```
////////////////////////////////////////////////////////////////////////////////////
 //Terms of use
 ////////////////////////////////////////////////////////////////////////////////////
 //THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 //IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 //FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 //AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 //LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 //OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 //THE SOFTWARE.
 ////////////////////////////////////////////////////////////////////////////////////
 //Safety note
 ////////////////////////////////////////////////////////////////////////////////////
 //Always remove the propellers and stay away from the motors unless you
 //are 100% certain of what you are doing.
 ////////////////////////////////////////////////////////////////////////////////////

 #include <Wire.h>               //Include the Wire.h library so we can communicate with the gyro
 #include <EEPROM.h>             //Include the EEPROM.h library so we can store information onto the EEPROM

 //Declaring Global Variables
 byte last_channel_1, last_channel_2, last_channel_3, last_channel_4;
 byte lowByte, highByte, type, gyro_address, error, clockspeed_ok;
 byte channel_1_assign, channel_2_assign, channel_3_assign, channel_4_assign;
 byte roll_axis, pitch_axis, yaw_axis;
 byte receiver_check_byte, gyro_check_byte;
 volatile int receiver_input_channel_1, receiver_input_channel_2, receiver_input_channel_3,
receiver_input_channel_4;
 int center_channel_1, center_channel_2, center_channel_3, center_channel_4;
 int high_channel_1, high_channel_2, high_channel_3, high_channel_4;
 int low_channel_1, low_channel_2, low_channel_3, low_channel_4;
 int address, cal_int;
 unsigned long timer, timer_1, timer_2, timer_3, timer_4, current_time;
 float gyro_pitch, gyro_roll, gyro_yaw;
 float gyro_roll_cal, gyro_pitch_cal, gyro_yaw_cal;


 //Setup routine
 void setup(){
   pinMode(12, OUTPUT);
   //Arduino (Atmega) pins default to inputs, so they don't need to be explicitly declared as inputs
   PCICR |= (1 << PCIE0);    // set PCIE0 to enable PCMSK0 scan
   PCMSK0 |= (1 << PCINT0);  // set PCINT0 (digital input 8) to trigger an interrupt on state change
   PCMSK0 |= (1 << PCINT1);  // set PCINT1 (digital input 9)to trigger an interrupt on state change
   PCMSK0 |= (1 << PCINT2);  // set PCINT2 (digital input 10)to trigger an interrupt on state change
   PCMSK0 |= (1 << PCINT3);  // set PCINT3 (digital input 11)to trigger an interrupt on state change
   Wire.begin();             //Start the I2C as master
   Serial.begin(57600);      //Start the serial connetion @ 57600bps
   delay(250);               //Give the gyro time to start
 }
 //Main program
 void loop(){
```

```
//Show the YMFC-3D V2 intro
intro();

Serial.println(F(""));
Serial.println(F("======================================================"));
Serial.println(F("System check"));
Serial.println(F("======================================================"));
delay(1000);
Serial.println(F("Checking I2C clock speed."));
delay(1000);

TWBR = 12;                    //Set the I2C clock speed to 400kHz.

#if F_CPU == 16000000L        //If the clock speed is 16MHz include the next code line when compiling
  clockspeed_ok = 1;          //Set clockspeed_ok to 1
#endif                        //End of if statement

if(TWBR == 12 && clockspeed_ok){
  Serial.println(F("I2C clock speed is correctly set to 400kHz."));
}
else{
  Serial.println(F("I2C clock speed is not set to 400kHz. (ERROR 8)"));
  error = 1;
}

if(error == 0){
  Serial.println(F(""));
  Serial.println(F("======================================================"));
  Serial.println(F("Transmitter setup"));
  Serial.println(F("======================================================"));
  delay(1000);
  Serial.print(F("Checking for valid receiver signals."));
  //Wait 10 seconds until all receiver inputs are valid
  wait_for_receiver();
  Serial.println(F(""));
}
//Quit the program in case of an error
if(error == 0){
  delay(2000);
  Serial.println(F("Place all sticks and subtrims in the center position within 10 seconds."));
  for(int i = 9;i > 0;i--){
    delay(1000);
    Serial.print(i);
    Serial.print(" ");
  }
  Serial.println(" ");
  //Store the central stick positions
  center_channel_1 = receiver_input_channel_1;
  center_channel_2 = receiver_input_channel_2;
  center_channel_3 = receiver_input_channel_3;
  center_channel_4 = receiver_input_channel_4;
  Serial.println(F(""));
```

```
  Serial.println(F("Center positions stored."));
  Serial.print(F("Digital input 08 = "));
  Serial.println(receiver_input_channel_1);
  Serial.print(F("Digital input 09 = "));
  Serial.println(receiver_input_channel_2);
  Serial.print(F("Digital input 10 = "));
  Serial.println(receiver_input_channel_3);
  Serial.print(F("Digital input 11 = "));
  Serial.println(receiver_input_channel_4);
  Serial.println(F(""));
  Serial.println(F(""));
 }
 if(error == 0){
  Serial.println(F("Move the throttle stick to full throttle and back to center"));
