FreeRTOS em Arduino: leitura de sensores de gases inflamáveis

JULIA CAROLINE SOMAVILLA

Versão 1.0

Sábado, 9 de Julho de 2022

Índice dos Arquivos

Lista de Arquivos

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

Arquivos

Referência do Arquivo C:/Users/Julia/Desktop/cod/codigo.c

```
#include <Arduino_FreeRTOS.h>
#include <queue.h>
#include <task.h>
#include <semphr.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Ultrasonic.h>
```

Definições e Macros

- #define **BAUDRATE_SERIAL** 115200
- #define **LCD_16X2_CLEAN_LINE** " "
- #define **LCD_16X2_I2C_ADDRESS** 0x27
- #define **LCD_16X2_COLS** 16
- #define **LCD_16X2_ROWS** 2
- #define **LCD_TIMER_VERIFIC** 1000
- #define **ULTRASSONICO_TRIGGER** 13
- #define **ULTRASSONICO_ECHO** 12
- #define **ULTRASSONICO_TIMER_LEITURA** 500
- #define **ANALOG_A0** 0
- #define **MQ2_TIMER_LEITURA** 1000
- #define **TIMER_SEMPH_WAIT** (TickType_t) 100
- #define **TIMER_QUEUE_WAIT** (TickType_t) 100

Funções

- LiquidCrystal_I2C **lcd** (**LCD_16X2_I2C_ADDRESS**, **LCD_16X2_COLS**, **LCD_16X2_ROWS**)
- Ultrasonic **ultrasonic** (**ULTRASSONICO_TRIGGER**, **ULTRASSONICO_ECHO**)
- void **task_lcd** (void *pvParameters)
- void **task_MQ2** (void *pvParameters)
- void **task_ultrassonico** (void *pvParameters)
- void **setup** ()
- void **loop** ()

Variáveis

- QueueHandle_t **fila_MQ2**
- QueueHandle_t **fila_ultrassonico**
- SemaphoreHandle_t **semaforo_serial**

Definições e macros

#define ANALOG_A0 0

#define BAUDRATE_SERIAL 115200

#define LCD_16X2_CLEAN_LINE " "

#define LCD_16X2_COLS 16

#define LCD_16X2_I2C_ADDRESS 0x27

#define LCD_16X2_ROWS 2

#define LCD_TIMER_VERIFIC 1000

#define MQ2_TIMER_LEITURA 1000

#define TIMER_QUEUE_WAIT (TickType_t) 100

#define TIMER_SEMPH_WAIT (TickType_t) 100

#define ULTRASSONICO_ECHO 12

#define ULTRASSONICO_TIMER_LEITURA 500

#define ULTRASSONICO_TRIGGER 13

Funções

LiquidCrystal_I2C lcd (LCD_16X2_I2C_ADDRESS , LCD_16X2_COLS , LCD_16X2_ROWS)

void loop ()

```
113 {
114     // tudo é realizado pelas tarefas, portando não é necessário programar neste
bloco.
115 }
```

void setup ()

```
53              {
54     Serial.begin(BAUDRATE_SERIAL);  //inicializa o serial
55
56     lcd.init();       //inicializa o LCD
57     lcd.backlight();  //liga o backlight
58     lcd.clear();      //limpa o LCD.
59
60     /* Criação das filas  */
61     fila_MQ2 = xQueueCreate( 1, sizeof(int) );              //aloca-se um espaço
de memória com a função xQueueCreate
62     fila_ultrassonico = xQueueCreate( 1, sizeof(float) );
63
64     /*Validação das filas*/
65     if ( (fila_MQ2 == NULL) || (fila_ultrassonico == NULL) )    {
66         Serial.println("Fila MQ2 ou Ultrasonico não criado.");
67         Serial.println("Encerrando o programa.");
68         while(1){
69
```

```
70          }
71      }
72
73      /* Criação do semáforo */
74      semaforo_serial = xSemaphoreCreateMutex();
75
76       /*Valida o Semaforo*/
77      if (semaforo_serial == NULL){
78          Serial.println("Semaforo não criado.");
79          Serial.println("Encerrando o programa.");
80          while(1){
81
82          }
83      }
84
85      /* Criação das tarefas */
86      xTaskCreate(
87        task_lcd
88        ,  (const portCHAR *)"lcd"   //nome
89        ,  156                       //tamanho (em palavra)
90        ,  NULL                      //parametro passado (como nao possui, usa-se
null
91        ,  1                         //prioridade da tarefa
92        ,  NULL );                   //handle da tarefa (opcional)
93
94      xTaskCreate(
95        task_MQ2
96        ,  (const portCHAR *) "MQ2"
97        ,  156
98        ,  NULL
99        ,  2
100        ,  NULL );
101
102      xTaskCreate(
103        task_ultrassonico
104        ,  (const portCHAR *)"ultrassonico"
105        ,  156
106        ,  NULL
107        ,  3
108        ,  NULL );
109
110 }
```

