```
/* ****************************************************************** */
/* File name:        IMU.h                                            */
/* File description: MPU-6050 interface header file                   */
/* Author name:      Giacomo Dollevedo, Gustavo Fernandes             */
/* Creation date:    18nov2020                                        */
/* Revision date:    12jan2021                                        */
/* ****************************************************************** */


#ifndef IMU_h
#define IMU_h

//Register Map MPU-6050 https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf
#include <Wire.h>
#include <Arduino.h>
#include <math.h>

#define I2C_SDA 21
#define I2C_SCL 22


#define MPU_ADDR        0x68 // definição do endereço do sensor MPU6050
#define WHO_AM_I        0x75 // registro de identificação do dispositivo
#define PWR_MGMT_1      0x6B // registro de configuração do gerenciamento de energia
#define GYRO_CONFIG     0x1B // registro de configuração do giroscópio
#define ACCEL_CONFIG    0x1C // registro de configuração do acelerômetro
#define ACCEL_XOUT      0x3B // registro de leitura do eixo X do acelerômetro
#define LED_BUILTIN     2    // LED do DevKit v1

#define CF_GY           0.99 //  Fator Gyro filtro complementar
#define CF_AC           0.01 //  Fator Accel filtro complementar

#define RAD_2_DEG       57.2958  //  Conversao Radianos para Graus
#define DEG_2_RAD       0.01745  //  Conversao Graus para Radianos

typedef struct mpu
 {
  int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
 };

typedef struct processedMpu
{
 float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
};

typedef struct angles
{
  float GyRoll = 0;
  float GyPitch = 0;
  float GyYaw = 0;
  float AclRoll = 0;
  float AclPitch = 0;
  float AclYaw = 0;
};

typedef struct processedAngles
{
```

```
  float Roll = 0;
  float Pitch = 0;
  float Yaw = 0;
};

typedef struct gyroVel
{
  float Roll = 0;
  float Pitch = 0;
  float Yaw = 0;
};

class IMU
{
public:
/*
*****************************************************************************
*/
/* Method's name:          initMPU
   */
/* Description:            Initialize I2C bus and MPU-6050
   */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      n/a
   */
/*
*****************************************************************************
*/
 void initMPU();


/*
*****************************************************************************
*/
/* Method's name:          readRawMPU
  */
/* Description:            Reads all sensor registers from MPU-6050 through I2C bus
  */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      mpu -> Struct containing raw read values
  */
/*
*****************************************************************************
*/
 mpu readRawMPU();

/*
*****************************************************************************
*/
```

```
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 /* Method's name:          getRotations
      */
129 /* Description:            Returns processed angular displacement after the filter
       */
130 /*                         on Roll, Pitch and Yaw
      */
131 /*
       */
132 /* Entry parameters:       n/a
       */
133 /*
       */
134 /* Return parameters:      _procAng -> internal processed angular displacement struct
      */
135 /*
    ********************************************************************************
    */
136  processedAngles getRotations();
137
138
139 /*
    ********************************************************************************
    */
140 /* Method's name:          getGyroVel
      */
141 /* Description:            Returns the velocity mean from the gyroscope sensor
       */
142 /*                         on Roll, Pitch and Yaw
      */
143 /*
       */
144 /* Entry parameters:       n/a
       */
145 /*
       */
146 /* Return parameters:      gyroVel -> internal mean velocities struct
      */
147 /*
    ********************************************************************************
    */
148 gyroVel getGyroVel();
149
150 /*
    ********************************************************************************
    */
151 /* Method's name:          getPitchVel
       */
152 /* Description:            Returns gyro pitch velocity after complementary filter
      */
153 /*
       */
154 /* Entry parameters:       n/a
       */
155 /*
       */
```

```
/* Return parameters:      float -> Pitch velocity
   */
/*
*******************************************************************************
*/
float getPitchVel();

/*
*******************************************************************************
*/
/* Method's name:          getRollVel
  */
/* Description:            Returns gyro roll velocity after complementary filter
   */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      float -> Roll velocity
  */
/*
*******************************************************************************
*/
float getRollVel();

/*
*******************************************************************************
*/
/* Method's name:          getYawVel
   */
/* Description:            Returns gyro yaw velocity after complementary filter
  */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      float -> Yaw velocity
   */
/*
*******************************************************************************
*/
float getYawVel();

/*
*******************************************************************************
*/
/* Method's name:          CalibrateGyro
   */
/* Description:            Set gyro calibration values for baseline shift
  */
/*
   */
/* Entry parameters:       float X -> X axis calibration value
   */
```

```
/*                      float Y -> Y axis calibration value
   */
/*                      float Z -> Z axis calibration value
   */
/*
   */
/* Return parameters:      n/a
   */
/*
*************************************************************************
*/
 void CalibrateGyro(float X, float Y, float Z);


/*
*************************************************************************
*/
/* Method's name:          CalibrateAcl
  */
/* Description:            Set accelerometer calibration values for baseline shift
   */
/*
   */
/* Entry parameters:       float X -> X axis calibration value
   */
/*                         float Y -> Y axis calibration value
   */
/*
   */
/* Return parameters:      n/a
   */
/*
*************************************************************************
*/
 void CalibrateAcl(float X, float Y);

/*
*************************************************************************
*/
/* Method's name:          update
  */
/* Description:            Reads from MPU-6050 and process data, updating internal
   */
/*                         values
  */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      n/a
   */
/*
*************************************************************************
*/
 void update();