  //Check for throttle movement
  check_receiver_inputs(1);
  Serial.print(F("Throttle is connected to digital input "));
  Serial.println((channel_3_assign & 0b00000111) + 7);
  if(channel_3_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
  else Serial.println(F("Channel inverted = no"));
  wait_sticks_zero();

  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Move the roll stick to simulate left wing up and back to center"));
  //Check for throttle movement
  check_receiver_inputs(2);
  Serial.print(F("Roll is connected to digital input "));
  Serial.println((channel_1_assign & 0b00000111) + 7);
  if(channel_1_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
  else Serial.println(F("Channel inverted = no"));
  wait_sticks_zero();
 }
 if(error == 0){
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Move the pitch stick to simulate nose up and back to center"));
  //Check for throttle movement
  check_receiver_inputs(3);
  Serial.print(F("Pitch is connected to digital input "));
  Serial.println((channel_2_assign & 0b00000111) + 7);
  if(channel_2_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
  else Serial.println(F("Channel inverted = no"));
  wait_sticks_zero();
 }
 if(error == 0){
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Move the yaw stick to simulate nose right and back to center"));
  //Check for throttle movement
  check_receiver_inputs(4);
  Serial.print(F("Yaw is connected to digital input "));
```

```
  Serial.println((channel_4_assign & 0b00000111) + 7);
  if(channel_4_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
  else Serial.println(F("Channel inverted = no"));
  wait_sticks_zero();
 }
 if(error == 0){
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Gently move all the sticks simultaneously to their extends"));
  Serial.println(F("When ready put the sticks back in their center positions"));
  //Register the min and max values of the receiver channels
  register_min_max();
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("High, low and center values found during setup"));
  Serial.print(F("Digital input 08 values:"));
  Serial.print(low_channel_1);
  Serial.print(F(" - "));
  Serial.print(center_channel_1);
  Serial.print(F(" - "));
  Serial.println(high_channel_1);
  Serial.print(F("Digital input 09 values:"));
  Serial.print(low_channel_2);
  Serial.print(F(" - "));
  Serial.print(center_channel_2);
  Serial.print(F(" - "));
  Serial.println(high_channel_2);
  Serial.print(F("Digital input 10 values:"));
  Serial.print(low_channel_3);
  Serial.print(F(" - "));
  Serial.print(center_channel_3);
  Serial.print(F(" - "));
  Serial.println(high_channel_3);
  Serial.print(F("Digital input 11 values:"));
  Serial.print(low_channel_4);
  Serial.print(F(" - "));
  Serial.print(center_channel_4);
  Serial.print(F(" - "));
  Serial.println(high_channel_4);
  Serial.println(F("Move stick 'nose up' and back to center to continue"));
  check_to_continue();
 }

 if(error == 0){
  //What gyro is connected
  Serial.println(F(""));
  Serial.println(F("======================================================"));
  Serial.println(F("Gyro search"));
  Serial.println(F("======================================================"));
  delay(2000);

  Serial.println(F("Searching for MPU-6050 on address 0x68/104"));
```

```
delay(1000);
if(search_gyro(0x68, 0x75) == 114){
  Serial.println(F("MPU-6050 found on address 0x68"));
  type = 1;
  gyro_address = 0x68;
}

if(type == 0){
  Serial.println(F("Searching for MPU-6050 on address 0x69/105"));
  delay(1000);
  if(search_gyro(0x69, 0x75) == 0x68){
    Serial.println(F("MPU-6050 found on address 0x69"));
    type = 1;
    gyro_address = 0x69;
  }
}

if(type == 0){
  Serial.println(F("Searching for L3G4200D on address 0x68/104"));
  delay(1000);
  if(search_gyro(0x68, 0x0F) == 0xD3){
    Serial.println(F("L3G4200D found on address 0x68"));
    type = 2;
    gyro_address = 0x68;
  }
}

if(type == 0){
  Serial.println(F("Searching for L3G4200D on address 0x69/105"));
  delay(1000);
  if(search_gyro(0x69, 0x0F) == 0xD3){
    Serial.println(F("L3G4200D found on address 0x69"));
    type = 2;
    gyro_address = 0x69;
  }
}

if(type == 0){
  Serial.println(F("Searching for L3GD20H on address 0x6A/106"));
  delay(1000);
  if(search_gyro(0x6A, 0x0F) == 0xD7){
    Serial.println(F("L3GD20H found on address 0x6A"));
    type = 3;
    gyro_address = 0x6A;
  }
}

if(type == 0){
 Serial.println(F("Searching for L3GD20H on address 0x6B/107"));
  delay(1000);
  if(search_gyro(0x6B, 0x0F) == 0xD7){
    Serial.println(F("L3GD20H found on address 0x6B"));
```

```
      type = 3;
      gyro_address = 0x6B;
    }
  }

  if(type == 0){
    Serial.println(F("No gyro device found!!! (ERROR 3)"));
    error = 1;
  }

  else{
    delay(3000);
    Serial.println(F(""));
    Serial.println(F("======================================================="));
    Serial.println(F("Gyro register settings"));
    Serial.println(F("======================================================="));
    start_gyro(); //Setup the gyro for further use
  }
}

//If the gyro is found we can setup the correct gyro axes.
