FreeRTOS em Arduino: leitura de sensores de gases inflamáveis

JULIA CAROLINE SOMAVILLA

Versão 1.1

Quinta, 4 de Agosto de 2022

Sumário

Table of contents

Índice dos Arquivos

Lista de Arquivos

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

Arquivos

Referência do Arquivo doc/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

Definição na linha **25** do arquivo **Codigo.c**.

#define BAUDRATE_SERIAL 115200

Definição na linha **10** do arquivo **Codigo.c**.

#define LCD_16X2_CLEAN_LINE " "

Definição na linha **13** do arquivo **Codigo.c**.

#define LCD_16X2_COLS 16

Definição na linha **15** do arquivo **Codigo.c**.

#define LCD_16X2_I2C_ADDRESS 0x27

Definição na linha **14** do arquivo **Codigo.c**.

#define LCD_16X2_ROWS 2

Definição na linha **16** do arquivo **Codigo.c**.

#define LCD_TIMER_VERIFIC 1000

Definição na linha **17** do arquivo **Codigo.c**.

#define MQ2_TIMER_LEITURA 1000

Definição na linha **26** do arquivo **Codigo.c**.

#define TIMER_QUEUE_WAIT (TickType_t) 100

Definição na linha **30** do arquivo **Codigo.c**.

#define TIMER_SEMPH_WAIT (TickType_t) 100

Definição na linha **29** do arquivo **Codigo.c**.

#define ULTRASSONICO_ECHO 12

Definição na linha **21** do arquivo **Codigo.c**.

#define ULTRASSONICO_TIMER_LEITURA 500

Definição na linha **22** do arquivo **Codigo.c**.

#define ULTRASSONICO_TRIGGER 13

Definição na linha **20** do arquivo **Codigo.c**.

Funções

LiquidCrystal_I2C lcd (LCD_16X2_I2C_ADDRESS , LCD_16X2_COLS , LCD_16X2_ROWS)

void loop ()

Definição na linha **112** do arquivo **Codigo.c**.

```
00113 {
00114     // tudo é realizado pelas tarefas, portando não é necessário programar neste
bloco.
00115 }
```

void setup ()

Definição na linha **53** do arquivo **Codigo.c**.

```
00053             {
00054     Serial.begin(BAUDRATE_SERIAL);  //inicializa o serial
00055 
00056     lcd.init();       //inicializa o LCD
00057     lcd.backlight();  //liga o backlight
00058     lcd.clear();      //limpa o LCD.
00059 
00060     /* Criação das filas  */
00061     fila_MQ2 = xQueueCreate( 1, sizeof(int) );              //aloca-se um espaço
de memória com a função xQueueCreate
00062     fila_ultrassonico = xQueueCreate( 1, sizeof(float) );
00063 
00064     /*Validação das filas*/
00065     if ( (fila_MQ2 == NULL) || (fila_ultrassonico == NULL) )    {
00066         Serial.println("Fila MQ2 ou Ultrasonico não criado.");
00067         Serial.println("Encerrando o programa.");
00068         while(1){
00069 
00070         }
00071     }
00072 
00073     /* Criação do semáforo */
00074     semaforo_serial = xSemaphoreCreateMutex();
00075 
00076      /*Valida o Semaforo*/
00077     if (semaforo_serial == NULL){
00078         Serial.println("Semaforo não criado.");
00079         Serial.println("Encerrando o programa.");
00080         while(1){
00081 
00082         }
00083     }
00084 
00085     /* Criação das tarefas */
00086     xTaskCreate(
00087       task_lcd
00088       ,  (const portCHAR *)"lcd"   //nome
00089       ,  156                       //tamanho (em palavra)
00090       ,  NULL                      //parametro passado (como nao possui, usa-se
null
00091       ,  1                         //prioridade da tarefa
00092       ,  NULL );                   //handle da tarefa (opcional)
00093 
00094     xTaskCreate(
00095       task_MQ2
00096       ,  (const portCHAR *) "MQ2"
00097       ,  156
00098       ,  NULL
00099       ,  2
00100       ,  NULL );
00101 
00102     xTaskCreate(
00103       task_ultrassonico
00104       ,  (const portCHAR *)"ultrassonico"
```

```
00105       ,  156
00106       ,  NULL
00107       ,  3
00108       ,  NULL );
00109
00110 }
```

void task_lcd (void * *pvParameters*)

