Flight Software 3.0

Latest version of the TVC control software and MPU6050 & BMP280 calibration code + interruption overflow detection

Robin Liu 12/2/2021

```
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI*-1);
        #endif



        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
    }
    X08_1.write (ypr[0] * 180/M_PI);
    X08_2.write (ypr[2] * 180/M_PI);
    if(ypr[0] * 180/M_PI > 30 or ypr[0] * 180/M_PI < -30)
    {

      digitalWrite(15, LOW);

    }
    else
    {

      digitalWrite(15, HIGH);
    }

    if(ypr[2] * 180/M_PI*-1> 120 or ypr[2] * 180/M_PI*-1 <
    {

      digitalWrite(16, LOW);


    }
    else
    {

      digitalWrite(16, HIGH);
```

Background

Teensy 4.1 and Arduino

-Microcontroller

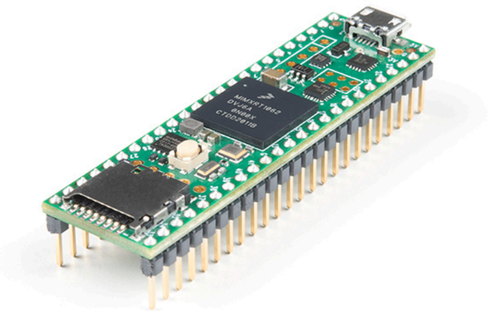
-IDE (Integrated Development Environment)

-Sensors [MPU6050; BMP280]

-PCB (Printable Circuit Board)

-TVC? (Thrust Vector Control)

-How is my code related to these?



Code

Open Source

```
1  // I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)
2  // 6/21/2012 by Jeff Rowberg <jeff@rowberg.net>
3  // Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
4  /* ============================================
5  I2Cdev device library code is placed under the MIT license
6  Copyright (c) 2012 Jeff Rowberg
7
8  Permission is hereby granted, free of charge, to any person obtaining a copy
9  of this software and associated documentation files (the "Software"), to deal
10 in the Software without restriction, including without limitation the rights
11 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
12 copies of the Software, and to permit persons to whom the Software is
13 furnished to do so, subject to the following conditions:
14
15 The above copyright notice and this permission notice shall be included in
16 all copies or substantial portions of the Software.
17
18 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
19 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
20 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
21 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
22 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
23 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
24 THE SOFTWARE.
25 ===============================================
26 */
27
28 // I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
29 // for both classes must be in the include path of your project
30
31 // ===============================================================
```

The date it was updated

THANKS Jeff!

github link

Code

Libraries

```
28 // I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
29 // for both classes must be in the include path of your project
30
31 // ================================================================
32 // ===                      Libraries                           ===
33 // ================================================================
34 #include "I2Cdev.h"
35 #include <Servo.h>
36 #include "MPU6050_6Axis_MotionApps20.h"
37 #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
38 #include "Wire.h"
39 #endif
40
41
42 //#include "MPU6050.h" // not necessary if using MotionApps include file
43 // AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
44 // AD0 high = 0x69
45
```

What is a library???

Code

Data Output

```
63 // uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual
64 // quaternion components in a [w, x, y, z] format (not best for parsing
65 // on a remote host such as Processing or something though)
66 //#define OUTPUT_READABLE_QUATERNION
67
68 // uncomment "OUTPUT_READABLE_EULER" if you want to see Euler angles
69 // (in degrees) calculated from the quaternions coming from the FIFO.
70 // Note that Euler angles suffer from gimbal lock (for more info, see
71 // http://en.wikipedia.org/wiki/Gimbal_lock)
72 //#define OUTPUT_READABLE_EULER
73
74 // uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
75 // pitch/roll angles (in degrees) calculated from the quaternions coming
76 // from the FIFO. Note this also requires gravity vector calculations.
77 // Also note that yaw/pitch/roll angles suffer from gimbal lock (for
78 // more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
79 #define OUTPUT_READABLE_YAWPITCHROLL
80
81 // uncomment "OUTPUT_READABLE_REALACCEL" if you want to see acceleration
82 // components with gravity removed. This acceleration reference frame is
83 // not compensated for orientation, so +X is always +X according to the
84 // sensor, just without the effects of gravity. If you want acceleration
85 // compensated for orientation, us OUTPUT_READABLE_WORLDACCEL instead.
86 #define OUTPUT_READABLE_REALACCEL
87
88 // uncomment "OUTPUT_READABLE_WORLDACCEL" if you want to see acceleration
89 // components with gravity removed and adjusted for the world frame of
90 // reference (yaw is relative to initial orientation, since no magnetomet
91 // is present in this case). Could be quite handy in some cases.
92 //#define OUTPUT_READABLE_WORLDACCEL
93
```