if(error == 0){
  delay(3000);
  Serial.println(F(""));
  Serial.println(F("======================================================="));
  Serial.println(F("Gyro calibration"));
  Serial.println(F("======================================================="));
  Serial.println(F("Don't move the quadcopter!! Calibration starts in 3 seconds"));
  delay(3000);
  Serial.println(F("Calibrating the gyro, this will take +/- 8 seconds"));
  Serial.print(F("Please wait"));
  //Let's take multiple gyro data samples so we can determine the average gyro offset (calibration).
  for (cal_int = 0; cal_int < 2000 ; cal_int ++){              //Take 2000 readings for calibration.
    if(cal_int % 100 == 0)Serial.print(F("."));                //Print dot to indicate calibration.
    gyro_signalen();                                           //Read the gyro output.
    gyro_roll_cal += gyro_roll;                                //Ad roll value to gyro_roll_cal.
    gyro_pitch_cal += gyro_pitch;                              //Ad pitch value to gyro_pitch_cal.
    gyro_yaw_cal += gyro_yaw;                                  //Ad yaw value to gyro_yaw_cal.
    delay(4);                                                  //Wait 3 milliseconds before the next loop.
  }
  //Now that we have 2000 measures, we need to devide by 2000 to get the average gyro offset.
  gyro_roll_cal /= 2000;                                       //Divide the roll total by 2000.
  gyro_pitch_cal /= 2000;                                      //Divide the pitch total by 2000.
  gyro_yaw_cal /= 2000;                                        //Divide the yaw total by 2000.

  //Show the calibration results
  Serial.println(F(""));
  Serial.print(F("Axis 1 offset="));
  Serial.println(gyro_roll_cal);
  Serial.print(F("Axis 2 offset="));
  Serial.println(gyro_pitch_cal);
  Serial.print(F("Axis 3 offset="));
```

```
Serial.println(gyro_yaw_cal);
Serial.println(F(""));

Serial.println(F("======================================================"));
Serial.println(F("Gyro axes configuration"));
Serial.println(F("======================================================"));

//Detect the left wing up movement
Serial.println(F("Lift the left side of the quadcopter to a 45 degree angle within 10 seconds"));
//Check axis movement
check_gyro_axes(1);
if(error == 0){
  Serial.println(F("OK!"));
  Serial.print(F("Angle detection = "));
  Serial.println(roll_axis & 0b00000011);
  if(roll_axis & 0b10000000)Serial.println(F("Axis inverted = yes"));
  else Serial.println(F("Axis inverted = no"));
  Serial.println(F("Put the quadcopter back in its original position"));
  Serial.println(F("Move stick 'nose up' and back to center to continue"));
  check_to_continue();

  //Detect the nose up movement
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Lift the nose of the quadcopter to a 45 degree angle within 10 seconds"));
  //Check axis movement
  check_gyro_axes(2);
}
if(error == 0){
  Serial.println(F("OK!"));
  Serial.print(F("Angle detection = "));
  Serial.println(pitch_axis & 0b00000011);
  if(pitch_axis & 0b10000000)Serial.println(F("Axis inverted = yes"));
  else Serial.println(F("Axis inverted = no"));
  Serial.println(F("Put the quadcopter back in its original position"));
  Serial.println(F("Move stick 'nose up' and back to center to continue"));
  check_to_continue();

  //Detect the nose right movement
  Serial.println(F(""));
  Serial.println(F(""));
  Serial.println(F("Rotate the nose of the quadcopter 45 degree to the right within 10 seconds"));
  //Check axis movement
  check_gyro_axes(3);
}
if(error == 0){
  Serial.println(F("OK!"));
  Serial.print(F("Angle detection = "));
  Serial.println(yaw_axis & 0b00000011);
  if(yaw_axis & 0b10000000)Serial.println(F("Axis inverted = yes"));
  else Serial.println(F("Axis inverted = no"));
  Serial.println(F("Put the quadcopter back in its original position"));
```

```
    Serial.println(F("Move stick 'nose up' and back to center to continue"));
    check_to_continue();
   }
  }
  if(error == 0){
   Serial.println(F(""));
   Serial.println(F("======================================================"));
   Serial.println(F("LED test"));
   Serial.println(F("======================================================"));
   digitalWrite(12, HIGH);
   Serial.println(F("The LED should now be lit"));
   Serial.println(F("Move stick 'nose up' and back to center to continue"));
   check_to_continue();
   digitalWrite(12, LOW);
  }

  Serial.println(F(""));

  if(error == 0){
   Serial.println(F("======================================================"));
   Serial.println(F("Final setup check"));
   Serial.println(F("======================================================"));