void task_lcd (void * *pvParameters*)

```
117                                      {
118     float distancia = 0.0;
119     int leitura_MQ2 = 0;
120     char linha_str[16] = {0x00};  //formata a linha a ser escrita no display
121     int distancia_cm = 0;         //mostra somente a parte inteira da distancia,
porém o decimal é transportado para fila
122
123     while(1){
124       // xQueuePeek -> "espia" a fila do sensor
125         if( xQueuePeek(fila_ultrassonico, &distancia, TIMER_QUEUE_WAIT) ){
//escreve a ultima leitura do sensor
126             lcd.setCursor(0,0);
//posiciono no começo da linha no display
127             lcd.print(LCD_16X2_CLEAN_LINE);
//escrevo uma linha "em branco"
128             lcd.setCursor(0,0);
//reposiciono no começo da linha
129
130             distancia_cm = (int)distancia;
131             sprintf (linha_str, "Dist: %d cm", distancia_cm);
//formato a escrita no displaY
132             lcd.print(linha_str);
133         }
134
135
136         if( xQueuePeek(fila_MQ2, &leitura_MQ2, TIMER_QUEUE_WAIT) ){
137             lcd.setCursor(0,1);
138             lcd.print(LCD_16X2_CLEAN_LINE);
139             lcd.setCursor(0,1);
140
141             sprintf (linha_str, "MQ2: %d", leitura_MQ2);
142             lcd.print(linha_str);
```

```
143         }
144
145
146         vTaskDelay( LCD_TIMER_VERIFIC / portTICK_PERIOD_MS ); // tempo de
verificação de atualização do display
147         // porTICK_PERIOD_MS converte o tempo setado em ms no LCD_TIMER_VERIFIC
em ticks de processador.
148     }
149 }
```

void task_MQ2 (void * *pvParameters*)

```
151                                     {
152     int leitura_analogica = 0;
153
154     while(1){
155         leitura_analogica = analogRead(ANALOG_A0);
156
157         //Insere leitura na fila
158         xQueueOverwrite(fila_MQ2, (void *)&leitura_analogica);
159
160        /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
161          até o tempo definido em TIME_SEMPH_WAIT*/
162
163        if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
164            Serial.print("- Leitura MQ-2: ");
165            Serial.println(leitura_analogica);
166            xSemaphoreGive(semaforo_serial);
167        }
168
169         vTaskDelay( MQ2_TIMER_LEITURA/ portTICK_PERIOD_MS );  //aguarda tempo
determinado em MQ2_TIMER_LEITURA para realizar prox leitura;
170     }
171 }
```

void task_ultrassonico (void * *pvParameters*)

```
173                                         {
174     float distancia_cm = 0.0;
175     long microsec = 0;
176
177     while(1){
178        //mede distância em CM
179        microsec = ultrasonic.timing();
180        distancia_cm = ultrasonic.convert(microsec, Ultrasonic::CM);
181
182        //Insere leitura na fila
183        xQueueOverwrite(fila_ultrassonico, (void *)&distancia_cm);
184
185        /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
186          até o tempo definido em TIME_SEMPH_WAIT*/
187
188        if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
189            Serial.print("- Distancia: ");
190            Serial.print(distancia_cm);
191            Serial.println("cm");
192            xSemaphoreGive(semaforo_serial);
193        }
194
195        vTaskDelay( ULTRASSONICO_TIMER_LEITURA / portTICK_PERIOD_MS );
196     }
197 }
```

Ultrasonic ultrasonic (ULTRASSONICO_TRIGGER , ULTRASSONICO_ECHO)

Variáveis

QueueHandle_t fila_MQ2

QueueHandle_t fila_ultrassonico

SemaphoreHandle_t semaforo_serial