From the library → turn on gyroscope

turn the accelerometer on

Code

Interrupt Detection Routine

```
99  // ================================================================
100 // ===                      INTERRUPT DETECTION ROUTINE           ===
101 // ================================================================
102 #define INTERRUPT_PIN 2
103 #define LED_PIN 13
104 bool blinkState = false;
105
106 // MPU control/status vars
107 bool dmpReady = false;  // set true if DMP init was successful
108 uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
109 uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
110 uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
111 uint16_t fifoCount;     // count of all bytes currently in FIFO
112 uint8_t fifoBuffer[64]; // FIFO storage buffer
113
114 // orientation/motion vars
115 Quaternion q;           // [w, x, y, z]         quaternion container
116 VectorInt16 aa;         // [x, y, z]            accel sensor measurements
117 VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
118 VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
119 VectorFloat gravity;    // [x, y, z]            gravity vector
120 float euler[3];         // [psi, theta, phi]    Euler angle container
121 float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
122
123 // packet structure for InvenSense teapot demo
124 uint8_t teapotPacket[14] = {[redacted]};
125
126 volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
127 void dmpDataReady() {
128     mpuInterrupt = true;
129 }
```

define variables

Boolean

lists!!!

Booleans

Code

Initial Setups

```
132 void setup() {
133 // ================================================================================
134 // ===                          INITIAL SETUP (pin16=GRN; pin15=BLU; pin14=RED)                          ===
135 // ================================================================================
136   pinMode(16, OUTPUT); //turn on GRN
137   pinMode(15, OUTPUT); //turn on BLU
138   //pinMode(14, OUTPUT); //turn on RED
139   pinMode(10, OUTPUT); //turn on buzzer
140   pinMode(LED_PIN, OUTPUT); //turn on teensy LED-pin13
141
142 // ================================================================
143 // ===                          SERVO SETUP                       ===
144 // ================================================================
145   Servo X08_1;     //X SERVO
146   Servo X08_2;     //Y SERVO
147
148   int servo_pin1 = 3; //TVC SERVO X
149   int servo_pin2 = 4; //TVC SERVO Y
150
151   X08_1.attach (servo_pin1); //attach SERVO X to pin3 (X)
152   X08_2.attach (servo_pin2); //attach SERVO Y to pin3 (Y)
```

