```
#include<EEPROM.h>
void setup(){
  pinMode(13,OUTPUT);
  for(int i=0;i<EEPROM.length();i++){
    EEPROM.write(i,0);
  }
 digitalWrite(13,HIGH);
}
void loop(){}
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
#include <Wire.h>
#include <EEPROM.h>
float pid_p_gain_roll = 1.3;
float pid_i_gain_roll = 0.04;
float pid_d_gain_roll = 18.0;
int pid_max_roll = 400;
float pid_p_gain_pitch = pid_p_gain_roll;
float pid_i_gain_pitch = pid_i_gain_roll;
float pid_d_gain_pitch = pid_d_gain_roll;
int pid_max_pitch = pid_max_roll;
float pid_p_gain_yaw = 4.0;
float pid_i_gain_yaw = 0.02;
float pid_d_gain_yaw = 0.0;
int pid_max_yaw = 400;
boolean auto_level = true;
byte last_channel_1, last_channel_2,
last_channel_3, last_channel_4;
byte eeprom_data[36];
byte highByte, lowByte;
volatile int receiver_input_channel_1,
receiver_input_channel_2,
receiver_input_channel_3,
receiver_input_channel_4;
int counter_channel_1, counter_channel_2,
counter_channel_3, counter_channel_4,
loop_counter;
int esc_1, esc_2, esc_3, esc_4;
int throttle, battery_voltage;
int cal_int, start, gyro_address;
int receiver_input[5];
int temperature;
int acc_axis[4], gyro_axis[4];
float roll_level_adjust, pitch_level_adjust;

long acc_x, acc_y, acc_z, acc_total_vector;
unsigned long timer_channel_1,
timer_channel_2, timer_channel_3,
timer_channel_4, esc_timer,
esc_loop_timer;
unsigned long timer_1, timer_2, timer_3,
timer_4, current_time;
unsigned long loop_timer;
double gyro_pitch, gyro_roll, gyro_yaw;
double gyro_axis_cal[4];
float pid_error_temp;
float pid_i_mem_roll, pid_roll_setpoint,
gyro_roll_input, pid_output_roll,
pid_last_roll_d_error;
float pid_i_mem_pitch, pid_pitch_setpoint,
gyro_pitch_input, pid_output_pitch,
pid_last_pitch_d_error;
float pid_i_mem_yaw, pid_yaw_setpoint,
gyro_yaw_input, pid_output_yaw,
pid_last_yaw_d_error;
float angle_roll_acc, angle_pitch_acc,
angle_pitch, angle_roll;
boolean gyro_angles_set;
void setup(){
  for(start = 0; start <= 35;
start++)eeprom_data[start] =
EEPROM.read(start);
  start = 0;
  gyro_address = eeprom_data[32];
  Wire.begin();
  TWBR = 12;
  DDRD |= B11110000;
```

```
  DDRB |= B00110000;
  digitalWrite(12,HIGH);
  while(eeprom_data[33] != 'J' ||
eeprom_data[34] != 'M' || eeprom_data[35]
!= 'B')delay(10);
  if(eeprom_data[31] == 2 ||
eeprom_data[31] == 3)delay(10);
  set_gyro_registers();
  for (cal_int = 0; cal_int < 1250 ; cal_int
++){
    PORTD |= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    delayMicroseconds(3000);
  }
  for (cal_int = 0; cal_int < 2000 ; cal_int
++){
    if(cal_int % 15 == 0)digitalWrite(12,
!digitalRead(12);
    gyro_signalen();
    gyro_axis_cal[1] += gyro_axis[1];
    gyro_axis_cal[2] += gyro_axis[2];
    gyro_axis_cal[3] += gyro_axis[3];
    //We don't want the esc's to be beeping
annoyingly. So let's give them a 1000us puls
while calibrating the gyro.