Definição na linha **117** do arquivo **Codigo.c**.

```
00117                                     {
00118     float distancia = 0.0;
00119     int leitura_MQ2 = 0;
00120     char linha_str[16] = {0x00};  //formata a linha a ser escrita no display
00121     int distancia_cm = 0;         //mostra somente a parte inteira da distancia,
porém o decimal é transportado para fila
00122
00123     while(1){
00124       // xQueuePeek -> "espia" a fila do sensor
00125         if( xQueuePeek(fila_ultrassonico, &distancia, TIMER_QUEUE_WAIT) ){
//escreve a ultima leitura do sensor
00126             lcd.setCursor(0,0);
//posiciono no começo da linha no display
00127             lcd.print(LCD_16X2_CLEAN_LINE);
//escrevo uma linha "em branco"
00128             lcd.setCursor(0,0);
//reposiciono no começo da linha
00129
00130             distancia_cm = (int)distancia;
00131             sprintf (linha_str, "Dist: %d cm", distancia_cm);
//formato a escrita no displaY
00132             lcd.print(linha_str);
00133         }
00134
00135
00136         if( xQueuePeek(fila_MQ2, &leitura_MQ2, TIMER_QUEUE_WAIT) ){
00137             lcd.setCursor(0,1);
00138             lcd.print(LCD_16X2_CLEAN_LINE);
00139             lcd.setCursor(0,1);
00140
00141             sprintf (linha_str, "MQ2: %d", leitura_MQ2);
00142             lcd.print(linha_str);
00143         }
00144
00145
00146         vTaskDelay( LCD_TIMER_VERIFIC / portTICK_PERIOD_MS ); // tempo de
verificação de atualização do display
00147         // porTICK_PERIOD_MS converte o tempo setado em ms no LCD_TIMER_VERIFIC
em ticks de processador.
00148     }
00149 }
```

void task_MQ2 (void * *pvParameters*)

Definição na linha **151** do arquivo **Codigo.c**.

```
00151                                     {
00152     int leitura_analogica = 0;
00153     int Pinbuzzer = 8; //PINO UTILIZADO PELO BUZZER
00154     int pinled = 7; //PINO UTILIZADO PELO LED
00155
00156     int leitura_sensor = 400;//DEFININDO UM VALOR LIMITE (NÍVEL DE GÁS NORMAL)
00157
00158     while(1){
00159         leitura_analogica = analogRead(ANALOG_A0);
00160
00161         //Insere leitura na fila
00162         xQueueOverwrite(fila_MQ2, (void *)&leitura_analogica);
00163
00164        /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
00165          até o tempo definido em TIME_SEMPH_WAIT*/
00166
00167        if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
```

```
00168             Serial.print("- Leitura MQ-2: ");
00169             Serial.println(leitura_analogica);
00170             xSemaphoreGive(semaforo_serial);
00171         }
00172 
00173          vTaskDelay( MQ2_TIMER_LEITURA/ portTICK_PERIOD_MS );  //aguarda tempo
determinado em MQ2_TIMER_LEITURA para realizar prox leitura;
00174 
00175          pinMode(ANALOG_A0, INPUT); //DEFINE O PINO COMO ENTRADA
00176          pinMode(Pinbuzzer, OUTPUT); //DEFINE O PINO COMO SAÍDA
00177          pinMode(pinled, OUTPUT); //DEFINE O PINO COMO SAIDA
00178          Serial.begin(115200);//INICIALIZA A SERIAL
00179 
00180          int valor_analogico = analogRead(ANALOG_A0); //VARIÁVEL RECEBE O VALOR
LIDO NO PINO ANALÓGICO
00181 
00182          if (valor_analogico > leitura_sensor){//SE VALOR LIDO NO PINO ANALÓGICO
FOR MAIOR QUE O VALOR LIMITE, FAZ
00183              digitalWrite(Pinbuzzer, HIGH); //ATIVA O BUZZER E O MESMO EMITE O
SINAL SONORO
00184              digitalWrite(pinled, HIGH); //ativa o led
00185           }
00186           else{ //SENÃO, FAZ
00187              digitalWrite(Pinbuzzer, LOW);//BUZZER DESLIGADO
00188              digitalWrite(pinled, LOW);//LED DESLIGADO
00189           }
00190           delay(100); //INTERVALO DE 100 MILISSEGUNDOS
00191     }
00192 
00193 }
```

