Controle de temperatura

Gerado por Doxygen 1.9.2

Capítulo 1

Índice dos Arquivos

1.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

Capítulo 2

Arquivos

2.1 Referência do Arquivo D:/BACKUP/Faculdade/16_Embarcados/↩ Controlador_Temperatura/Controle-de-temperatura/↩ Codigos/controle_temperatura_RTOS/Core/Src/FIRFilter.c

implmentação de um filtro digital tipo FIR

```
#include "FIRFilter.h"
```

Funções

- void FIRFilter_Init (FIRFilter ∗fir)

  Função de inicialização Filtro FIR.
- float FIRFilter_Update (FIRFilter ∗fir, float inp)

  Função de atualização do filtro FIR.

2.1.1 Descrição detalhada

implmentação de um filtro digital tipo FIR

Autor

Arthur Damasceno

Atenção

Controlador de temperatura

Este código apresenta a implementação de um filtro digital passa baixas FIR

Definição no arquivo FIRFilter.c.

2.1.2 Funções

2.1.2.1 FIRFilter_Init()

```
void FIRFilter_Init (
            FIRFilter * fir )
```

Função de inicialização Filtro FIR.

Parâmetros

FIRFilter	*fir: ponteiro de estrutura do filtro

Valores Retornados

None	

Definição na linha 25 do arquivo FIRFilter.c.

```
00025                                       {
00026
00027     /* Limpa buffer do filtro */
00028     for(uint8_t n=0; n<FIR_FILTER_LENGTH;n++){
00029
00030         fir->buf[n] = 0.0f;
00031
00032     }
00033
00034     /* Reseta o index do buffer */
00035     fir->bufindex = 0;
00036
00037     /* Limpa saída do filtro */
00038     fir->out = 0.0f;
00039 }
```

2.1.2.2 FIRFilter_Update()

```
float FIRFilter_Update (
            FIRFilter * fir,
            float inp )
```

Função de atualização do filtro FIR.

Parâmetros

FIRFilter	*fir: ponteiro de estrutura do filtro
float	inp: variável a ser filtrada

Valores Retornados

float	fir->out: variável filtrada

Definição na linha 47 do arquivo FIRFilter.c.

```
00047                                                {
00048 
00049     /* Salva ultima amostra no buffer */
00050     fir->buf[fir->bufindex] = inp;
00051 
00052     /* incrementa o index e "da a volta" no buffer circular caso necessário */
00053     fir->bufindex++;
00054 
00055     if(fir->bufindex == FIR_FILTER_LENGTH){
00056 
00057         fir->bufindex = 0;
00058 
00059     }
00060 
00061     /* Calcula nova saída via convolução */
00062     fir->out = 0.0f;
00063 
00064     uint8_t sumIndex = fir->bufindex;
00065 
00066     for(uint8_t n=0; n<FIR_FILTER_LENGTH;n++){
00067 
00068         /* Decrementa o index e "da a vola" no buffer circular caso necessário */
00069         if(sumIndex>0){
00070 
00071             sumIndex--;
00072 
00073         }else{
00074 
00075             sumIndex = FIR_FILTER_LENGTH -1;
00076 
00077         }
00078 
00079         /* Multiplica a resposta ao impulso com a amostra deslocada e soma a saída */
00080         fir->out += FIR_IMPULSE_RESPONSE[n] * fir->buf[sumIndex];
00081     }
00082 
00083     /* Retorna a saída filtrada */
00084     return fir->out;
00085 
00086 }
```

2.2 FIRFilter.c

Vá para a documentação desse arquivo.

```
00001 
00015 #include "FIRFilter.h"
00016 
00017 /* Coeficientes projetados de filtro */
00018 static float FIR_IMPULSE_RESPONSE[FIR_FILTER_LENGTH] = {-0.01238356f, 0.10332170f, 0.81812371f,
       0.10332170f, -0.01238356f};
00019 
00025 void FIRFilter_Init(FIRFilter *fir){
00026 
00027     /* Limpa buffer do filtro */
00028     for(uint8_t n=0; n<FIR_FILTER_LENGTH;n++){
00029 
00030         fir->buf[n] = 0.0f;
00031 
00032     }
00033 
00034     /* Reseta o index do buffer */
00035     fir->bufindex = 0;
00036 
00037     /* Limpa saída do filtro */
00038     fir->out = 0.0f;
00039 }
00040 
00047 float FIRFilter_Update(FIRFilter *fir, float inp){
00048 
00049     /* Salva ultima amostra no buffer */
00050     fir->buf[fir->bufindex] = inp;
00051 
00052     /* incrementa o index e "da a volta" no buffer circular caso necessário */
00053     fir->bufindex++;
00054 
00055     if(fir->bufindex == FIR_FILTER_LENGTH){
00056 
00057         fir->bufindex = 0;
00058 
00059     }
```

```
00060
00061     /* Calcula nova saída via convolução */
00062     fir->out = 0.0f;
00063
00064     uint8_t sumIndex = fir->bufindex;
00065
00066     for(uint8_t n=0; n<FIR_FILTER_LENGTH;n++){
00067
00068         /* Decrementa o index e "da a vola" no buffer circular caso necessário */
00069         if(sumIndex>0){
00070
00071             sumIndex--;
00072
00073         }else{
00074
00075             sumIndex = FIR_FILTER_LENGTH -1;
00076
00077         }
00078
00079         /* Multiplica a resposta ao impulso com a amostra deslocada e soma a saída */
00080         fir->out += FIR_IMPULSE_RESPONSE[n] * fir->buf[sumIndex];
00081     }
00082
00083     /* Retorna a saída filtrada */
00084     return fir->out;
00085
00086 }
```

Índice Remissivo