    PORTD |= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    delay(3);
  }
  gyro_axis_cal[1] /= 2000;
  gyro_axis_cal[2] /= 2000;
  gyro_axis_cal[3] /= 2000;
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT0);
  PCMSK0 |= (1 << PCINT1);
  PCMSK0 |= (1 << PCINT2);
  PCMSK0 |= (1 << PCINT3);
  while(receiver_input_channel_3 < 990 ||
receiver_input_channel_3 > 1020 ||
receiver_input_channel_4 < 1400){
    receiver_input_channel_3 =
convert_receiver_channel(3);
    receiver_input_channel_4 =
convert_receiver_channel(4);
    start ++;
    PORTD |= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    delay(3);
    if(start == 125){
      digitalWrite(12, !digitalRead(12));
      start = 0;
    }
  }
  start = 0;
  battery_voltage = (analogRead(0) + 65) *
1.2317;
  loop_timer = micros();
  digitalWrite(12,LOW);
void loop(){
  gyro_roll_input = (gyro_roll_input * 0.7) +
((gyro_roll / 65.5) * 0.3);
  gyro_pitch_input = (gyro_pitch_input *
0.7) + ((gyro_pitch / 65.5) * 0.3);
gyro_yaw_input = (gyro_yaw_input * 0.7) +
((gyro_yaw / 65.5) * 0.3);
  angle_pitch += gyro_pitch * 0.0000611;
  angle_roll += gyro_roll * 0.0000611;
  angle_pitch -= angle_roll * sin(gyro_yaw *
0.000001066);
  angle_roll += angle_pitch * sin(gyro_yaw
* 0.000001066);
```

```
  acc_total_vector =
sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*
acc_z));
  if(abs(acc_y) < acc_total_vector){
    angle_pitch_acc =
asin((float)acc_y/acc_total_vector)* 57.296;
  }
  if(abs(acc_x) < acc_total_vector){
    angle_roll_acc =
asin((float)acc_x/acc_total_vector)* -
57.296;
  }
  angle_pitch_acc -= 0.0;
  angle_roll_acc -= 0.0;
  angle_pitch = angle_pitch * 0.9996 +
angle_pitch_acc * 0.0004;
  angle_roll = angle_roll * 0.9996 +
angle_roll_acc * 0.0004;
  pitch_level_adjust = angle_pitch * 15;
  roll_level_adjust = angle_roll * 15;
  if(!auto_level){
    pitch_level_adjust = 0;
    roll_level_adjust = 0;
  }
  if(receiver_input_channel_3 < 1050 &&
receiver_input_channel_4 < 1050)start = 1;
  if(start == 1 && receiver_input_channel_3
< 1050 && receiver_input_channel_4 >
1450){
    start = 2;
    angle_pitch = angle_pitch_acc;
    angle_roll = angle_roll_acc;
    gyro_angles_set = true;
    pid_i_mem_roll = 0;
    pid_last_roll_d_error = 0;
    pid_i_mem_pitch = 0;
    pid_last_pitch_d_error = 0;
    pid_i_mem_yaw = 0;
    pid_last_yaw_d_error = 0;
  }
  if(start == 2 && receiver_input_channel_3
< 1050 && receiver_input_channel_4 >
1950)start = 0;
  pid_roll_setpoint = 0;
  if(receiver_input_channel_1 >
1508)pid_roll_setpoint =
receiver_input_channel_1 - 1508;
  else if(receiver_input_channel_1 <
1492)pid_roll_setpoint =
receiver_input_channel_1 - 1492;

  pid_roll_setpoint -= roll_level_adjust;
  pid_roll_setpoint /= 3.0;
  pid_pitch_setpoint = 0;
  if(receiver_input_channel_2 >
1508)pid_pitch_setpoint =
receiver_input_channel_2 - 1508;
  else if(receiver_input_channel_2 <
1492)pid_pitch_setpoint =
receiver_input_channel_2 - 1492;

  pid_pitch_setpoint -= pitch_level_adjust;
  pid_pitch_setpoint /= 3.0;
  pid_yaw_setpoint = 0;
  if(receiver_input_channel_3 > 1050){ //Do
not yaw when turning off the motors.
    if(receiver_input_channel_4 >
1508)pid_yaw_setpoint =
(receiver_input_channel_4 - 1508)/3.0;
    else if(receiver_input_channel_4 <
1492)pid_yaw_setpoint =
(receiver_input_channel_4 - 1492)/3.0;
  }
  calculate_pid();
  battery_voltage = battery_voltage * 0.92 +
(analogRead(0) + 65) * 0.09853;
```

```
  if(battery_voltage < 1000 &&
battery_voltage > 600)digitalWrite(12,
HIGH);
  throttle = receiver_input_channel_3;
  if (start == 2){
    if (throttle > 1800) throttle = 1800;
    esc_1 = throttle - pid_output_pitch +
pid_output_roll - pid_output_yaw;
    esc_2 = throttle + pid_output_pitch +
pid_output_roll + pid_output_yaw;
    esc_3 = throttle + pid_output_pitch -
pid_output_roll - pid_output_yaw;
    esc_4 = throttle - pid_output_pitch -
pid_output_roll + pid_output_yaw;
    if (battery_voltage < 1240 &&
battery_voltage > 800){
      esc_1 += esc_1 * ((1240 -