   delay(1000);
   if(receiver_check_byte == 0b00001111){
    Serial.println(F("Receiver channels ok"));
   }
   else{
    Serial.println(F("Receiver channel verification failed!!! (ERROR 6)"));
    error = 1;
   }
   delay(1000);
   if(gyro_check_byte == 0b00000111){
    Serial.println(F("Gyro axes ok"));
   }
   else{
    Serial.println(F("Gyro exes verification failed!!! (ERROR 7)"));
    error = 1;
   }
  }

  if(error == 0){
   //If all is good, store the information in the EEPROM
   Serial.println(F(""));
   Serial.println(F("======================================================"));
   Serial.println(F("Storing EEPROM information"));
   Serial.println(F("======================================================"));
   Serial.println(F("Writing EEPROM"));
   delay(1000);
   Serial.println(F("Done!"));
   EEPROM.write(0, center_channel_1 & 0b11111111);
   EEPROM.write(1, center_channel_1 >> 8);
   EEPROM.write(2, center_channel_2 & 0b11111111);
```

```
EEPROM.write(3, center_channel_2 >> 8);
EEPROM.write(4, center_channel_3 & 0b11111111);
EEPROM.write(5, center_channel_3 >> 8);
EEPROM.write(6, center_channel_4 & 0b11111111);
EEPROM.write(7, center_channel_4 >> 8);
EEPROM.write(8, high_channel_1 & 0b11111111);
EEPROM.write(9, high_channel_1 >> 8);
EEPROM.write(10, high_channel_2 & 0b11111111);
EEPROM.write(11, high_channel_2 >> 8);
EEPROM.write(12, high_channel_3 & 0b11111111);
EEPROM.write(13, high_channel_3 >> 8);
EEPROM.write(14, high_channel_4 & 0b11111111);
EEPROM.write(15, high_channel_4 >> 8);
EEPROM.write(16, low_channel_1 & 0b11111111);
EEPROM.write(17, low_channel_1 >> 8);
EEPROM.write(18, low_channel_2 & 0b11111111);
EEPROM.write(19, low_channel_2 >> 8);
EEPROM.write(20, low_channel_3 & 0b11111111);
EEPROM.write(21, low_channel_3 >> 8);
EEPROM.write(22, low_channel_4 & 0b11111111);
EEPROM.write(23, low_channel_4 >> 8);
EEPROM.write(24, channel_1_assign);
EEPROM.write(25, channel_2_assign);
EEPROM.write(26, channel_3_assign);
EEPROM.write(27, channel_4_assign);
EEPROM.write(28, roll_axis);
EEPROM.write(29, pitch_axis);
EEPROM.write(30, yaw_axis);
EEPROM.write(31, type);
EEPROM.write(32, gyro_address);
//Write the EEPROM signature
EEPROM.write(33, 'J');
EEPROM.write(34, 'M');
EEPROM.write(35, 'B');


//To make sure evrything is ok, verify the EEPROM data.
Serial.println(F("Verify EEPROM data"));
delay(1000);
if(center_channel_1 != ((EEPROM.read(1) << 8) | EEPROM.read(0)))error = 1;
if(center_channel_2 != ((EEPROM.read(3) << 8) | EEPROM.read(2)))error = 1;
if(center_channel_3 != ((EEPROM.read(5) << 8) | EEPROM.read(4)))error = 1;
if(center_channel_4 != ((EEPROM.read(7) << 8) | EEPROM.read(6)))error = 1;

if(high_channel_1 != ((EEPROM.read(9) << 8) | EEPROM.read(8)))error = 1;
if(high_channel_2 != ((EEPROM.read(11) << 8) | EEPROM.read(10)))error = 1;
if(high_channel_3 != ((EEPROM.read(13) << 8) | EEPROM.read(12)))error = 1;
if(high_channel_4 != ((EEPROM.read(15) << 8) | EEPROM.read(14)))error = 1;

if(low_channel_1 != ((EEPROM.read(17) << 8) | EEPROM.read(16)))error = 1;
if(low_channel_2 != ((EEPROM.read(19) << 8) | EEPROM.read(18)))error = 1;
if(low_channel_3 != ((EEPROM.read(21) << 8) | EEPROM.read(20)))error = 1;
```

```
    if(low_channel_4 != ((EEPROM.read(23) << 8) | EEPROM.read(22)))error = 1;

    if(channel_1_assign != EEPROM.read(24))error = 1;
    if(channel_2_assign != EEPROM.read(25))error = 1;
    if(channel_3_assign != EEPROM.read(26))error = 1;
    if(channel_4_assign != EEPROM.read(27))error = 1;

    if(roll_axis != EEPROM.read(28))error = 1;
    if(pitch_axis != EEPROM.read(29))error = 1;
    if(yaw_axis != EEPROM.read(30))error = 1;
    if(type != EEPROM.read(31))error = 1;
    if(gyro_address != EEPROM.read(32))error = 1;

    if('J' != EEPROM.read(33))error = 1;
    if('M' != EEPROM.read(34))error = 1;
    if('B' != EEPROM.read(35))error = 1;

    if(error == 1)Serial.println(F("EEPROM verification failed!!! (ERROR 5)"));
    else Serial.println(F("Verification done"));
   }


  if(error == 0){
    Serial.println(F("Setup is finished."));
    Serial.println(F("You can now calibrate the esc's and upload the YMFC-AL code."));
   }
   else{
    Serial.println(F("The setup is aborted due to an error."));
    Serial.println(F("Check the Q and A page of the YMFC-AL project on:"));
    Serial.println(F("www.brokking.net for more information about this error."));
   }
   while(1);
 }

 //Search for the gyro and check the Who_am_I register
 byte search_gyro(int gyro_address, int who_am_i){
   Wire.beginTransmission(gyro_address);
   Wire.write(who_am_i);
   Wire.endTransmission();
   Wire.requestFrom(gyro_address, 1);
   timer = millis() + 100;
