ITG3200_h
   #define FIMU_ITG3200_h
30
   #include "Arduino.h"
32 #include <Wire.h>

34 #define ITG3200_ADDR_AD0_HIGH  0x69   //AD0=1 0x69 I2C address when AD0 is connected
       to HIGH (VCC) - default for sparkfun breakout
   #define ITG3200_ADDR_AD0_LOW   0x68   //AD0=0 0x68 I2C address when AD0 is connected
       to LOW (GND)
36 // "The LSB bit of the 7 bit address is determined by the logic level on pin 9.
   // This allows two ITG-3200 devices to be connected to the same I2C bus.
38 // One device should have pin9 (or bit0) LOW and the other should be HIGH." source:
       ITG3200 datasheet
   // Note that pin9 (AD0 - I2C Slave Address LSB) may not be available on some breakout
        boards so check
40 // the schematics of your breakout board for the correct address to use.


42
   #define GYROSTART_UP_DELAY  70    // 50ms from gyro startup + 20ms register r/w
       startup
44
   /* ---- Registers ---- */
46 #define WHO_AM_I           0x00  // RW   SETUP: I2C address
   #define SMPLRT_DIV         0x15  // RW   SETUP: Sample Rate Divider
48 #define DLPF_FS            0x16  // RW   SETUP: Digital Low Pass Filter/ Full Scale
       range
   #define INT_CFG            0x17  // RW   Interrupt: Configuration
50 #define INT_STATUS         0x1A  // R Interrupt: Status
   #define TEMP_OUT           0x1B  // R SENSOR: Temperature 2bytes
52 #define GYRO_XOUT          0x1D  // R SENSOR: Gyro X 2bytes
   #define GYRO_YOUT          0x1F  // R SENSOR: Gyro Y 2bytes
54 #define GYRO_ZOUT          0x21  // R SENSOR: Gyro Z 2bytes
   #define PWR_MGM            0x3E  // RW  Power Management
56
   /* ---- bit maps ---- */
58 #define DLPFFS_FS_SEL              0x18  // 00011000
   #define DLPFFS_DLPF_CFG            0x07  // 00000111
60 #define INTCFG_ACTL                0x80  // 10000000
   #define INTCFG_OPEN                0x40  // 01000000
62 #define INTCFG_LATCH_INT_EN        0x20  // 00100000
   #define INTCFG_INT_ANYRD_2CLEAR    0x10  // 00010000
64 #define INTCFG_ITG_RDY_EN          0x04  // 00000100
   #define INTCFG_RAW_RDY_EN          0x01  // 00000001
66 #define INTSTATUS_ITG_RDY          0x04  // 00000100
   #define INTSTATUS_RAW_DATA_RDY     0x01  // 00000001
68 #define PWRMGM_HRESET              0x80  // 10000000
   #define PWRMGM_SLEEP               0x40  // 01000000
70 #define PWRMGM_STBY_XG             0x20  // 00100000
   #define PWRMGM_STBY_YG             0x10  // 00010000
72 #define PWRMGM_STBY_ZG             0x08  // 00001000
   #define PWRMGM_CLK_SEL             0x07  // 00000111
74
   /**********************************/
```

```
/*     REGISTERS PARAMETERS     */
/************************************/
// Sample Rate Divider
#define NOSRDIVIDER          0 // default     FsampleHz=SampleRateHz/(divider+1)
// Gyro Full Scale Range
#define RANGE2000            3   // default
// Digital Low Pass Filter BandWidth and SampleRate
#define BW256_SR8            0   // default     256Khz BW and 8Khz SR
#define BW188_SR1            1
#define BW098_SR1            2
#define BW042_SR1            3
#define BW020_SR1            4
#define BW010_SR1            5
#define BW005_SR1            6
// Interrupt Active logic lvl
#define ACTIVE_ONHIGH        0 // default
#define ACTIVE_ONLOW         1
// Interrupt drive type
#define PUSH_PULL            0 // default
#define OPEN_DRAIN           1
// Interrupt Latch mode
#define PULSE_50US           0 // default
#define UNTIL_INT_CLEARED    1
// Interrupt Latch clear method
#define READ_STATUSREG       0 // default
#define READ_ANYREG          1
// Power management
#define NORMAL               0 // default
#define STANDBY              1
// Clock Source - user parameters
#define INTERNALOSC          0   // default
#define PLL_XGYRO_REF        1
#define PLL_YGYRO_REF        2
#define PLL_ZGYRO_REF        3
#define PLL_EXTERNAL32       4   // 32.768 kHz
#define PLL_EXTERNAL19       5   // 19.2 Mhz

class ITG3200 {

public:
  float gains[3];
  int offsets[3];
  float polarities[3];