→ mess around with LED

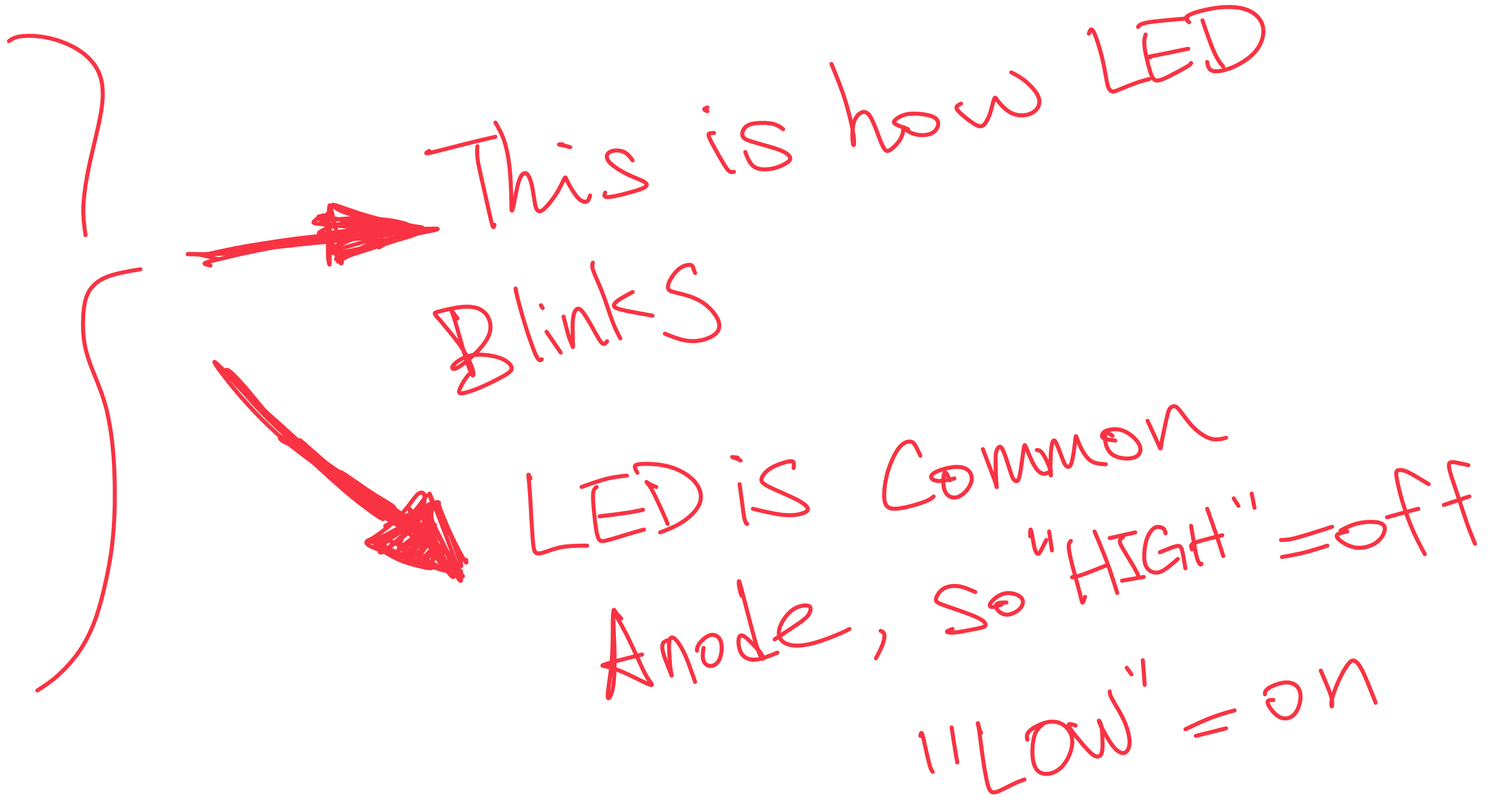
→ define 2 servos on the rocket

→ connect two servos to the predeterminated pin.

Code

System Initiation Indication (SII)

```
154 // ================================================================================
155 // ===                          SYSTEM INITIATION INDICATION                    ===
156 // ================================================================================
157   //GRN Light on for 100
158   digitalWrite(16, LOW);
159   digitalWrite(10, HIGH);
160   delay(100);
161
162   //GRN Light off for 100
163   digitalWrite(16, HIGH);
164   digitalWrite(10, LOW);
165   delay(100);
166
167   //BLU Light on for 100
168   digitalWrite(15, LOW);
169   digitalWrite(10, HIGH);
170   delay(100);
171
172   //BLU Light off for 100
173   digitalWrite(15, HIGH);
174   digitalWrite(10, LOW);
175   delay(100);
176
177   //GRN Light on for 100
178   digitalWrite(16, LOW);
179   digitalWrite(10, HIGH);
180   delay(100);
181
182   //GRN Light off for 100
183   digitalWrite(16, HIGH);
184   digitalWrite(10, LOW);
185   delay(100);
```



Code

Serial Setup

```
// ===============================================================
// ===                      SERIAL SETUP                       ===
// ===============================================================
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot Demo output, but it's
    Serial.begin(115200);
    //while (!Serial); // wait for Leonardo enumeration, others continue immediately

    // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or Arduino
    // Pro Mini running at 3.3V, cannot handle this baud rate reliably due to
    // the baud timing being too misaligned with processor ticks. You must use
    // 38400 or slower in these cases, or use some kind of external separate
    // crystal solution for the UART timer.

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset
    mpu.setYGyroOffset
    mpu.setZGyroOffset
    mpu.setZAccelOffset      ; // 1688 factory default for my test chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // Calibration Time: generate offsets and calibrate our MPU6050
        mpu.CalibrateAccel
        mpu.CalibrateGyro
        mpu.PrintActiveOffsets();
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled
        // enable Arduino interrupt detection
        Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
        Serial.print(digitalPinToInterrupt(
        Serial.println(F(")..."));
        attachInterrupt(digitalPinToInterrupt
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady =
        // get expected DMP packet size for later comparison
        ;
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

// ===============================================================
// ===                  INITIATION COMPLETE INDICATION         ===
// ===============================================================
    digitalWrite(15, LOW); //BLU Light on for 1000
    digitalWrite(10, HIGH); //buzzer on for 1000
    delay(1000);
    digitalWrite(15, HIGH); //BLU Light off
    digitalWrite(10, LOW); //buzzer off
    delay(10);
}
```

clock Speed

Baud Rate

what the serial plotter will see.

off sets for sensors

Blinking LEDs again!