   while(Wire.available() < 1 && timer > millis());
   lowByte = Wire.read();
   address = gyro_address;
   return lowByte;
 }

 void start_gyro(){
   //Setup the L3G4200D or L3GD20H
   if(type == 2 || type == 3){
     Wire.beginTransmission(address);                    //Start communication with the gyro with the address
found during search
```

```
    Wire.write(0x20);                       //We want to write to register 1 (20 hex)
    Wire.write(0x0F);                       //Set the register bits as 00001111 (Turn on the gyro and enable
all axis)
    Wire.endTransmission();                     //End the transmission with the gyro

    Wire.beginTransmission(address);                //Start communication with the gyro (adress 1101001)
    Wire.write(0x20);                       //Start reading @ register 28h and auto increment with every
read
    Wire.endTransmission();                     //End the transmission
    Wire.requestFrom(address, 1);                 //Request 6 bytes from the gyro
    while(Wire.available() < 1);                //Wait until the 1 byte is received
    Serial.print(F("Register 0x20 is set to:"));
    Serial.println(Wire.read(),BIN);

    Wire.beginTransmission(address);                //Start communication with the gyro  with the address
found during search
    Wire.write(0x23);                       //We want to write to register 4 (23 hex)
    Wire.write(0x90);                       //Set the register bits as 10010000 (Block Data Update active &
500dps full scale)
    Wire.endTransmission();                     //End the transmission with the gyro

    Wire.beginTransmission(address);                //Start communication with the gyro (adress 1101001)
    Wire.write(0x23);                       //Start reading @ register 28h and auto increment with every
read
    Wire.endTransmission();                     //End the transmission
    Wire.requestFrom(address, 1);                 //Request 6 bytes from the gyro
    while(Wire.available() < 1);                //Wait until the 1 byte is received
    Serial.print(F("Register 0x23 is set to:"));
    Serial.println(Wire.read(),BIN);

  }
  //Setup the MPU-6050
  if(type == 1){

    Wire.beginTransmission(address);                //Start communication with the gyro
    Wire.write(0x6B);                       //PWR_MGMT_1 register
    Wire.write(0x00);                       //Set to zero to turn on the gyro
    Wire.endTransmission();                     //End the transmission

    Wire.beginTransmission(address);                //Start communication with the gyro
    Wire.write(0x6B);                       //Start reading @ register 28h and auto increment with every
read
    Wire.endTransmission();                     //End the transmission
    Wire.requestFrom(address, 1);                 //Request 1 bytes from the gyro
    while(Wire.available() < 1);                //Wait until the 1 byte is received
    Serial.print(F("Register 0x6B is set to:"));
    Serial.println(Wire.read(),BIN);

    Wire.beginTransmission(address);                //Start communication with the gyro
    Wire.write(0x1B);                       //GYRO_CONFIG register
    Wire.write(0x08);                       //Set the register bits as 00001000 (500dps full scale)
    Wire.endTransmission();                     //End the transmission
```

```
    Wire.beginTransmission(address);                    //Start communication with the gyro (adress 1101001)
    Wire.write(0x1B);                                   //Start reading @ register 28h and auto increment with every
read
    Wire.endTransmission();                             //End the transmission
    Wire.requestFrom(address, 1);                       //Request 1 bytes from the gyro
    while(Wire.available() < 1);                        //Wait until the 1 byte is received
    Serial.print(F("Register 0x1B is set to:"));
    Serial.println(Wire.read(),BIN);

  }
 }

 void gyro_signalen(){
  if(type == 2 || type == 3){
    Wire.beginTransmission(address);                    //Start communication with the gyro
    Wire.write(168);                                    //Start reading @ register 28h and auto increment with every read
    Wire.endTransmission();                             //End the transmission
    Wire.requestFrom(address, 6);                       //Request 6 bytes from the gyro
    while(Wire.available() < 6);                        //Wait until the 6 bytes are received
    lowByte = Wire.read();                              //First received byte is the low part of the angular data
    highByte = Wire.read();                             //Second received byte is the high part of the angular data