  ITG3200();

  // Gyro initialization
  void init(unsigned int address);
  void init(unsigned int address, byte _SRateDiv, byte _Range, byte _filterBW, byte
    _ClockSrc, bool _ITGReady, bool _INTRawDataReady);

  // Who Am I
  byte getDevAddr();
  void setDevAddr(unsigned int _addr);
  // Sample Rate Divider
  byte getSampleRateDiv();
  void setSampleRateDiv(byte _SampleRate);
  // Digital Low Pass Filter BandWidth and SampleRate
  byte getFSRange();
  void setFSRange(byte _Range); // RANGE2000
  byte getFilterBW();
  void setFilterBW(byte _BW); // see register parameters above
  // Interrupt Configuration
  bool isINTActiveOnLow();
  void setINTLogiclvl(bool _State); //ACTIVE_ONHIGH, ACTIVE_ONLOW
  // Interrupt drive type
  bool isINTOpenDrain();
  void setINTDriveType(bool _State); //OPEN_DRAIN, PUSH_PULL
  // Interrupt Latch mode
  bool isLatchUntilCleared();
  void setLatchMode(bool _State); //UNTIL_INT_CLEARED, PULSE_50US
  // Interrupt Latch clear method
  bool isAnyRegClrMode();
  void setLatchClearMode(bool _State); //READ_ANYREG, READ_STATUSREG
  // INT pin triggers
  bool isITGReadyOn();
```

```
  void setITGReady(bool _State);
  bool isRawDataReadyOn();
  void setRawDataReady(bool _State);
  // Trigger Status
  bool isITGReady();
  bool isRawDataReady();
  // Gyro Sensors
  void readTemp(float *_Temp);
  void readGyroRaw(int  *_GyroXYZ);
  void readGyroRaw(int *_GyroX, int *_GyroY, int *_GyroZ);
  void setRevPolarity(bool _Xpol, bool _Ypol, bool _Zpol);  // true = Reversed  false = default
  void setGains(float _Xgain, float _Ygain, float _Zgain);
  void setOffsets(int _Xoffset, int _Yoffset, int _Zoffset);
  void zeroCalibrate(unsigned int totSamples, unsigned int sampleDelayMS);  // assuming gyroscope is stationary (updates XYZ offsets for zero)
  void readGyroRawCal(int *_GyroX, int *_GyroY, int *_GyroZ);
  void readGyroRawCal(int *_GyroXYZ);
  void readGyro(float *_GyroXYZ); // includes gain and offset
  void readGyro(float *_GyroX, float *_GyroY, float *_GyroZ); // includes gain and offset
  // Power management
  void reset(); // after reset all registers have default values
  bool isLowPower();
  void setPowerMode(bool _State); // NORMAL, STANDBY
  bool isXgyroStandby();
  bool isYgyroStandby();
  bool isZgyroStandby();
  void setXgyroStandby(bool _Status); // NORMAL, STANDBY
  void setYgyroStandby(bool _Status);
  void setZgyroStandby(bool _Status);
  byte getClockSource();
  void setClockSource(byte _CLKsource); // see register parameters above

  void writemem(uint8_t _addr, uint8_t _val);
  void readmem(uint8_t _addr, uint8_t _nbytes, uint8_t __buff[]);

private:

  uint8_t _dev_address;
  uint8_t _buff[6];
};
#endif
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FIMU_ITG3200.h

```
/*
FreeSixIMU.cpp - A libre and easy to use orientation sensing library for Arduino
Copyright (C) 2011 Fabio Varesano <fabio at varesano dot net>

Development of this code has been supported by the Department of Computer Science,
Universita' degli Studi di Torino, Italy within the Piemonte Project
http://www.piemonte.di.unito.it/


This program is free software: you can redistribute it and/or modify
it under the terms of the version 3 GNU General Public License as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

*/

#include <inttypes.h>
//#define DEBUG
#include "FreeSixIMU.h"
// #include "WireUtils.h"
//#include "DebugUtils.h"