battery_voltage)/(float)3500);
      esc_2 += esc_2 * ((1240 -
battery_voltage)/(float)3500);
      esc_3 += esc_3 * ((1240 -
battery_voltage)/(float)3500);
      esc_4 += esc_4 * ((1240 -
battery_voltage)/(float)3500);
    }
    if (esc_1 < 1100) esc_1 = 1100;
    if (esc_2 < 1100) esc_2 = 1100;
    if (esc_3 < 1100) esc_3 = 1100;
    if (esc_4 < 1100) esc_4 = 1100;
    if(esc_1 > 2000)esc_1 = 2000;
    if(esc_2 > 2000)esc_2 = 2000;
    if(esc_3 > 2000)esc_3 = 2000;
    if(esc_4 > 2000)esc_4 = 2000;
  }
  else{
    esc_1 = 1000;
    esc_2 = 1000;
    esc_3 = 1000;
    esc_4 = 1000;
  if(micros() - loop_timer >
4050)digitalWrite(12, HIGH);
  while(micros() - loop_timer < 4000);
  loop_timer = micros();
  PORTD |= B11110000;
  timer_channel_1 = esc_1 + loop_timer;
  timer_channel_2 = esc_2 + loop_timer;
  timer_channel_3 = esc_3 + loop_timer;
  timer_channel_4 = esc_4 + loop_timer;
  gyro_signalen();

  while(PORTD >= 16){
    esc_loop_timer = micros();
    if(timer_channel_1 <=
esc_loop_timer)PORTD &= B11101111;
    if(timer_channel_2 <=
esc_loop_timer)PORTD &= B11011111;
    if(timer_channel_3 <=
esc_loop_timer)PORTD &= B10111111;
    if(timer_channel_4 <=
esc_loop_timer)PORTD &= B01111111;
  }
}
ISR(PCINT0_vect){
  current_time = micros();
  //Channel
1================================
==========
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
```

```
    receiver_input[1] = current_time - timer_1;
  }
  //Channel 2==========================
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input[2] = current_time - timer_2;
  }
  //Channel 3==========================
  if(PINB & B00000100 ){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;
    receiver_input[3] = current_time - timer_3;
  }
  //Channel 4==========================
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    receiver_input[4] = current_time - timer_4;
  }
}
void gyro_signalen(){
  //Read the MPU-6050
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address,14);
    receiver_input_channel_1 = convert_receiver_channel(1);
    receiver_input_channel_2 = convert_receiver_channel(2);
    receiver_input_channel_3 = convert_receiver_channel(3);
    receiver_input_channel_4 = convert_receiver_channel(4);
    while(Wire.available() < 14);
    acc_axis[1] = Wire.read()<<8|Wire.read();
    acc_axis[2] = Wire.read()<<8|Wire.read();
    acc_axis[3] = Wire.read()<<8|Wire.read();
    temperature = Wire.read()<<8|Wire.read();
    gyro_axis[1] = Wire.read()<<8|Wire.read();
    gyro_axis[2] = Wire.read()<<8|Wire.read();
    gyro_axis[3] = Wire.read()<<8|Wire.read();
  }
  if(cal_int == 2000){
    gyro_axis[1] -= gyro_axis_cal[1];
```

```
    gyro_axis[2] -= gyro_axis_cal[2];
    gyro_axis[3] -= gyro_axis_cal[3];
  }
  gyro_roll = gyro_axis[eeprom_data[28] &
0b00000011];
  if(eeprom_data[28] &
0b10000000)gyro_roll *= -1;
  gyro_pitch = gyro_axis[eeprom_data[29]
& 0b00000011];
  if(eeprom_data[29] &
0b10000000)gyro_pitch *= -1;
  gyro_yaw = gyro_axis[eeprom_data[30] &
0b00000011];
  if(eeprom_data[30] &
0b10000000)gyro_yaw *= -1;
  acc_x = acc_axis[eeprom_data[29] &
0b00000011];
  if(eeprom_data[29] & 0b10000000)acc_x
*= -1;
  acc_y = acc_axis[eeprom_data[28] &
0b00000011];
  if(eeprom_data[28] & 0b10000000)acc_y
*= -1;
  acc_z = acc_axis[eeprom_data[30] &
0b00000011];
  if(eeprom_data[30] & 0b10000000)acc_z
*= -1;
}
void calculate_pid(){
  //Roll calculations
  pid_error_temp = gyro_roll_input -
pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll *
pid_error_temp;
  if(pid_i_mem_roll >
pid_max_roll)pid_i_mem_roll =
pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -
1)pid_i_mem_roll = pid_max_roll * -1;
  pid_output_roll = pid_p_gain_roll *
pid_error_temp + pid_i_mem_roll +
pid_d_gain_roll * (pid_error_temp -
pid_last_roll_d_error);
  if(pid_output_roll >
pid_max_roll)pid_output_roll =
pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -
1)pid_output_roll = pid_max_roll * -1;
  pid_last_roll_d_error = pid_error_temp;
  pid_error_temp = gyro_pitch_input -
