Arquivos

Referência do Arquivo Trabalho.c

```
#include <Arduino_FreeRTOS.h>
#include <semphr.h>
```

Definições e Macros

- #define **PINPWM** 10
- #define **PINTEMP** A0

Funções

- void **TaskAnalogRead** (void *pvParameters)
- void **TaskPWM** (void *pvParameters)
- void **TaskPID** (void *pvParameters)
- void **setup** ()
- void **loop** ()

Variáveis

- SemaphoreHandle_t **mutex**
- volatile float **setpoint** = 30
- volatile float **Pid**
- volatile int **temperatura**
- volatile int **anteriortemp** = 0

Definições e macros

#define PINPWM 10

#define PINTEMP A0

Funções

void loop ()

```
60 {
61   // Vazio
62 }
```

void setup ()

```
20              {
21
22   // Inicialização da conexão serial com 9600 bits por segundo
23   Serial.begin(9600);
24
25   while (!Serial) {
26     ;                 //Aguardando conexão com o serial
27   }
28   //criação do mutex
29   mutex = xSemaphoreCreateMutex();
30   if (mutex != NULL) {
31     Serial.println("Mutex criado");
32   }
33   //Criação das tasks do sistema
34   xTaskCreate(
35     TaskAnalogRead
36     ,  "AnalogRead"
```

```
37      ,  128  // Tamanho da pilha
38      ,  NULL
39      ,  2  // Prioridade , 2 - MAX e 0 - MIN
40      ,  NULL );
41
42    xTaskCreate(
43      TaskPID
44      ,  "PID"
45      ,  128  // Tamanho da pilha
46      ,  NULL
47      ,  1  // Prioridade
48      ,  NULL );
49
50    xTaskCreate(
51      TaskPWM
52      ,  "PWM"   // Nome
53      ,  128  // Tamanho da pilha
54      ,  NULL
55      ,  0  // Prioridade
56      ,  NULL );
57  }
```

void TaskAnalogRead (void * *pvParameters*)

```
66  {
67    (void) pvParameters;
68
69    //definindo o pino A0 como entrada.
70    pinMode(PINTEMP, INPUT);
71
72    volatile float Sensor;
73
74    for (;;)  //laço infinito
75    {
76      //entrando no mutex
77      if ( mutex != NULL ) {
78        if ( xSemaphoreTake( mutex, ( TickType t ) 10 ) == pdTRUE ) {
79          // ler o valor analogico do pino 0:
80          Sensor = analogRead(PINTEMP);
81          //fazendo a conversão do valor recebido para a faixa de temperatura do sensor
tmp36
82          //esses valores de 20 ate 358 foram retirados do datasheet do dispositivo.
83          anteriortemp = temperatura;
84          temperatura = map(Sensor, 20, 358, -40, 125);
85          // print no terminal do valor da temperatura:
86          Serial.print("Temperatura: ");
87          Serial.println(temperatura);
88          delay(10);
89          //saindo do mutex
90          xSemaphoreGive(mutex);
91          vTaskDelay(10);  // delay para manter estavel os prints.
92        }
93      }
94    }
95  }
```

void TaskPID (void * *pvParameters*)

```
98  {
99    (void) pvParameters;
100
101    volatile float k = 100;        //ganho de referencia
102    volatile float h = 0.1;        //taxa de amostragem (segundos)
103    volatile float ti = 10;        //tempo de integração
104    volatile float td = 0.1;       //tempo de derivação
105
106    volatile float kp = k * (1 + h / (2 * ti)); //ganho proporcional
107    volatile float ki = k / ti;           //ganho integral
108    volatile float kd = k * td;           //ganho derivativo
109
110    float p = 0, i = 0, d = 0;       //sinal proporcional, integral e derivativo
111
112    for (;;) //laço infinito
113    {
114      //entrando no mutex
115      if ( mutex != NULL ) {
116        if ( xSemaphoreTake( mutex, ( TickType_t ) 10 ) == pdTRUE ) {
```

```
117         //calculo do erro
118         volatile float erro = temperatura - setpoint;
119         Serial.print("Erro = ");
120         Serial.println(erro);
121         //calculo do P, I e D separados
122         // Parte proporcional
123         p = kp * erro;
124         //Parte integral
125         i = i + (ki * h * erro);
126         //Parte derivativa
127         d = (anteriortemp - temperatura) * kd;
128         //P.I.D (sinal de controle)
129         Pid = p + i + d;
130         //saindo do mutex
131         xSemaphoreGive(mutex);
132         vTaskDelay(10);
133       }
134     }
135   }
136 }
```

void TaskPWM (void * *pvParameters*)

```
139 {
140   (void) pvParameters;
141 
142   // definindo o pino 10(PWM) como saída.
143   pinMode(PINPWM, OUTPUT);
144   volatile float PWM;
145 
146   for (;;) // laço infinito
147   {
148     //entrando no mutex
149     if ( mutex != NULL ) {
150       if ( xSemaphoreTake( mutex, ( TickType_t ) 10 ) == pdTRUE ) {
151         //tornar compatível com as saídas PWM do arduino
152         PWM = map(Pid, -4098, 4098, 0, 255);  //váriavel manipulada
153         Serial.print("PWM = ");
154         Serial.println(PWM);
155         analogWrite(PINPWM, PWM);
156         //saindo do mutex
157         xSemaphoreGive(mutex);
158         vTaskDelay(10);
159       }
160     }
161   }
162 }
```

Variáveis

volatile int anteriortemp = 0

SemaphoreHandle_t mutex

volatile float Pid

volatile float setpoint = 30

volatile int temperatura

Sumário

INDEX