```

```
FreeSixIMU::FreeSixIMU() {
  acc = ADXL345();
  gyro = ITG3200();
  //magn = HMC58X3();

  // initialize quaternion
  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;
}

void FreeSixIMU::init() {
  init(FIMU_ACC_ADDR, FIMU_ITG3200_DEF_ADDR, false);
}

void FreeSixIMU::init(bool fastmode) {
  init(FIMU_ACC_ADDR, FIMU_ITG3200_DEF_ADDR, fastmode);
}

void FreeSixIMU::init(int acc_addr, int gyro_addr, bool fastmode) {
  delay(5);

  // disable internal pullups of the ATMEGA which Wire enable by default
  #if defined(__AVR_ATmega168__) || defined(__AVR_ATmega8__) || defined(
    __AVR_ATmega328P__)
    // deactivate internal pull-ups for twi
    // as per note from atmega8 manual pg167
    cbi(PORTC, 4);
    cbi(PORTC, 5);
  #endif
  #if defined(ARDUINO_ARCH_SAM)
  #else
    // deactivate internal pull-ups for twi
    // as per note from atmega128 manual pg204
    cbi(PORTD, 0);
    cbi(PORTD, 1);
  #endif

   if(fastmode) { // switch to 400KHz I2C - eheheh
  #if not defined(ARDUINO_ARCH_SAM)
    TWBR = ((16000000L / 400000L) - 16) / 2; // see twi_init in Wire/utility/twi.c
    // TODO: make the above usable also for 8MHz arduinos..
  #endif
   }

   // init ADXL345
   acc.init(acc_addr);


   // init ITG3200
   gyro.init(gyro_addr);
   delay(1000);
   // calibrate the ITG3200
   gyro.zeroCalibrate(128,5);

   // init HMC5843
   //magn.init(false); // Don't set mode yet, we'll do that later on.
   // Calibrate HMC using self test, not recommended to change the gain after
     calibration.
   //magn.calibrate(1); // Use gain 1=default, valid 0-7, 7 not recommended.
   // Single mode conversion was used in calibration, now set continuous mode
   //magn.setMode(0);
   //delay(10);
   //magn.setDOR(B110);

}
```

```
106 void FreeSixIMU::getRawValues(int * raw_values) {
    acc.readAccel(&raw_values[0], &raw_values[1], &raw_values[2]);
108   gyro.readGyroRaw(&raw_values[3], &raw_values[4], &raw_values[5]);
    //magn.getValues(&raw_values[6], &raw_values[7], &raw_values[8]);
110
  }
112
114 void FreeSixIMU::getValues(float * values) {
    int accval[3];
116   acc.readAccel(&accval[0], &accval[1], &accval[2]);
    values[0] = ((float) accval[0]);
118   values[1] = ((float) accval[1]);
    values[2] = ((float) accval[2]);
120
    gyro.readGyro(&values[3]);
122
    //magn.getValues(&values[6]);
124 }
126
  // Quaternion implementation of the 'DCM filter' [Mayhony et al]. Incorporates the
      magnetic distortion
128 // compensation algorithms from Sebastian Madgwick filter which eliminates the need
      for a reference
  // direction of flux (bx bz) to be predefined and limits the effect of magnetic
      distortions to yaw
130 // axis only.
  //
132 // See: http://www.x-io.co.uk/node/8#open_source_ahrs_and_imu_algorithms
  //
134 //=============================================================================================

  void FreeSixIMU::AHRSupdate(float gx, float gy, float gz, float ax, float ay, float
      az, float mx, float my, float mz) {
136   float recipNorm;
    float q0q0, q0q1, q0q2, q0q3, q1q1, q1q2, q1q3, q2q2, q2q3, q3q3;
138   float halfex = 0.0f, halfey = 0.0f, halfez = 0.0f;
    float qa, qb, qc;
140
    // Auxiliary variables to avoid repeated arithmetic
142   q0q0 = q0 * q0;
    q0q1 = q0 * q1;
144   q0q2 = q0 * q2;
    q0q3 = q0 * q3;
146   q1q1 = q1 * q1;
    q1q2 = q1 * q2;
148   q1q3 = q1 * q3;
    q2q2 = q2 * q2;
150   q2q3 = q2 * q3;
    q3q3 = q3 * q3;
152


154   /*
    // Use magnetometer measurement only when valid (avoids NaN in magnetometer
      normalisation)
156   if((mx != 0.0f) && (my != 0.0f) && (mz != 0.0f)) {
      float hx, hy, bx, bz;
158     float halfwx, halfwy, halfwz;

160     // Normalise magnetometer measurement
      recipNorm = invSqrt(mx * mx + my * my + mz * mz);
162     mx *= recipNorm;
      my *= recipNorm;
164     mz *= recipNorm;

166     // Reference direction of Earth's magnetic field
      hx = 2.0f * (mx * (0.5f - q2q2 - q3q3) + my * (q1q2 - q0q3) + mz * (q1q3 + q0q2))
      ;
168     hy = 2.0f * (mx * (q1q2 + q0q3) + my * (0.5f - q1q1 - q3q3) + mz * (q2q3 - q0q1))
      ;
      bx = sqrt(hx * hx + hy * hy);
170     bz = 2.0f * (mx * (q1q3 - q0q2) + my * (q2q3 + q0q1) + mz * (0.5f - q1q1 - q2q2))
      ;
```

```
172     // Estimated direction of magnetic field
        halfwx = bx * (0.5f - q2q2 - q3q3) + bz * (q1q3 - q0q2);
174     halfwy = bx * (q1q2 - q0q3) + bz * (q0q1 + q2q3);
        halfwz = bx * (q0q2 + q1q3) + bz * (0.5f - q1q1 - q2q2);
176
        // Error is sum of cross product between estimated direction and measured
        direction of field vectors
178     halfex = (my * halfwz - mz * halfwy);
