

```

1  /* ***** */
2  /* File name:      IMU.h */
3  /* File description: MPU-6050 interface header file */
4  /* Author name:     Giacomo Dollevedo, Gustavo Fernandes */
5  /* Creation date:    18nov2020 */
6  /* Revision date:    12jan2021 */
7  /* ***** */
8
9
10 #ifndef IMU_h
11 #define IMU_h
12
13 //Register Map MPU-6050 https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf
14 #include <Wire.h>
15 #include <Arduino.h>
16 #include <math.h>
17
18 #define I2C_SDA 21
19 #define I2C_SCL 22
20
21
22 #define MPU_ADDR      0x68 // definição do endereço do sensor MPU6050
23 #define WHO_AM_I      0x75 // registro de identificação do dispositivo
24 #define PWR_MGMT_1    0x6B // registro de configuração do gerenciamento de energia
25 #define GYRO_CONFIG    0x1B // registro de configuração do giroscópio
26 #define ACCEL_CONFIG   0x1C // registro de configuração do acelerômetro
27 #define ACCEL_XOUT     0x3B // registro de leitura do eixo X do acelerômetro
28 #define LED_BUILTIN    2    // LED do DevKit v1
29
30 #define CF_GY          0.99 // Fator Gyro filtro complementar
31 #define CF_AC          0.01 // Fator Accel filtro complementar
32
33 #define RAD_2_DEG      57.2958 // Conversao Radianos para Graus
34 #define DEG_2_RAD      0.01745 // Conversao Graus para Radianos
35
36 typedef struct mpu
37 {
38     int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
39 };
40
41 typedef struct processedMpu
42 {
43     float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
44 };
45
46 typedef struct angles
47 {
48     float GyRoll = 0;
49     float GyPitch = 0;
50     float GyYaw = 0;
51     float AclRoll = 0;
52     float AclPitch = 0;
53     float AclYaw = 0;
54 };
55
56 typedef struct processedAngles
57 {

```

```

58     float Roll = 0;
59     float Pitch = 0;
60     float Yaw = 0;
61 };
62
63 typedef struct gyroVel
64 {
65     float Roll = 0;
66     float Pitch = 0;
67     float Yaw = 0;
68 };
69
70 class IMU
71 {
72 public:
73     /*
74     ****
75     */
76     /* Method's name:          initMPU
77     */
78     /* Description:           Initialize I2C bus and MPU-6050
79     */
80     /*
81     */
82     /* Entry parameters:      n/a
83     */
84     /*
85     */
86     /* Return parameters:     n/a
87     */
88     /*
89     ****
90     */
91     void initMPU();
92
93
94     /*
95     ****
96     */
97     /* Method's name:          readRawMPU
98     */
99     /* Description:           Reads all sensor registers from MPU-6050 through I2C bus
100    */
101    /*
102    */
103    /* Entry parameters:      n/a
104    */
105    /*
106    */
107    /* Return parameters:     mpu -> Struct containing raw read values
108    */
109    /*
110    ****
111    */
112    mpu readRawMPU();
113
114
115    /*
116    ****
117    */

```

```

95  /* Method's name:          getData
    */
96  /* Description:           Returns internal processed data struct
    */
97  /*
    */
98  /* Entry parameters:      n/a
    */
99  /*
    */
100 /* Return parameters:      processedMpu -> internal processed data struct
    */
101 /*
    *****
    */
102 processedMpu getData();
103
104
105 /*
    *****
    */
106 /* Method's name:          getRawData
    */
107 /* Description:           Returns internal raw sensor data struct
    */
108 /*
    */
109 /* Entry parameters:
    */
110 /*
    */
111 /* Return parameters:      mpu -> raw data struct
    */
112 /*
    *****
    */
113 mpu getRawData();
114
115
116 /*
    *****
    */
117 /* Method's name:          getRawAngles
    */
118 /* Description:           Returns raw angles from gyro and accelerometer calculation
    */
119 /*
    */
120 /* Entry parameters:      n/a
    */
121 /*
    */
122 /* Return parameters:      angles -> internal processed data struct
    */
123 /*
    *****
    */
124 angles getRawAngles();
125