pid_pitch_setpoint;
  pid_i_mem_pitch += pid_i_gain_pitch *
pid_error_temp;
  if(pid_i_mem_pitch >
pid_max_pitch)pid_i_mem_pitch =
pid_max_pitch;
  else if(pid_i_mem_pitch < pid_max_pitch
* -1)pid_i_mem_pitch = pid_max_pitch * -
1;

  pid_output_pitch = pid_p_gain_pitch *
pid_error_temp + pid_i_mem_pitch +
pid_d_gain_pitch * (pid_error_temp -
pid_last_pitch_d_error);
  if(pid_output_pitch >
pid_max_pitch)pid_output_pitch =
pid_max_pitch;
  else if(pid_output_pitch < pid_max_pitch *
-1)pid_output_pitch = pid_max_pitch * -1;

  pid_last_pitch_d_error = pid_error_temp;

  pid_error_temp = gyro_yaw_input -
pid_yaw_setpoint;
```

```
  pid_i_mem_yaw += pid_i_gain_yaw *
pid_error_temp;
  if(pid_i_mem_yaw >
pid_max_yaw)pid_i_mem_yaw =
pid_max_yaw;
  else if(pid_i_mem_yaw < pid_max_yaw *
-1)pid_i_mem_yaw = pid_max_yaw * -1;
  pid_output_yaw = pid_p_gain_yaw *
pid_error_temp + pid_i_mem_yaw +
pid_d_gain_yaw * (pid_error_temp -
pid_last_yaw_d_error);
  if(pid_output_yaw >
pid_max_yaw)pid_output_yaw =
pid_max_yaw;
  else if(pid_output_yaw < pid_max_yaw * -
1)pid_output_yaw = pid_max_yaw * -1;
  pid_last_yaw_d_error = pid_error_temp;
}
int convert_receiver_channel(byte
function){
  byte channel, reverse;
  int low, center, high, actual;
  int difference;
  channel = eeprom_data[function + 23] &
0b00000111;
  if(eeprom_data[function + 23] &
0b10000000)reverse = 1;
  else reverse = 0;
  actual = receiver_input[channel];
  low = (eeprom_data[channel * 2 + 15] <<
8) | eeprom_data[channel * 2 + 14];
  center = (eeprom_data[channel * 2 - 1] <<
8) | eeprom_data[channel * 2 - 2];
  high = (eeprom_data[channel * 2 + 7] <<
8) | eeprom_data[channel * 2 + 6];
  if(actual < center){
    if(actual < low)actual = low;
    difference = ((long)(center - actual) *
(long)500) / (center - low);
    if(reverse == 1)return 1500 + difference;
    else return 1500 - difference;
  }
  else if(actual > center){
    if(actual > high)actual = high;
    difference = ((long)(actual - center) *
(long)500) / (high - center);
    else return 1500 + difference;
  }
  else return 1500;
}
void set_gyro_registers(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1B);
    Wire.write(0x08);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1C);
    Wire.write(0x10);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 1);
    while(Wire.available() < 1);
    if(Wire.read() != 0x08){
      digitalWrite(12,HIGH);
      while(1)delay(10);
    }
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1A);
```

```
    Wire.write(0x03);
    Wire.endTransmission();
  }
}
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^
#include <Wire.h>
#include <EEPROM.h>
byte last_channel_1, last_channel_2,
last_channel_3, last_channel_4;
byte eeprom_data[36], start, data;
boolean new_function_request,first_angle;
volatile int receiver_input_channel_1,
receiver_input_channel_2,
receiver_input_channel_3,
receiver_input_channel_4;
int esc_1, esc_2, esc_3, esc_4;
int counter_channel_1, counter_channel_2,
counter_channel_3, counter_channel_4;
int receiver_input[5];
int loop_counter, gyro_address,
vibration_counter;
int temperature;
long acc_x, acc_y, acc_z,
acc_total_vector[20], acc_av_vector,
vibration_total_result;
unsigned long timer_channel_1,
timer_channel_2, timer_channel_3,
timer_channel_4, esc_timer,
esc_loop_timer;
unsigned long zero_timer, timer_1, timer_2,
timer_3, timer_4, current_time;
int acc_axis[4], gyro_axis[4];
double gyro_pitch, gyro_roll, gyro_yaw;
float angle_roll_acc, angle_pitch_acc,
angle_pitch, angle_roll;
int cal_int;
double gyro_axis_cal[4];
void setup(){
  Serial.begin(57600);
  Wire.begin();
  TWBR = 12;
  DDRD |= B11110000;
  DDRB |= B00010000;
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT0);
  PCMSK0 |= (1 << PCINT1);
  PCMSK0 |= (1 << PCINT2);
  PCMSK0 |= (1 << PCINT3);
  for(data = 0; data <= 35;
data++)eeprom_data[data] =
EEPROM.read(data);
  gyro_address = eeprom_data[32];
  set_gyro_registers();
  //Check the EEPROM signature to make
sure that the setup program is executed.