        halfey = (mz * halfwx - mx * halfwz);
180     halfez = (mx * halfwy - my * halfwx);
    }
182 */

184 // Compute feedback only if accelerometer measurement valid (avoids NaN in
      accelerometer normalisation)
    if((ax != 0.0f) && (ay != 0.0f) && (az != 0.0f)) {
186   float halfvx, halfvy, halfvz;

188   // Normalise accelerometer measurement
      recipNorm = invSqrt(ax * ax + ay * ay + az * az);
190   ax *= recipNorm;
      ay *= recipNorm;
192   az *= recipNorm;

194   // Estimated direction of gravity
      halfvx = q1q3 - q0q2;
196   halfvy = q0q1 + q2q3;
      halfvz = q0q0 - 0.5f + q3q3;
198
      // Error is sum of cross product between estimated direction and measured
      direction of field vectors
200   halfex += (ay * halfvz - az * halfvy);
      halfey += (az * halfvx - ax * halfvz);
202   halfez += (ax * halfvy - ay * halfvx);
    }
204
    // Apply feedback only when valid data has been gathered from the accelerometer or
      magnetometer
206 if(halfex != 0.0f && halfey != 0.0f && halfez != 0.0f) {
      // Compute and apply integral feedback if enabled
208   if(twoKi > 0.0f) {
        integralFBx += twoKi * halfex * (1.0f / sampleFreq);  // integral error scaled
      by Ki
210     integralFBy += twoKi * halfey * (1.0f / sampleFreq);
        integralFBz += twoKi * halfez * (1.0f / sampleFreq);
212     gx += integralFBx;  // apply integral feedback
        gy += integralFBy;
214     gz += integralFBz;
      }
216   else {
        integralFBx = 0.0f; // prevent integral windup
218     integralFBy = 0.0f;
        integralFBz = 0.0f;
220   }

222   // Apply proportional feedback
      gx += twoKp * halfex;
224   gy += twoKp * halfey;
      gz += twoKp * halfez;
226 }

228 // Integrate rate of change of quaternion
    gx *= (0.5f * (1.0f / sampleFreq));   // pre-multiply common factors
230 gy *= (0.5f * (1.0f / sampleFreq));
    gz *= (0.5f * (1.0f / sampleFreq));
232 qa = q0;
    qb = q1;
234 qc = q2;
    q0 += (-qb * gx - qc * gy - q3 * gz);
236 q1 += (qa * gx + qc * gz - q3 * gy);
    q2 += (qa * gy - qb * gz + q3 * gx);
238 q3 += (qa * gz + qb * gy - qc * gx);

240 // Normalise quaternion
    recipNorm = invSqrt(q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3);
242 q0 *= recipNorm;
```

```
  q1 *= recipNorm;
  q2 *= recipNorm;
  q3 *= recipNorm;
}


void FreeSixIMU::getQ(float * q) {
  float val[9];
  getValues(val);

  /*
  DEBUG_PRINT(val[3] * M_PI/180);
  DEBUG_PRINT(val[4] * M_PI/180);
  DEBUG_PRINT(val[5] * M_PI/180);
  DEBUG_PRINT(val[0]);
  DEBUG_PRINT(val[1]);
  DEBUG_PRINT(val[2]);
  DEBUG_PRINT(val[6]);
  DEBUG_PRINT(val[7]);
  DEBUG_PRINT(val[8]);
  */


  now = micros();
  sampleFreq = 1.0 / ((now - lastUpdate) / 1000000.0);
  lastUpdate = now;
  // gyro values are expressed in deg/sec, the * M_PI/180 will convert it to radians/sec
  //AHRSupdate(val[3] * M_PI/180, val[4] * M_PI/180, val[5] * M_PI/180, val[0], val[1], val[2], val[6], val[7], val[8]);
  // use the call below when using a 6DOF IMU
  AHRSupdate(val[3] * M_PI/180, val[4] * M_PI/180, val[5] * M_PI/180, val[0], val[1], val[2], 0, 0, 0);
  q[0] = q0;
  q[1] = q1;
  q[2] = q2;
  q[3] = q3;
}

// Returns the Euler angles in radians defined with the Aerospace sequence.
// See Sebastian O.H. Madwick report
// "An efficient orientation filter for inertial and intertial/magnetic sensor arrays" Chapter 2 Quaternion representation
void FreeSixIMU::getEuler(float * angles) {
float q[4]; // quaternion
  getQ(q);
  angles[0] = atan2(2 * q[1] * q[2] - 2 * q[0] * q[3], 2 * q[0]*q[0] + 2 * q[1] * q[1] - 1) * 180/M_PI; // psi
  angles[1] = -asin(2 * q[1] * q[3] + 2 * q[0] * q[2]) * 180/M_PI; // theta
  angles[2] = atan2(2 * q[2] * q[3] - 2 * q[0] * q[1], 2 * q[0] * q[0] + 2 * q[3] * q[3] - 1) * 180/M_PI; // phi
}


void FreeSixIMU::getAngles(float * angles) {
  float a[3]; //Euler
  getEuler(a);

  angles[0] = a[0];
  angles[1] = a[1];
  angles[2] = a[2];

  if(angles[0] < 0)angles[0] += 360;
  if(angles[1] < 0)angles[1] += 360;
  if(angles[2] < 0)angles[2] += 360;

}




void FreeSixIMU::getYawPitchRoll(float * ypr) {
  float q[4]; // quaternion
  float gx, gy, gz; // estimated gravity direction
  getQ(q);
```

```
  gx = 2 * (q[1]*q[3] - q[0]*q[2]);
  gy = 2 * (q[0]*q[1] + q[2]*q[3]);
  gz = q[0]*q[0] - q[1]*q[1] - q[2]*q[2] + q[3]*q[3];

  ypr[0] = atan2(2 * q[1] * q[2] - 2 * q[0] * q[3], 2 * q[0]*q[0] + 2 * q[1] * q[1] -
      1) * 180/M_PI;
  ypr[1] = atan(gx / sqrt(gy*gy + gz*gz))  * 180/M_PI;
  ypr[2] = atan(gy / sqrt(gx*gx + gz*gz))  * 180/M_PI;
}


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;
}
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FreeSixIMU.cpp