    gyro_roll = ((highByte<<8)|lowByte);                //Multiply highByte by 256 (shift left by 8) and ad
lowByte
    if(cal_int == 2000)gyro_roll -= gyro_roll_cal;      //Only compensate after the calibration
    lowByte = Wire.read();                              //First received byte is the low part of the angular data
    highByte = Wire.read();                             //Second received byte is the high part of the angular data
    gyro_pitch = ((highByte<<8)|lowByte);               //Multiply highByte by 256 (shift left by 8) and ad
lowByte
    if(cal_int == 2000)gyro_pitch -= gyro_pitch_cal;    //Only compensate after the calibration
    lowByte = Wire.read();                              //First received byte is the low part of the angular data
    highByte = Wire.read();                             //Second received byte is the high part of the angular data
    gyro_yaw = ((highByte<<8)|lowByte);                 //Multiply highByte by 256 (shift left by 8) and ad
lowByte
    if(cal_int == 2000)gyro_yaw -= gyro_yaw_cal;        //Only compensate after the calibration
  }
  if(type == 1){
    Wire.beginTransmission(address);                    //Start communication with the gyro
    Wire.write(0x43);                                   //Start reading @ register 43h and auto increment with every
read
    Wire.endTransmission();                             //End the transmission
    Wire.requestFrom(address,6);                        //Request 6 bytes from the gyro
    while(Wire.available() < 6);                        //Wait until the 6 bytes are received
    gyro_roll=Wire.read()<<8|Wire.read();               //Read high and low part of the angular data
    if(cal_int == 2000)gyro_roll -= gyro_roll_cal;      //Only compensate after the calibration
    gyro_pitch=Wire.read()<<8|Wire.read();              //Read high and low part of the angular data
    if(cal_int == 2000)gyro_pitch -= gyro_pitch_cal;    //Only compensate after the calibration
    gyro_yaw=Wire.read()<<8|Wire.read();                //Read high and low part of the angular data
    if(cal_int == 2000)gyro_yaw -= gyro_yaw_cal;        //Only compensate after the calibration
  }
 }
```

```
//Check if a receiver input value is changing within 30 seconds
void check_receiver_inputs(byte movement){
  byte trigger = 0;
  int pulse_length;
  timer = millis() + 30000;
  while(timer > millis() && trigger == 0){
    delay(250);
    if(receiver_input_channel_1 > 1750 || receiver_input_channel_1 < 1250){
      trigger = 1;
      receiver_check_byte |= 0b00000001;
      pulse_length = receiver_input_channel_1;
    }
    if(receiver_input_channel_2 > 1750 || receiver_input_channel_2 < 1250){
      trigger = 2;
      receiver_check_byte |= 0b00000010;
      pulse_length = receiver_input_channel_2;
    }
    if(receiver_input_channel_3 > 1750 || receiver_input_channel_3 < 1250){
      trigger = 3;
      receiver_check_byte |= 0b00000100;
      pulse_length = receiver_input_channel_3;
    }
    if(receiver_input_channel_4 > 1750 || receiver_input_channel_4 < 1250){
      trigger = 4;
      receiver_check_byte |= 0b00001000;
      pulse_length = receiver_input_channel_4;
    }
  }
  if(trigger == 0){
    error = 1;
    Serial.println(F("No stick movement detected in the last 30 seconds!!! (ERROR 2)"));
  }
  //Assign the stick to the function.
  else{
    if(movement == 1){
      channel_3_assign = trigger;
      if(pulse_length < 1250)channel_3_assign += 0b10000000;
    }
    if(movement == 2){
      channel_1_assign = trigger;
      if(pulse_length < 1250)channel_1_assign += 0b10000000;
    }
    if(movement == 3){
      channel_2_assign = trigger;
      if(pulse_length < 1250)channel_2_assign += 0b10000000;
    }
    if(movement == 4){
      channel_4_assign = trigger;
      if(pulse_length < 1250)channel_4_assign += 0b10000000;
    }
  }
}
```

```
 void check_to_continue(){
   byte continue_byte = 0;
   while(continue_byte == 0){
     if(channel_2_assign == 0b00000001 && receiver_input_channel_1 > center_channel_1 + 150)continue_byte
= 1;
     if(channel_2_assign == 0b10000001 && receiver_input_channel_1 < center_channel_1 - 150)continue_byte
= 1;
     if(channel_2_assign == 0b00000010 && receiver_input_channel_2 > center_channel_2 + 150)continue_byte
= 1;
     if(channel_2_assign == 0b10000010 && receiver_input_channel_2 < center_channel_2 - 150)continue_byte
= 1;
     if(channel_2_assign == 0b00000011 && receiver_input_channel_3 > center_channel_3 + 150)continue_byte
= 1;
     if(channel_2_assign == 0b10000011 && receiver_input_channel_3 < center_channel_3 - 150)continue_byte
= 1;
     if(channel_2_assign == 0b00000100 && receiver_input_channel_4 > center_channel_4 + 150)continue_byte