Code

Main Program Loop

```
// ================================================================
// ===                      MAIN PROGRAM LOOP                     ===
// ================================================================

void loop() {
    // if programming failed, don't try to do anything
    if ([redacted] return;

    // wait for MPU interrupt or extra packet(s) available
    while ([redacted]) {
        if [redacted] {
          // try to get out of the infinite loop
          fifoCount = mpu.getFIFOCount();
        }
        // other program behavior stuff here
        // .
        // .
        // .
        // if you are really paranoid you can frequently test in between other
        // stuff to see if mpuInterrupt is true, and if so, "break;" from the
        // while() loop to immediately process the MPU data
        // .
        // .
        // .
    }
    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();

    // otherwise, check for DMP data ready interrupt (this should happen frequently)
    } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;

        #ifdef OUTPUT_READABLE_YAWPITCHROLL
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
            Serial.print("ypr\t");
            Serial.print(ypr[redacted]/M_PI);
            Serial.print("\t");
            Serial.print(ypr[redacted]/M_PI);
            Serial.print("\t");
            Serial.println(ypr[redacted]/M_PI*-1);
        #endif


        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
    }
    X08_1.write (ypr[0] [redacted];
    X08_2.write (ypr[2] [redacted];
    if(ypr[0] * 180/[redacted] ypr[0] [redacted]
    {

      digitalWrite(15, LOW);

    }
    else
    {

      digitalWrite(15, HIGH);
    }

    if(ypr[2] * 180/M_PI*-1> 120 or ypr[2] * 180/M_PI*-1 < 60)
    {

      digitalWrite(16, LOW);


    }
    else
    {

      digitalWrite(16, HIGH);

    }


}
```

if statements

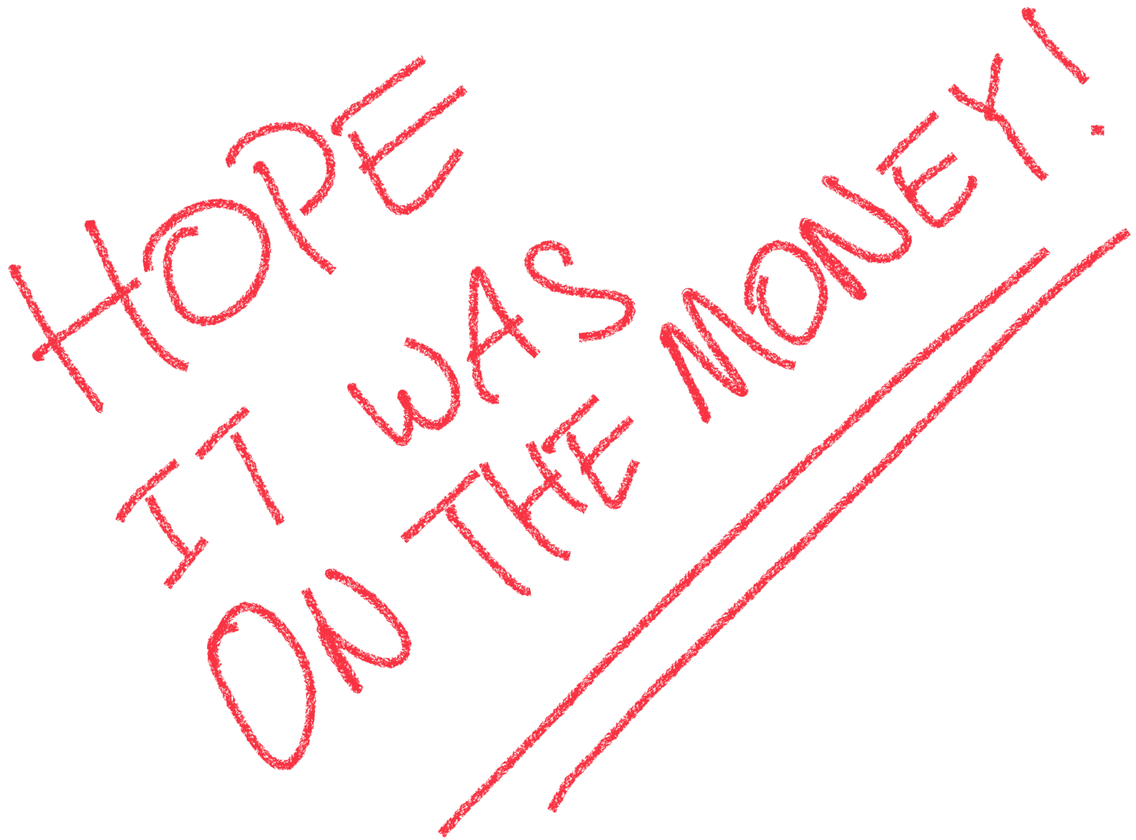
Sry, this is classified.

→ go back to the #2, now print out values

if statements

Yay!

Thanks!
&
Questions?



Robin Liu