```
/*
FreeSixIMU.h - A libre and easy to use orientation sensing library for Arduino
Copyright (C) 2011 Fabio Varesano <fabio at varesano dot net>

Development of this code has been supported by the Department of Computer Science,
Universita' degli Studi di Torino, Italy within the Piemonte Project
http://www.piemonte.di.unito.it/


This program is free software: you can redistribute it and/or modify
it under the terms of the version 3 GNU General Public License as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

*/

#include <Wire.h>
#include "Arduino.h"

#include <FIMU_ADXL345.h>
#define FIMU_ACC_ADDR ADXL345_ADDR_ALT_LOW // SDO connected to GND
#include <FIMU_ITG3200.h>


#ifndef FreeSixIMU_h
#define FreeSixIMU_h


#define FIMU_BMA180_DEF_ADDR BMA180_ADDRESS_SDO_LOW
#define FIMU_ITG3200_DEF_ADDR ITG3200_ADDR_AD0_LOW // AD0 connected to GND
// HMC5843 address is fixed so don't bother to define it


#define twoKpDef  (2.0f * 0.5f) // 2 * proportional gain
#define twoKiDef  (2.0f * 0.1f) // 2 * integral gain

#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif

class FreeSixIMU
```

```
{
  public:
    FreeSixIMU();
    void init();
    void init(bool fastmode);
    void init(int acc_addr, int gyro_addr, bool fastmode);
    void getRawValues(int * raw_values);
    void getValues(float * values);
    void getQ(float * q);
    void getEuler(float * angles);
    void getYawPitchRoll(float * ypr);
    void getAngles(float * angles);


