```cpp
/*****************************************************************
 * Descrição: Arquivo c++ que implementa o controlador de voo do drone.
 * Autores: Gustavo L. Fernandes e Giácomo A. Dollevedo
 * Ultima Atualização: 6/01/2021
*****************************************************************/
#include "FlightControl.h"
#include "ThrottleControl.h"
#include "Arduino.h"
#include "IMU.h"
#include <analogWrite.h>
#include <ESP32PWM.h>
#include <ESP32Servo.h>
#include <ESP32Tone.h>

/* ********************************************************************************
*/
/* Nome do metodo:          FlighControl  - Construtor Vazio
*/
/* Descrição:               Inicializa um objeto controlador padrão, ou seja
*/
/*                          com valores de ganho nulos
*/
/*                          Setpoint nulo e constante
*/
/*
*/
/* Parametros de entrada:  Nenhum (Vazio)
*/
/*
*/
/*
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ********************************************************************************
*/

FlightControl::FlightControl(char Axis)
{

//Ganho de cada um dos controladores, proporcional, integrativo e derivativo
    _gains.fkp = 0;
    _gains.fki = 0;
    _gains.fkd = 0;
    _axis = Axis;
    _fimem = 0;


//Set point de entrada para o controlador (Inicial = 0)
    _setPoint = 0;
}

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

```
/* Nome do metodo:          FlighControl  - Construtor Pre Definido
*/
/* Descrição:               Inicializa um objeto controlador com ganhos definidos
*/
/*
*/
/*
*/
/* Parametros de entrada: (float fkp, float fki, float fkd) ganhos dos
controladores*/
/*                         proporcional, integrador e derivativo
*/
/*                          Setpoint nulo e constante
*/
/*                         (char Axis) podendo ser "r", "p", "y" para determinar
*/
/*                         qual o eixo que o objeto será definido
*/
/*
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ************************************************************************
*/

FlightControl::FlightControl(float fkp, float fki, float fkd, char Axis)
{

//Ganho de cada um dos controladores, proporcional, integrativo e derivativo
    _gains.fkp = fkp;
    _gains.fki = fki;
    _gains.fkd = fkd;
    _axis = Axis;
    _fimem = 0;
    _ferrorPrevious = 0;

//Set point de entrada para o controlador (Inicial = 0)
    _setPoint = 0;
}


/* ************************************************************************
*/
/* Nome do metodo:          pidVelControl
*/
/* Descrição:                - Rotina de Controle de velocidade
*/
/*                          para um determinado eixo
*/
/*
*/
/*
*/
/* Parametros de entrada: (float axisVel) variavel que contem os valores medidos
*/
```

```
/*  de velocidade lidos da IMU que será dado como entrada pro controlador
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/


void FlightControl::pidVelControl(float axisVel)
{
//YAW = Z
//ROLL = Y
//PITCH = X
//Taxas de variação angular em cada um dos eixos da IMU
    float fgyro;

//Testa qual eixo o objeto foi relacionado para coletar a informação adequada da IMU
        fgyro = axisVel;


//Erro instantaneo e acumulado
    float ferror_temp = fgyro - _setPoint;
    _fimem += _gains.fki * ferror_temp;

//limitando o sinal de erro acumulado

  if(_fimem > PIDMAX) {
    _fimem = PIDMAX;
  }
  else if(_fimem < (PIDMAX * -1)){
    _fimem = PIDMAX * -1;
  }

//Sinal de saida do PID e atualização do erro previo

    _fpidCalculated = _gains.fkp*ferror_temp + _fimem + _gains.fkd*(ferror_temp -
_ferrorPrevious);

//limitando o limite de ajuste possível

  if(_fpidCalculated > PIDMAX) {
    _fpidCalculated = PIDMAX;
  }
  else if(_fpidCalculated < (PIDMAX * -1)){
    _fpidCalculated = PIDMAX * -1;
  }