```

```
126 /*
127  ****
128  */
129 /* Method's name:          getRotations
130  */
131 /* Description:           Returns processed angular displacement after the filter
132  */
133 /*                          on Roll, Pitch and Yaw
134  */
135 /*
136  */
137 /* Entry parameters:       n/a
138  */
139 /*
140  */
141 /* Return parameters:      _procAng -> internal processed angular displacement struct
142  */
143 /*
144  ****
145  */
146 processedAngles getRotations();
147
148
149
150 /*
151  ****
152  */
153 /* Method's name:          getGyroVel
154  */
155 /* Description:           Returns the velocity mean from the gyroscope sensor
156  */
157 /*                          on Roll, Pitch and Yaw
158  */
159 /*
160  */
161 /* Entry parameters:       n/a
162  */
163 /*
164  */
165 /* Return parameters:      gyroVel -> internal mean velocities struct
166  */
167 /*
168  ****
169  */
170 gyroVel getGyroVel();
171
172
173
174 /*
175  ****
176  */
177 /* Method's name:          getPitchVel
178  */
179 /* Description:           Returns gyro pitch velocity after complementary filter
180  */
181 /*
182  */
183 /* Entry parameters:       n/a
184  */
185 /*
186  */
187 /*
188  */
```

```

156 /* Return parameters:      float -> Pitch velocity
    */
157 /*
    ****
    */
158 float getPitchVel();
159
160 /*
    ****
    */
161 /* Method's name:          getRollVel
    */
162 /* Description:            Returns gyro roll velocity after complementary filter
    */
163 /*
    */
164 /* Entry parameters:       n/a
    */
165 /*
    */
166 /* Return parameters:      float -> Roll velocity
    */
167 /*
    ****
    */
168 float getRollVel();
169
170 /*
    ****
    */
171 /* Method's name:          getYawVel
    */
172 /* Description:            Returns gyro yaw velocity after complementary filter
    */
173 /*
    */
174 /* Entry parameters:       n/a
    */
175 /*
    */
176 /* Return parameters:      float -> Yaw velocity
    */
177 /*
    ****
    */
178 float getYawVel();
179
180 /*
    ****
    */
181 /* Method's name:          CalibrateGyro
    */
182 /* Description:            Set gyro calibration values for baseline shift
    */
183 /*
    */
184 /* Entry parameters:       float X -> X axis calibration value
    */

```

```

185 /*                                     float Y -> Y axis calibration value
      */
186 /*                                     float Z -> Z axis calibration value
      */
187 /*
      */
188 /* Return parameters:          n/a
      */
189 /*
      ****
      */
190 void CalibrateGyro(float X, float Y, float Z);
191
192
193 /*
      ****
      */
194 /* Method's name:              CalibrateAcl
      */
195 /* Description:                Set accelerometer calibration values for baseline shift
      */
196 /*
      */
197 /* Entry parameters:          float X -> X axis calibration value
      */
198 /*                             float Y -> Y axis calibration value
      */
199 /*
      */
200 /* Return parameters:          n/a
      */
201 /*
      ****
      */
202 void CalibrateAcl(float X, float Y);
203
204 /*
      ****
      */
205 /* Method's name:              update
      */
206 /* Description:                Reads from MPU-6050 and process data, updating internal
      */
207 /*                             values
      */
208 /*
      */
209 /* Entry parameters:          n/a
      */
210 /*
      */
211 /* Return parameters:          n/a
      */
212 /*
      ****
      */
213 void update();
214
215

```

```

216 /*
*****
*/
217 /* Method's name:          enableDebug
   */
218 /* Description:            Enables serial communication for debbuging
   */
219 /*
   */
220 /* Entry parameters:       n/a
   */
221 /*
   */
222 /* Return parameters:      n/a
   */
223 /*
*****
*/
224 void enableDebug();
225
226
227 /*
*****
*/
228 /* Method's name:          disableDebug
   */
229 /* Description:            Disables serial communication for debbuging
   */
230 /*
   */
231 /* Entry parameters:       n/a
   */
232 /*
   */
233 /* Return parameters:      n/a
   */
234 /*
*****
*/
235 void disableDebug();
236
237 /*
*****
*/
238 /* Method's name:          disableYawComp
   */
239 /* Description:            Disables roll and pitch angle compensation using yaw
   */
240 /*
   */
241 /* Entry parameters:       n/a
   */
242 /*
   */
243 /* Return parameters:      n/a
   */
244 /*
*****
*/

```

```

245 void disableYawComp();
246
247
248 /*
249  *
250  * Method's name:          enableYawComp
251  *
252  * Description:           Enables roll and pitch angle compensation using yaw
253  *
254  * Entry parameters:      n/a
255  *
256  * Return parameters:     n/a
257  *
258  */
259
260 void enableYawComp();
261
262 private:
263
264 /*
265  *
266  * Method's name:          writeRegMPU
267  *
268  * Description:           Writes to a MPU-6005 register through I2C bus
269  *
270  * Entry parameters:      int reg -> Register to write to
271  *
272  *                        int val -> Value to write
273  *
274  * Return parameters:     n/a
275  *
276  */
277
278 void writeRegMPU(int reg, int val);
279
280
281 /*
282  *
283  * Method's name:          readRegMPU
284  *
285  * Description:           Reads from a MPU-6005 register through I2C bus
286  *
287  * Entry parameters:      unsigned char reg -> Register to read from
288  *
289  */