  while(eeprom_data[33] != 'J' ||
eeprom_data[34] != 'M' || eeprom_data[35]
!= 'B'){
    delay(500);
    digitalWrite(12, !digitalRead(12));
  }
  wait_for_receiver();
  zero_timer = micros();

  while(Serial.available())data =
Serial.read();
  data = 0;
}
void loop(){
  while(zero_timer + 4000 > micros());
  zero_timer = micros();
  if(Serial.available() > 0){
    data = Serial.read();
    delay(100);
```

```
    while(Serial.available() > 0)loop_counter = Serial.read();
    new_function_request = true;
    loop_counter = 0;
    cal_int = 0;
    start = 0;
    first_angle = false;
    if(data == 'r')Serial.println("Reading receiver signals.");
    if(data == 'a')Serial.println("Print the quadcopter angles.");
    if(data == 'a')Serial.println("Gyro calibration starts in 2 seconds (don't move the quadcopter).");
    if(data == '1')Serial.println("Test motor 1 (right front CCW.)");
    if(data == '2')Serial.println("Test motor 2 (right rear CW.)");
    if(data == '3')Serial.println("Test motor 3 (left rear CCW.)");
    if(data == '4')Serial.println("Test motor 4 (left front CW.)");
    if(data == '5')Serial.println("Test all motors together");
    for(vibration_counter = 0; vibration_counter < 625; vibration_counter++){
      delay(3);
      esc_1 = 1000;
      esc_2 = 1000;
      esc_3 = 1000;
      esc_4 = 1000;
      esc_pulse_output();
    }
    vibration_counter = 0;
  }
  receiver_input_channel_3 = convert_receiver_channel(3);
  if(receiver_input_channel_3 < 1025)new_function_request = false;
  if(data == 0 && new_function_request == false){
    receiver_input_channel_3 = convert_receiver_channel(3);
    esc_1 = receiver_input_channel_3;
    esc_2 = receiver_input_channel_3;
    esc_3 = receiver_input_channel_3;
    esc_4 = receiver_input_channel_3;
    esc_pulse_output();
  }
  if(data == 'r'){
    loop_counter ++;
    receiver_input_channel_1 = convert_receiver_channel(1);
    receiver_input_channel_2 = convert_receiver_channel(2);
    receiver_input_channel_3 = convert_receiver_channel(3);
    receiver_input_channel_4 = convert_receiver_channel(4);
    if(loop_counter == 125){
      print_signals();
      loop_counter = 0;
    }
    if(receiver_input_channel_3 < 1050 && receiver_input_channel_4 < 1050)start = 1;
    if(start == 1 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 > 1450)start = 2;
    if(start == 2 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 > 1950)start = 0;
    esc_1 = 1000;
    esc_2 = 1000;
    esc_3 = 1000;
    esc_4 = 1000;
```

```
    esc_pulse_output();
  }
  if(data == '1' || data == '2' || data == '3' ||
data == '4' || data == '5'){
    loop_counter ++;
    if(new_function_request == true &&
loop_counter == 250){
      Serial.print("Set throttle to 1000 (low).
It's now set to: ");

Serial.println(receiver_input_channel_3);
      loop_counter = 0;
    }
    if(new_function_request == false){
      receiver_input_channel_3 =
convert_receiver_channel(3);
      if(data == '1' || data == '5')esc_1 =
receiver_input_channel_3;
      else esc_1 = 1000;
      if(data == '2' || data == '5')esc_2 =
receiver_input_channel_3;
      else esc_2 = 1000;
      if(data == '3' || data == '5')esc_3 =
receiver_input_channel_3;
      else esc_3 = 1000;
      if(data == '4' || data == '5')esc_4 =
receiver_input_channel_3;
      else esc_4 = 1000;
      esc_pulse_output();
      if(eeprom_data[31] == 1){

Wire.beginTransmission(gyro_address);
        Wire.write(0x3B);
        Wire.endTransmission();
        Wire.requestFrom(gyro_address,6);
        while(Wire.available() < 6);
        acc_x = Wire.read()<<8|Wire.read();
        acc_y = Wire.read()<<8|Wire.read();
        acc_z = Wire.read()<<8|Wire.read();
        acc_total_vector[0] =
sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*
acc_z));
        acc_av_vector = acc_total_vector[0];
        for(start = 16; start > 0; start--){
          acc_total_vector[start] =
acc_total_vector[start - 1];
          acc_av_vector +=
acc_total_vector[start];
        }
        acc_av_vector /= 17;
        if(vibration_counter < 20){
          vibration_counter ++;
          vibration_total_result +=
abs(acc_total_vector[0] - acc_av_vector);
}
        else{
          vibration_counter = 0;

Serial.println(vibration_total_result/50);
          vibration_total_result = 0;
        }
      }
    }
  }
  if(data == 'a'){

    if(cal_int != 2000){
      Serial.print("Calibrating the gyro");
      for (cal_int = 0; cal_int < 2000 ; cal_int
++){
        digitalWrite(12, !digitalRead(12));
//Change the led status to indicate
calibration.