    _ferrorPrevious = ferror_temp;
}

/* ************************************************************************
*/
/* Nome do metodo:          getPID_Calculated
*/
/* Descrição:                - retorna o valor de PID calculado a ser utilizado
*/
```

```
/*                          para saida do sistema
*/
/*
*/
/* Parametros de entrada: Nenhum (Vazio)
*/
/*
*/
/* Parametros de saida: float _pidCalculated
*/
/*
*/
/*
*/
/* ************************************************************************
*/


float FlightControl::getPID_Calculated(){
    return _fpidCalculated;
}

/* ************************************************************************
*/
/* Nome do metodo:          getGains
*/
/* Descrição:                - retorna o valor dos ganhos do controlador PID
*/
/*
*/
/*
*/
/* Parametros de entrada: Nenhum (Vazio)
*/
/*
*/
/* Parametros de saida: kPID _gains
*/
/*
*/
/*
*/
/* ************************************************************************
*/

kPID FlightControl::getGains(){
    return _gains;
}

/* ************************************************************************
*/
/* Nome do metodo:          setKp
*/
/* Descrição:                - Seta um novo valor para o ganho do controlador
*/
/*                             proporcional
*/
/*
*/
```

```
/* Parametros de entrada: float kp
*/
/*
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ************************************************************************
*/

void FlightControl::setKp(float kp){
    _gains.fkp = kp;
}

/* ************************************************************************
*/
/* Nome do metodo:          setKd
*/
/* Descrição:                - Seta um novo valor para o ganho do controlador
*/
/*                             derivativo
*/
/*
*/
/* Parametros de entrada: float kd
*/
/*
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ************************************************************************
*/
void FlightControl::setKd(float kd){
    _gains.fkd = kd;

}

/* ************************************************************************
*/
/* Nome do metodo:          setKi
*/
/* Descrição:                - Seta um novo valor para o ganho do controlador
*/
/*                             integrativo
*/
/*
*/
/* Parametros de entrada: float ki
*/
/*
*/
```

```
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ************************************************************************
*/

void FlightControl::setKi(float ki){
    _gains.fki = ki;

}


/* ************************************************************************
*/
/* Nome do metodo:          setSetPoint
*/
/* Descrição:                - Seta um novo valor para o setPoint do controlador
*/
/*                             de velocidade
*/
/*
*/
/* Parametros de entrada: float newSetPoint
*/
/*
*/
/* Parametros de saida: Nenhum (Vazio)
*/
/*
*/
/*
*/
/* ************************************************************************
*/

void FlightControl::setSetPoint(float newSetPoint){
  _setPoint = newSetPoint;
}


/* ************************************************************************
*/
/* Nome do metodo:          getSetPoint
*/
/* Descrição:                - Consulta o atual valor para o setPoint do
controlador*/
/*
*/
/*
*/
/* Parametros de entrada: Nenhum (Vazio)
*/
/*
*/
/* Parametros de saida: float _setPoint
*/
```

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

float FlightControl::getSetPoint(){
  return _setPoint;
}

/* ************************************************************************
*/
/* Nome do metodo:          getAccError
*/
/* Descrição:                - Consulta o erro acumulado do controlador
*/
/*
*/
/*
*/
/* Parametros de entrada: Nenhum (Vazio)
*/
/*
*/
/* Parametros de saida: float _fimem
*/
/*
*/
/*
*/
/* ************************************************************************
*/

float FlightControl::getAccError(){
    return _fimem;
}


/* ************************************************************************
*/
/* Nome do metodo:          getPreviousError
*/
/* Descrição:                - Consulta o erro previo  do controlador
*/
/*
*/
/*
*/
/* Parametros de entrada: Nenhum (Vazio)
*/
/*
*/
/* Parametros de saida: float _ferrorPrevious
*/
/*
*/
/*
*/
```

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

float FlightControl::getPreviousError(){
    return _ferrorPrevious;
}

```