```
/*****************************************************************************************************************
* This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General    *
* Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.  *
* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied   *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more  *
* details. You should have received a copy of the GNU Lesser General Public License along with this program.          *
* If not, see <http://www.gnu.org/licenses/> or <http://blog.robotica.eng.br/gnu-lesser-general-public-license/>.     *
*                                                                                                                    *
* © Copyright 2021 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                          *
*****************************************************************************************************************
* Esta biblioteca aplica o filtro de Kalman ao eixo X do MPU6050. Adaptada do trabalho de Kristian Sloth Lauszus      *
* (https://lauszus.com/) através da licença "License version 2 (GPL2)" do GNU.                                       *
*****************************************************************************************************************/

#ifndef Kalman_X_h
#define Kalman_X_h

#include <I2CRW.h>                                          // Biblioteca para ler e gravar registros do MPU6050

Kalman kalmanX;                                             // Instância Kalman

double accX,accY,accZ;                                      // Dados do acelerometro nos eixos X,Y,Z
double gyroX,gyroY,gyroZ;                                   // Dados do giroscópio nos eixos X,Y,Z
int16_t tempRaw;                                            // Temporario

uint32_t timer;                                             // Marcação de tempo
uint8_t i2cData[14];                                        // Buffer para dados I2C

//_____________________________________________________________________________________________________________

void Kalman_X_begin(uint8_t adr)                            // Inicialização do processo de filtragem
{

  I2CRW_begin(adr,400000UL,3000,true);                      // Inicializa a biblioteca <I2C_Write_Read.h>

  // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[0] = 7;
  // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro filtering, 8 KHz sampling
  i2cData[1] = 0x00;
  // Set Gyro Full Scale Range to ±250deg/s
  i2cData[2] = 0x00;
  // Set Accelerometer Full Scale Range to ±2g
  i2cData[3] = 0x00;
  while (i2cWrite(0x19,i2cData,4,false));                   // Envia os dados e não libera o barramento I2C
  while (i2cWrite(0x6B,1,true));                            // PLL(Phase locked loop) with X axis gyroscope
                                                            // reference and disable sleep mode
  delay(100);                                               // Aguarda estabilização do sensor

  // Set kalman and gyro starting angle
  while (i2cRead(0x3B,i2cData,6));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);

  // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
  // atan2 outputs the value of -π to π (radians) - see http://en.wikipedia.org/wiki/Atan2
  // It is then converted from radians to degrees

  double roll = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  kalmanX.setAngle(roll);                                   // Set starting angle

  timer = micros();                                         // Inicializa timer
}

//_____________________________________________________________________________________________________________

void Kalman_X_run(double *ang)                              // Execução do processo de filtragem
{
  // Atualiza todos os valores
  while (i2cRead(0x3B,i2cData,14));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
  tempRaw = (int16_t)((i2cData[6] << 8) | i2cData[7]);
  gyroX = (int16_t)((i2cData[8] << 8) | i2cData[9]);
  gyroY = (int16_t)((i2cData[10] << 8) | i2cData[11]);
  gyroZ = (int16_t)((i2cData[12] << 8) | i2cData[13]);;

  double dt = (double)(micros() - timer) / 1000000;         // calcula dt
  timer = micros();

  // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
  // atan2 outputs the value of -π to π (radians) - see http://en.wikipedia.org/wiki/Atan2
  // It is then converted from radians to degrees

  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double gyroXrate = gyroX / 131.0;                         // Converte para graus/seg
```

```
  *ang = kalmanX.getAngle(roll,gyroXrate,dt);                  // Calcula o angulo usando o filtro de Kalman
}

//_____________________________________________________________________________________________________

#endif
```