        Serial.print(".");
        }
        gyro_signalen();
```

```
      gyro_axis_cal[1] += gyro_axis[1];
      gyro_axis_cal[2] += gyro_axis[2];
      gyro_axis_cal[3] += gyro_axis[3];
      //We don't want the esc's to be beeping annoyingly. So let's give them a 1000us puls while calibrating the gyro.
      PORTD |= B11110000;
      delayMicroseconds(1000);
      PORTD &= B00001111;
      delay(3);
    }
    Serial.println(".");
    //Now that we have 2000 measures, we need to devide by 2000 to get the average gyro offset.
    gyro_axis_cal[1] /= 2000;
    gyro_axis_cal[2] /= 2000;
    gyro_axis_cal[3] /= 2000;
  }
  else{

    PORTD |= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    gyro_signalen();
    angle_pitch += gyro_pitch * 0.0000611;
    angle_roll += gyro_roll * 0.0000611;
    angle_pitch -= angle_roll * sin(gyro_yaw * 0.000001066);
    angle_roll += angle_pitch * sin(gyro_yaw * 0.000001066);
    acc_total_vector[0] = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));
    angle_pitch_acc = asin((float)acc_y/acc_total_vector[0])* 57.296;
    angle_roll_acc = asin((float)acc_x/acc_total_vector[0])* -57.296;
    if(!first_angle){
      angle_pitch = angle_pitch_acc;
      angle_roll = angle_roll_acc;
      first_angle = true;
    }
    else{
      angle_pitch = angle_pitch * 0.9996 + angle_pitch_acc * 0.0004;
      angle_roll = angle_roll * 0.9996 + angle_roll_acc * 0.0004;
    }
    if(loop_counter == 0)Serial.print("Pitch: ");
    if(loop_counter == 1)Serial.print(angle_pitch ,0);
    if(loop_counter == 2)Serial.print(" Roll: ");
    if(loop_counter == 3)Serial.print(angle_roll ,0);
    if(loop_counter == 4)Serial.print(" Yaw: ");
    if(loop_counter == 5)Serial.println(gyro_yaw / 65.5 ,0);
    loop_counter ++;
    if(loop_counter == 60)loop_counter = 0;
   }
  }
}
ISR(PCINT0_vect){
  current_time = micros();
  //Channel 1==================
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
```

```
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time -
timer_1;
  }
  //Channel 2==================
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input[2] = current_time -
timer_2;
  }
  //Channel 3==================
  if(PINB & B00000100 ){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;
    receiver_input[3] = current_time -
timer_3;
  }
  //Channel 4==================
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    receiver_input[4] = current_time -
timer_4;
  }
}
void wait_for_receiver(){
  byte zero = 0;
  while(zero < 15){
    if(receiver_input[1] < 2100 &&
receiver_input[1] > 900)zero |=
0b00000001;
    if(receiver_input[2] < 2100 &&
receiver_input[2] > 900)zero |=
0b00000010;
    if(receiver_input[3] < 2100 &&
receiver_input[3] > 900)zero |=
0b00000100;
    if(receiver_input[4] < 2100 &&
receiver_input[4] > 900)zero |=
0b00001000;
    delay(500);
  }
}
int convert_receiver_channel(byte
function){
  byte channel, reverse;
  int low, center, high, actual;
  int difference;
  channel = eeprom_data[function + 23] &
0b00000111;
  if(eeprom_data[function + 23] &
0b10000000)reverse = 1;
  else reverse = 0;
  actual = receiver_input[channel];
  low = (eeprom_data[channel * 2 + 15] <<
8) | eeprom_data[channel * 2 + 14];
```

```
  center = (eeprom_data[channel * 2 - 1] <<
8) | eeprom_data[channel * 2 - 2];
  high = (eeprom_data[channel * 2 + 7] <<
8) | eeprom_data[channel * 2 + 6];
  if(actual < center){
    if(actual < low)actual = low;
    difference = ((long)(center - actual) *
(long)500) / (center - low);
    if(reverse == 1)return 1500 + difference;
    else return 1500 - difference;
  }
  else if(actual > center){
    if(actual > high)actual = high;
    difference = ((long)(actual - center) *
(long)500) / (high - center);
    if(reverse == 1)return 1500 - difference;
    else return 1500 + difference;
  }
  else return 1500;
}
void print_signals(){
  Serial.print("Start:");
  Serial.print(start);
  Serial.print("  Roll:");
  if(receiver_input_channel_1 - 1480 <
0)Serial.print("<<<");
  else if(receiver_input_channel_1 - 1520 >
0)Serial.print(">>>");
  else Serial.print("-+-");