void task_ultrassonico (void * *pvParameters*)

Definição na linha **194** do arquivo **Codigo.c**.

```
00194                                                 {
00195     float distancia_cm = 0.0;
00196     long microsec = 0;
00197 
00198     while(1){
00199        //mede distância em CM
00200        microsec = ultrasonic.timing();
00201        distancia_cm = ultrasonic.convert(microsec, Ultrasonic::CM);
00202 
00203        //Insere leitura na fila
00204        xQueueOverwrite(fila_ultrassonico, (void *)&distancia_cm);
00205 
00206        /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
00207          até o tempo definido em TIME_SEMPH_WAIT*/
00208 
00209        if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
00210            Serial.print("- Distancia: ");
00211            Serial.print(distancia_cm);
00212            Serial.println("cm");
00213            xSemaphoreGive(semaforo_serial);
00214        }
00215 
00216        vTaskDelay( ULTRASSONICO_TIMER_LEITURA / portTICK_PERIOD_MS );
00217     }
00218 }
```

Ultrasonic ultrasonic (ULTRASSONICO_TRIGGER , ULTRASSONICO_ECHO)

Variáveis

QueueHandle_t fila_MQ2

Definição na linha **33** do arquivo **Codigo.c**.

QueueHandle_t fila_ultrassonico

Definição na linha **34** do arquivo **Codigo.c**.

SemaphoreHandle_t semaforo_serial

Definição na linha **37** do arquivo **Codigo.c**.