```



```

277 /*
    */
278 /* Return parameters:      unsigned char -> value that was read
    */
279 /*
    *****
    */
280 unsigned char readRegMPU(unsigned char reg);
281
282
283 /*
    *****
    */
284 /* Method's name:          findMPU
    */
285 /* Description:            Check for MPU-6050 address on I2C bus
    */
286 /*
    */
287 /* Entry parameters:       n/a
    */
288 /*
    */
289 /* Return parameters:      unsigned char -> 0 == not found / 1 == found
    */
290 /*
    *****
    */
291 unsigned char findMPU();
292
293
294 /*
    *****
    */
295 /* Method's name:          checkMPU
    */
296 /* Description:            Check MPU-6050 status through I2C bus
    */
297 /*
    */
298 /* Entry parameters:       n/a
    */
299 /*
    */
300 /* Return parameters:      unsigned char -> 0 = not available / 1 = Active / 2 =
Sleep */
301 /*
    *****
    */
302 unsigned char checkMPU();
303
304
305 /*
    *****
    */
306 /* Method's name:          filterMPUData
    */
307 /* Description:            Complementary filter to keep angular displacement from
    */

```

```

308 /* drifting
    */
309 /*
    */
310 /* Entry parameters:      n/a
    */
311 /*
    */
312 /* Return parameters:      n/a
    */
313 /*
    ****
    */
314 void filterMPUData();
315
316
317 /*
    ****
    */
318 /* Method's name:          processMPUData
    */
319 /* Description:             Converts raw data to actual values. Also finds angular
    */
320 /* displacement
    */
321 /*
    */
322 /* Entry parameters:        n/a
    */
323 /*
    */
324 /* Return parameters:       processedMpu -> processed data struct
    */
325 /*
    ****
    */
326 processedMpu processMPUData();
327
328
329 /*
    ****
    */
330 /* Method's name:          processAngles
    */
331 /* Description:             Converts gyro and accel data into angular displacement
    */
332 /*
    */
333 /* Entry parameters:        processedMpu dados -> data struct to process
    */
334 /*
    */
335 /* Return parameters:        n/a
    */
336 /*
    ****
    */
337 void processAngles(processedMpu dados);
338

```

```

339 /*
340 *****
341 */
342 /* Method's name:          setSleepOff
343 */
344 /* Description:           Writes to specific register on MPU-6050 to set Active Mode
345 */
346 /* Entry parameters:      n/a
347 */
348 /* Return parameters:     n/a
349 */
350 void setSleepOff();
351
352 /*
353 *****
354 */
355 /* Method's name:          setGyroScale
356 */
357 /* Description:           Set gyroscope scale to +- 250°/s
358 */
359 /* Entry parameters:      n/a
360 */
361 /* Return parameters:     n/a
362 */
363 void setGyroScale();
364
365 /*
366 *****
367 */
368 /* Method's name:          setAccelScale
369 */
370 /* Description:           Set accelerometer scale to +- 2g
371 */
372 /* Entry parameters:      n/a
373 */
374 /* Return parameters:     n/a
375 */

```

```

369 /*
*****
*/
370 void setAccelScale();
371
372
373 mpu _rawData;
374 processedMpu _processedData;
375
376 float calGyX = 0;
377 float calGyY = 0;
378 float calGyZ = 0;
379
380 float calAcX = 0;
381 float calAcY = 0;
382
383
384 float _gyroRollInput = 0;
385 float _gyroPitchInput = 0;
386 float _gyroYawInput = 0;
387
388
389 unsigned long _lastTimestamp = 0;
390
391 angles _ang;
392 processedAngles _procAng;
393
394 unsigned char led_state = 0;
395 unsigned char yaw_compensation = 1;
396 unsigned char debbuging_enabled = 1;
397
398 /*NAO UTILIZADO*/
399 float _gyroRoll = 0;
400 float _gyroPitch = 0;
401 unsigned char _meanPos = 0;
402 float _roll_vel[50] = {0};
403 float _pitch_vel[50] = {0};
404
405 gyroVel _meanVel;
406 /*FIM NAO UTILIZADO*/
407 };
408
409 #endif
410

```