  Serial.print(receiver_input_channel_1);
  Serial.print("  Pitch:");
  if(receiver_input_channel_2 - 1480 <
0)Serial.print("^^^");
  else if(receiver_input_channel_2 - 1520 >
0)Serial.print("vvv");
  else Serial.print("-+-");
  Serial.print(receiver_input_channel_2);
  Serial.print("  Throttle:");
  if(receiver_input_channel_3 - 1480 <
0)Serial.print("vvv");
  else if(receiver_input_channel_3 - 1520 >
0)Serial.print("^^^");
  else Serial.print("-+-");
  Serial.print(receiver_input_channel_3);
  Serial.print("  Yaw:");
  if(receiver_input_channel_4 - 1480 <
0)Serial.print("<<<");
  else if(receiver_input_channel_4 - 1520 >
0)Serial.print(">>>");
  else Serial.print("-+-");
  Serial.println(receiver_input_channel_4);
}
void esc_pulse_output(){
  zero_timer = micros();
  PORTD |= B11110000;
  timer_channel_1 = esc_1 + zero_timer;
  timer_channel_2 = esc_2 + zero_timer;
  timer_channel_3 = esc_3 + zero_timer;
  timer_channel_4 = esc_4 + zero_timer;
  while(PORTD >= 16){
    esc_loop_timer = micros();
    if(timer_channel_1 <=
esc_loop_timer)PORTD &= B11101111;
    if(timer_channel_2 <=
esc_loop_timer)PORTD &= B11011111;
    if(timer_channel_3 <=
esc_loop_timer)PORTD &= B10111111;
    if(timer_channel_4 <=
esc_loop_timer)PORTD &= B01111111;
}
void set_gyro_registers(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
```

```
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1B);
    Wire.write(0x08);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1C);
    Wire.write(0x10);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 1);
    while(Wire.available() < 1);
    if(Wire.read() != 0x08){
      digitalWrite(12,HIGH);
      while(1)delay(10);
    }
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1A);
    Wire.write(0x03);
    Wire.endTransmission();
  }
}

void gyro_signalen(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address,14);
    while(Wire.available() < 14);
    acc_axis[1] = Wire.read()<<8|Wire.read();
    acc_axis[2] = Wire.read()<<8|Wire.read();
    acc_axis[3] = Wire.read()<<8|Wire.read();
    temperature =
Wire.read()<<8|Wire.read();
    gyro_axis[1] =
Wire.read()<<8|Wire.read();
    gyro_axis[2] =
Wire.read()<<8|Wire.read();
    gyro_axis[3] =
Wire.read()<<8|Wire.read();
  }
  if(cal_int == 2000){
    gyro_axis[1] -= gyro_axis_cal[1];
    gyro_axis[2] -= gyro_axis_cal[2];
    gyro_axis[3] -= gyro_axis_cal[3];
  }
  gyro_roll = gyro_axis[eeprom_data[28] &
0b00000011];
  if(eeprom_data[28] &
0b10000000)gyro_roll *= -1;
  gyro_pitch = gyro_axis[eeprom_data[29]
& 0b00000011];
  if(eeprom_data[29] &
0b10000000)gyro_pitch *= -1;
  gyro_yaw = gyro_axis[eeprom_data[30] &
0b00000011];
  if(eeprom_data[30] &
0b10000000)gyro_yaw *= -1;
  acc_x = acc_axis[eeprom_data[29] &
0b00000011];
  if(eeprom_data[29] & 0b10000000)acc_x
*= -1;
  acc_y = acc_axis[eeprom_data[28] &
0b00000011];
  if(eeprom_data[28] & 0b10000000)acc_y
*= -1;
  acc_z = acc_axis[eeprom_data[30] &
0b00000011];
  if(eeprom_data[30] & 0b10000000)acc_z
*= -1;
}
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^
#include <Wire.h>
```

```
#include <EEPROM.h>
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;
void setup(){
  pinMode(12, OUTPUT);
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT0);
  PCMSK0 |= (1 << PCINT1);
  PCMSK0 |= (1 << PCINT2);
  PCMSK0 |= (1 << PCINT3);
  Wire.begin();
  Serial.begin(57600);
  delay(250);
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;
  #if F_CPU == 16000000L
    clockspeed_ok = 1;
  #endif
  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_for_receiver();
    Serial.println(F(""));
  }
  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) == 0x68){
      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(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"));
   for (cal_int = 0; cal_int < 2000 ; cal_int ++){
    if(cal_int % 100 == 0)Serial.print(F("."));
     gyro_signalen();
     gyro_roll_cal += gyro_roll;
     gyro_pitch_cal += gyro_pitch;
     gyro_yaw_cal += gyro_yaw;
     delay(4);
   }
   //Now that we have 2000 measures, we need to devide by 2000 to get the average gyro offset.