= 1;
     if(channel_2_assign == 0b10000100 && receiver_input_channel_4 < center_channel_4 - 150)continue_byte
= 1;
     delay(100);
   }
   wait_sticks_zero();
 }

 //Check if the transmitter sticks are in the neutral position
 void wait_sticks_zero(){
   byte zero = 0;
   while(zero < 15){
     if(receiver_input_channel_1 < center_channel_1 + 20 && receiver_input_channel_1 > center_channel_1 -
20)zero |= 0b00000001;
     if(receiver_input_channel_2 < center_channel_2 + 20 && receiver_input_channel_2 > center_channel_2 -
20)zero |= 0b00000010;
     if(receiver_input_channel_3 < center_channel_3 + 20 && receiver_input_channel_3 > center_channel_3 -
20)zero |= 0b00000100;
     if(receiver_input_channel_4 < center_channel_4 + 20 && receiver_input_channel_4 > center_channel_4 -
20)zero |= 0b00001000;
     delay(100);
   }
 }

 //Checck if the receiver values are valid within 10 seconds
 void wait_for_receiver(){
   byte zero = 0;
   timer = millis() + 10000;
   while(timer > millis() && zero < 15){
     if(receiver_input_channel_1 < 2100 && receiver_input_channel_1 > 900)zero |= 0b00000001;
     if(receiver_input_channel_2 < 2100 && receiver_input_channel_2 > 900)zero |= 0b00000010;
     if(receiver_input_channel_3 < 2100 && receiver_input_channel_3 > 900)zero |= 0b00000100;
     if(receiver_input_channel_4 < 2100 && receiver_input_channel_4 > 900)zero |= 0b00001000;
     delay(500);
     Serial.print(F("."));
```

```
  }
  if(zero == 0){
    error = 1;
    Serial.println(F("."));
    Serial.println(F("No valid receiver signals found!!! (ERROR 1)"));
  }
  else Serial.println(F(" OK"));
}

//Register the min and max receiver values and exit when the sticks are back in the neutral position
void register_min_max(){
  byte zero = 0;
  low_channel_1 = receiver_input_channel_1;
  low_channel_2 = receiver_input_channel_2;
  low_channel_3 = receiver_input_channel_3;
  low_channel_4 = receiver_input_channel_4;
  while(receiver_input_channel_1 < center_channel_1 + 20 && receiver_input_channel_1 > center_channel_1 -
20)delay(250);
  Serial.println(F("Measuring endpoints...."));
  while(zero < 15){
    if(receiver_input_channel_1 < center_channel_1 + 20 && receiver_input_channel_1 > center_channel_1 -
20)zero |= 0b00000001;
    if(receiver_input_channel_2 < center_channel_2 + 20 && receiver_input_channel_2 > center_channel_2 -
20)zero |= 0b00000010;
    if(receiver_input_channel_3 < center_channel_3 + 20 && receiver_input_channel_3 > center_channel_3 -
20)zero |= 0b00000100;
    if(receiver_input_channel_4 < center_channel_4 + 20 && receiver_input_channel_4 > center_channel_4 -
20)zero |= 0b00001000;
    if(receiver_input_channel_1 < low_channel_1)low_channel_1 = receiver_input_channel_1;
    if(receiver_input_channel_2 < low_channel_2)low_channel_2 = receiver_input_channel_2;
    if(receiver_input_channel_3 < low_channel_3)low_channel_3 = receiver_input_channel_3;
    if(receiver_input_channel_4 < low_channel_4)low_channel_4 = receiver_input_channel_4;
    if(receiver_input_channel_1 > high_channel_1)high_channel_1 = receiver_input_channel_1;
    if(receiver_input_channel_2 > high_channel_2)high_channel_2 = receiver_input_channel_2;
    if(receiver_input_channel_3 > high_channel_3)high_channel_3 = receiver_input_channel_3;
    if(receiver_input_channel_4 > high_channel_4)high_channel_4 = receiver_input_channel_4;
    delay(100);
  }
}

//Check if the angular position of a gyro axis is changing within 10 seconds
void check_gyro_axes(byte movement){
  byte trigger_axis = 0;
  float gyro_angle_roll, gyro_angle_pitch, gyro_angle_yaw;
  //Reset all axes
  gyro_angle_roll = 0;
  gyro_angle_pitch = 0;
  gyro_angle_yaw = 0;
  gyro_signalen();
  timer = millis() + 10000;
  while(timer > millis() && gyro_angle_roll > -30 && gyro_angle_roll < 30 && gyro_angle_pitch > -30 &&
gyro_angle_pitch < 30 && gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
```

```
    gyro_signalen();
    if(type == 2 || type == 3){
      gyro_angle_roll += gyro_roll * 0.00007;             //0.00007 = 17.5 (md/s) / 250(Hz)
      gyro_angle_pitch += gyro_pitch * 0.00007;
      gyro_angle_yaw += gyro_yaw * 0.00007;
    }
    if(type == 1){
      gyro_angle_roll += gyro_roll * 0.0000611;          // 0.0000611 = 1 / 65.5 (LSB degr/s) / 250(Hz)
      gyro_angle_pitch += gyro_pitch * 0.0000611;
      gyro_angle_yaw += gyro_yaw * 0.0000611;
    }

    delayMicroseconds(3700); //Loop is running @ 250Hz. +/-300us is used for communication with the gyro
  }
  //Assign the moved axis to the orresponding function (pitch, roll, yaw)
  if((gyro_angle_roll < -30 || gyro_angle_roll > 30) && gyro_angle_pitch > -30 && gyro_angle_pitch < 30 &&
gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
    gyro_check_byte |= 0b00000001;
    if(gyro_angle_roll < 0)trigger_axis = 0b10000001;
    else trigger_axis = 0b00000001;
  }
  if((gyro_angle_pitch < -30 || gyro_angle_pitch > 30) && gyro_angle_roll > -30 && gyro_angle_roll < 30 &&
gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
    gyro_check_byte |= 0b00000010;
    if(gyro_angle_pitch < 0)trigger_axis = 0b10000010;
    else trigger_axis = 0b00000010;
  }
  if((gyro_angle_yaw < -30 || gyro_angle_yaw > 30) && gyro_angle_roll > -30 && gyro_angle_roll < 30 &&
gyro_angle_pitch > -30 && gyro_angle_pitch < 30){
    gyro_check_byte |= 0b00000100;
    if(gyro_angle_yaw < 0)trigger_axis = 0b10000011;
    else trigger_axis = 0b00000011;
  }

  if(trigger_axis == 0){
    error = 1;
    Serial.println(F("No angular motion is detected in the last 10 seconds!!! (ERROR 4)"));
  }
  else
  if(movement == 1)roll_axis = trigger_axis;
  if(movement == 2)pitch_axis = trigger_axis;
  if(movement == 3)yaw_axis = trigger_axis;

}

//This routine is called every time input 8, 9, 10 or 11 changed state
ISR(PCINT0_vect){
  current_time = micros();
  //Channel 1=========================================
  if(PINB & B00000001){                                  //Is input 8 high?
    if(last_channel_1 == 0){                             //Input 8 changed from 0 to 1
      last_channel_1 = 1;                                //Remember current input state
```

```
    timer_1 = current_time;                                  //Set timer_1 to current_time
   }
  }
  else if(last_channel_1 == 1){                              //Input 8 is not high and changed from 1 to 0
   last_channel_1 = 0;                                       //Remember current input state
   receiver_input_channel_1 = current_time - timer_1;        //Channel 1 is current_time - timer_1
  }
  //Channel 2=========================================
  if(PINB & B00000010 ){                                     //Is input 9 high?
   if(last_channel_2 == 0){                                  //Input 9 changed from 0 to 1
    last_channel_2 = 1;                                      //Remember current input state
    timer_2 = current_time;                                  //Set timer_2 to current_time
   }
  }
  else if(last_channel_2 == 1){                              //Input 9 is not high and changed from 1 to 0
   last_channel_2 = 0;                                       //Remember current input state
   receiver_input_channel_2 = current_time - timer_2;        //Channel 2 is current_time - timer_2
  }
  //Channel 3=========================================
  if(PINB & B00000100 ){                                     //Is input 10 high?
   if(last_channel_3 == 0){                                  //Input 10 changed from 0 to 1
    last_channel_3 = 1;                                      //Remember current input state
    timer_3 = current_time;                                  //Set timer_3 to current_time
   }
  }
  else if(last_channel_3 == 1){                              //Input 10 is not high and changed from 1 to 0
   last_channel_3 = 0;                                       //Remember current input state
   receiver_input_channel_3 = current_time - timer_3;        //Channel 3 is current_time - timer_3

  }
  //Channel 4=========================================
  if(PINB & B00001000 ){                                     //Is input 11 high?
   if(last_channel_4 == 0){                                  //Input 11 changed from 0 to 1
    last_channel_4 = 1;                                      //Remember current input state
    timer_4 = current_time;                                  //Set timer_4 to current_time
   }
  }
  else if(last_channel_4 == 1){                              //Input 11 is not high and changed from 1 to 0
   last_channel_4 = 0;                                       //Remember current input state
   receiver_input_channel_4 = current_time - timer_4;        //Channel 4 is current_time - timer_4
  }
}

//Intro subroutine
void intro(){
 Serial.println(F("================================================"));
 delay(1500);
 Serial.println(F(""));
 Serial.println(F("Your"));
 delay(500);
 Serial.println(F("  Multicopter"));
 delay(500);
```

```
  Serial.println(F("    Flight"));
  delay(500);
  Serial.println(F("      Controller"));
  delay(1000);
  Serial.println(F(""));
  Serial.println(F("YMFC-AL Setup Program"));
  Serial.println(F(""));
  Serial.println(F("========================================================"));
  delay(1500);
  Serial.println(F("For support and questions: www.brokking.net"));
  Serial.println(F(""));
  Serial.println(F("Have fun!"));
}
```