```

```
/*
*********************************************************************************
*/
/* Method's name:           enableDebug
   */
/* Description:             Enables serial communication for debbugging
   */
/*
   */
/* Entry parameters:        n/a
   */
/*
   */
/* Return parameters:       n/a
   */
/*
*********************************************************************************
*/
 void enableDebug();


/*
*********************************************************************************
*/
/* Method's name:           disableDebug
  */
/* Description:             Disables serial communication for debbugging
  */
/*
   */
/* Entry parameters:        n/a
   */
/*
   */
/* Return parameters:       n/a
   */
/*
*********************************************************************************
*/
 void disableDebug();

/*
*********************************************************************************
*/
/* Method's name:           disableYawComp
  */
/* Description:             Disables roll and pitch angle compensation using yaw
  */
/*
   */
/* Entry parameters:        n/a
   */
/*
   */
/* Return parameters:       n/a
   */
/*
*********************************************************************************
*/
```

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

```
/*
   */
/* Return parameters:      unsigned char -> value that was read
  */
/*
*************************************************************************
*/
  unsigned char readRegMPU(unsigned char reg);


/*
*************************************************************************
*/
/* Method's name:          findMPU
   */
/* Description:            Check for MPU-6050 address on I2C bus
   */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      unsigned char -> 0 == not found / 1 == found
  */
/*
*************************************************************************
*/
  unsigned char findMPU();


/*
*************************************************************************
*/
/* Method's name:          checkMPU
  */
/* Description:            Check MPU-6050 status through I2C bus
   */
/*
   */
/* Entry parameters:       n/a
   */
/*
   */
/* Return parameters:      unsigned char -> 0 = not available / 1 = Active / 2 =
Sleep  */
/*
*************************************************************************
*/
  unsigned char checkMPU();


 /*
*************************************************************************
*/
/* Method's name:          filterMPUData
   */
/* Description:            Complementary filter to keep angular displacement from
  */
```

```
/*                          drifting
  */
/*
   */
/* Entry parameters:        n/a
   */
/*
   */
/* Return parameters:       n/a
   */
/*
*****************************************************************************
*/
  void filterMPUData();


/*
*****************************************************************************
*/
/* Method's name:           processMPUData
  */
/* Description:             Converts raw data to actual values. Also finds angular
  */
/*                          displacement
  */
/*
   */
/* Entry parameters:        n/a
   */
/*
   */
/* Return parameters:       processedMpu -> processed data struct
   */
/*
*****************************************************************************
*/
  processedMpu processMPUData();


/*
*****************************************************************************
*/
/* Method's name:           processAngles
   */
/* Description:             Converts gyro and accel data into angular displacement
  */
/*
   */
/* Entry parameters:        processedMpu dados -> data struct to process
  */
/*
   */
/* Return parameters:       n/a
   */
/*
*****************************************************************************
*/
  void processAngles(processedMpu dados);

```

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

```
/*
*****************************************************************************
*/
  void setAccelScale();


  mpu   _rawData;
  processedMpu _processedData;

  float calGyX = 0;
  float calGyY = 0;
  float calGyZ = 0;

  float calAcX = 0;
  float calAcY = 0;


  float _gyroRollInput = 0;
  float _gyroPitchInput = 0;
  float _gyroYawInput = 0;


  unsigned long _lastTimestamp = 0;

  angles _ang;
  processedAngles _procAng;

  unsigned char led_state = 0;
  unsigned char yaw_compensation = 1;
  unsigned char debbuging_enabled = 1;

  /*NAO UTILIZADO*/
  float _gyroRoll = 0;
  float _gyroPitch = 0;
  unsigned char _meanPos = 0;
  float _roll_vel[50] = {0};
  float _pitch_vel[50] = {0};

  gyroVel _meanVel;
  /*FIM NAO UTILIZADO*/
};

#endif

```