   gyro_roll_cal /= 2000;
   gyro_pitch_cal /= 2000;
   gyro_yaw_cal /= 2000;
   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();
    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');
  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);
}
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(){
 if(type == 2 || type == 3){
  Wire.beginTransmission(address);
  Wire.write(0x20);
  Wire.write(0x0F);
  Wire.endTransmission();
  Wire.beginTransmission(address);
  Wire.write(0x20);
  Wire.endTransmission();
  Wire.requestFrom(address, 1);
  while(Wire.available() < 1);
```

```
  Serial.print(F("Register 0x20 is set to:"));
  Serial.println(Wire.read(),BIN);

  Wire.beginTransmission(address);
  Wire.write(0x23);
  Wire.write(0x90);
  Wire.endTransmission();
  Wire.beginTransmission(address);
  Wire.write(0x23);
  Wire.endTransmission();
  Wire.requestFrom(address, 1);
  while(Wire.available() < 1);
  Serial.print(F("Register 0x23 is set to:"));
  Serial.println(Wire.read(),BIN);
}
if(type == 1){
  Wire.beginTransmission(address);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.beginTransmission(address);
  Wire.write(0x6B);
  Wire.endTransmission();
  Wire.requestFrom(address, 1);
  while(Wire.available() < 1);
  Serial.print(F("Register 0x6B is set to:"));
  Serial.println(Wire.read(),BIN);
  Wire.beginTransmission(address);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();
  Wire.beginTransmission(address);
  Wire.write(0x1B);
  Wire.endTransmission();
  Wire.requestFrom(address, 1);
  while(Wire.available() < 1);
  Serial.print(F("Register 0x1B is set to:"));
  Serial.println(Wire.read(),BIN);
   }
  }
  void gyro_signalen(){
   if(type == 2 || type == 3){
    Wire.beginTransmission(address);
    Wire.write(168);
    Wire.endTransmission();
    Wire.requestFrom(address, 6);
    while(Wire.available() < 6);
    lowByte = Wire.read();
    highByte = Wire.read();
    gyro_roll = ((highByte<<8)|lowByte);
    if(cal_int == 2000)gyro_roll -=
  gyro_roll_cal;
    lowByte = Wire.read();
    highByte = Wire.read();
    gyro_pitch = ((highByte<<8)|lowByte);
    if(cal_int == 2000)gyro_pitch -=
  gyro_pitch_cal;
    lowByte = Wire.read();
    highByte = Wire.read();
    gyro_yaw = ((highByte<<8)|lowByte);
    if(cal_int == 2000)gyro_yaw -=
  gyro_yaw_cal;
   }
   if(type == 1){
    Wire.beginTransmission(address);
    Wire.write(0x43);
    Wire.endTransmission();
    Wire.requestFrom(address,6);
    while(Wire.available() < 6);
    gyro_roll=Wire.read()<<8|Wire.read();
    if(cal_int == 2000)gyro_roll -=
  gyro_roll_cal;
    gyro_pitch=Wire.read()<<8|Wire.read();
    if(cal_int == 2000)gyro_pitch -=
  gyro_pitch_cal;
```

```
    gyro_yaw=Wire.read()<<8|Wire.read();
    if(cal_int == 2000)gyro_yaw -= gyro_yaw_cal;
  }
}
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);
  }
}s
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;
      gyro_angle_pitch += gyro_pitch *
0.00007;
      gyro_angle_yaw += gyro_yaw *
0.00007;
    }
    if(type == 1){
      gyro_angle_roll += gyro_roll *
0.0000611;
      gyro_angle_pitch += gyro_pitch *
0.0000611;
      gyro_angle_yaw += gyro_yaw *
0.0000611;
    }
    delayMicroseconds(3700);
  }
  //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){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input_channel_1 = current_time
- timer_1;
  }
  //Channel 2=========
  if(PINB & B00000010 ){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input_channel_2 = current_time
- timer_2;
  //Channel 3=====
  if(PINB & B00000100 ){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;
    receiver_input_channel_3 = current_time
- timer_3;
  }
  //Channel 4============
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    receiver_input_channel_4 = 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!"));
}
```