  ADXL345 acc;
    ITG3200 gyro;

    int* raw_acc, raw_gyro, raw_magn;

  private:
    void AHRSupdate(float gx, float gy, float gz, float ax, float ay, float az, float
     mx, float my, float mz);
    //float q0, q1, q2, q3; // quaternion elements representing the estimated
    orientation
    float iq0, iq1, iq2, iq3;
    float exInt, eyInt, ezInt;  // scaled integral error
    volatile float twoKp;      // 2 * proportional gain (Kp)
    volatile float twoKi;      // 2 * integral gain (Ki)
    volatile float q0, q1, q2, q3; // quaternion of sensor frame relative to
    auxiliary frame
    volatile float integralFBx,  integralFBy, integralFBz;
    unsigned long lastUpdate, now; // sample period expressed in milliseconds
    float sampleFreq; // half the sample period expressed in seconds
    int startLoopTime;
};

float invSqrt(float number);

#endif // FreeSixIMU_h
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/FreeSixIMU.h

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
```

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the

product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly

documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will

be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
```

```
    but WITHOUT ANY WARRANTY; without even the implied warranty of
644 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
646
    You should have received a copy of the GNU General Public License
648 along with this program.  If not, see <http://www.gnu.org/licenses/>.

650 Also add information on how to contact you by electronic and paper mail.

652   If the program does terminal interaction, make it output a short
    notice like this when it starts in an interactive mode:
654
    <program>  Copyright (C) <year>  <name of author>
656 This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
    This is free software, and you are welcome to redistribute it
658 under certain conditions; type 'show c' for details.

660 The hypothetical commands 'show w' and 'show c' should show the appropriate
    parts of the General Public License.  Of course, your program's commands
662 might be different; for a GUI interface, you would use an "about box".

664   You should also get your employer (if you work as a programmer) or school,
    if any, to sign a "copyright disclaimer" for the program, if necessary.
666 For more information on this, and how to apply and follow the GNU GPL, see
    <http://www.gnu.org/licenses/>.
668
  The GNU General Public License does not permit incorporating your program
670 into proprietary programs.  If your program is a subroutine library, you
    may consider it more useful to permit linking proprietary applications with
672 the library.  If this is what you want to do, use the GNU Lesser General
    Public License instead of this License.  But first, please read
674 <http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

../Quadcopter/Quadcopter/libraries/FreeSixIMU/LICENSE.txt

```
#include "Arduino.h"
2 #include "Gyro.h"

4 Gyro::Gyro(int pin) {
    _pin = pin;
6 }

8 int PID::read() {
    return analogRead(_pin);
10 }
```

../Quadcopter/Quadcopter/libraries/Gyro/Gyro.cpp

```
class Gyro {
2  public:
    Gyro(int pin);
4   int read();
    void calibrate();
6
  private:
8   int _pin;
};
```

../Quadcopter/Quadcopter/libraries/Gyro/Gyro.h

```
1 Class PID

3 int PID::evaluate(int error)
  diff = _kd * (errorLast - errorNow)/(timeNow - timeLast)
5 return proportional + derivative + integral

7 PID::update(int setpoint)
  PID::updateParameters(int kp, int ki, int kd)
```

../Quadcopter/Quadcopter/libraries/PID/keywords.txt

```
#include "Arduino.h" //Tillgang til Arduinos API, delay, digitalWrite osv.
2 #include "PID.h"

4 PID::PID() {
```

```
  _setpoint = 0;
  _lastError = 0;
  _sumError = 0;
  _lastTime = millis();
  _kp = 1;
  _ki = 1;
  _kd = 1;
}

int PID::evaluate(int value) {
  int error = _setpoint - value;
  int time = millis();

  _sumError+=error;
  int diff = (_lastError - error)/(_lastTime - time);

  _lastError = error;
  _lastTime = time;
  return _kp*error + _ki*_sumError + _kd*diff;
}

void PID::update(int setpoint) {
  _setpoint = setpoint;
}

void PID::updateParameters(float kp, float ki, float kd) {
  _kp = kp;
  _ki = ki;
  _kd = kd;
}
```

../Quadcopter/Quadcopter/libraries/PID/PID.cpp

```
class PID {
 public:
  PID();
  int evaluate(int value);
  void update(int setpoint);
  void updateParameters(float kp, float ki, float kd);

 private:
  int _setpoint;
  int _lastTime;
  int _lastError;
  int _sumError;
  float _kp;
  float _ki;
  float _kd;
};
```

../Quadcopter/Quadcopter/libraries/PID/PID.h