Codigo.c

Vá para a documentação desse arquivo.

```
00001 #include <Arduino_FreeRTOS.h>
//inclusão do FreeRtos para arduino
00002 #include <queue.h>                    //biblioteca para utlização de filas
00003 #include <task.h>                     //biblioteca para utilização de tarefas
00004 #include <semphr.h>                   //biblioteca para utilização de semáforos.
00005 #include <Wire.h>                     //biblioteca para uso de dispositivos com
comunicação I2C.
00006 #include <LiquidCrystal_I2C.h>        //biblioteca do display
00007 #include <Ultrasonic.h>               //biblioteca do sensor ultrasonico
00008
00009 //DEFINIÇÕES
00010 #define BAUDRATE_SERIAL                    115200 //Utilizado para debug, no serial
monitor
00011
00012 /*LCD*/
00013 #define LCD_16X2_CLEAN_LINE                "               "//evita efeito de
flick, devido a atualizaçao com frequencia muito grande
00014 #define LCD_16X2_I2C_ADDRESS               0x27            // endereço I2C do
display
00015 #define LCD_16X2_COLS                      16
00016 #define LCD_16X2_ROWS                      2
00017 #define LCD_TIMER_VERIFIC                  1000            // a cada segundo, a
tarefa do display verifica se há novas medições disponíveis.
00018
00019 /*Sensor Ultrassonico*/
00020 #define ULTRASSONICO_TRIGGER                13  //GPIO
00021 #define ULTRASSONICO_ECHO                   12  //GPIO
00022 #define ULTRASSONICO_TIMER_LEITURA          500 //tempo de verificação de leitura
do sensor(a cada meio segundo)
00023
00024 /*Sensor MQ2*/
00025 #define ANALOG_A0                 0              // DEFINIÇÃO DA PORTA A0 ANALOGICA
00026 #define MQ2_TIMER_LEITURA         1000           //utiliza a leitura em 1 segundo
00027
00028 /*Tomada de Controle*/
00029 #define TIMER_SEMPH_WAIT        ( TickType_t ) 100  //tempo para aguardar a tomada
de controle do semáforo.
00030 #define TIMER QUEUE WAIT        ( TickType t ) 100  //tempo para aguardar tomada de
controle da fila.
00031
00032 /* filas*/
00033 QueueHandle_t fila_MQ2;              //fila mq2
00034 QueueHandle_t fila_ultrassonico;    //fila sensor ultrassonico
00035
00036 /* semaforos */
00037 SemaphoreHandle_t semaforo_serial;  //semaforo usado para controlar o uso do serial,
que é compartilhado entre ambas tarefas.
00038
00039 LiquidCrystal_I2C lcd(LCD_16X2_I2C_ADDRESS,
00040                       LCD_16X2_COLS,
00041                       LCD_16X2_ROWS); //controle do lcd, parametros: endereço,
coluna e linhas
00042
00043 Ultrasonic ultrasonic(ULTRASSONICO_TRIGGER, ULTRASSONICO_ECHO); //controle do
sensor ultrassonico
00044
00045 /*Protótipo das Tarefas*/
00046 /*Todas as tarefas dos dispositivos, retornam void, e também recebem um ponteiro do
00047 tipo void como parametro. Segue este padrão como requisito para criação de tarefas
no FreeRTOS.*/
00048 void task_lcd( void *pvParameters );
00049 void task_MQ2( void *pvParameters );
00050 void task_ultrassonico( void *pvParameters );
00051
00052
00053 void setup(){
00054     Serial.begin(BAUDRATE_SERIAL);  //inicializa o serial
00055
00056     lcd.init();       //inicializa o LCD
00057     lcd.backlight();  //liga o backlight
00058     lcd.clear();      //limpa o LCD.
00059
00060     /* Criação das filas  */
```

```
00061     fila_MQ2 = xQueueCreate( 1, sizeof(int) );              //aloca-se um espaço de
memória com a função xQueueCreate
00062     fila_ultrassonico = xQueueCreate( 1, sizeof(float) );
00063
00064     /*Validação das filas*/
00065     if ( (fila_MQ2 == NULL) || (fila_ultrassonico == NULL) )    {
00066         Serial.println("Fila MQ2 ou Ultrasonico não criado.");
00067         Serial.println("Encerrando o programa.");
00068         while(1){
00069
00070         }
00071     }
00072
00073     /* Criação do semáforo */
00074     semaforo_serial = xSemaphoreCreateMutex();
00075
00076      /*Valida o Semaforo*/
00077     if (semaforo_serial == NULL){
00078         Serial.println("Semaforo não criado.");
00079         Serial.println("Encerrando o programa.");
00080         while(1){
00081
00082         }
00083     }
00084
00085     /* Criação das tarefas */
00086     xTaskCreate(
00087       task_lcd
00088       ,  (const portCHAR *)"lcd"   //nome
00089       ,  156                       //tamanho (em palavra)
00090       ,  NULL                      //parametro passado (como nao possui, usa-se
null
00091       ,  1                         //prioridade da tarefa
00092       ,  NULL );                   //handle da tarefa (opcional)
00093
00094     xTaskCreate(
00095       task_MQ2
00096       ,  (const portCHAR *) "MQ2"
00097       ,  156
00098       ,  NULL
00099       ,  2
00100       ,  NULL );
00101
00102     xTaskCreate(
00103       task_ultrassonico
00104       ,  (const portCHAR *)"ultrassonico"
00105       ,  156
00106       ,  NULL
00107       ,  3
00108       ,  NULL );
00109
00110 }
00111
00112 void loop()
00113 {
00114     // tudo é realizado pelas tarefas, portando não é necessário programar neste
bloco.
00115 }
00116
00117 void task_lcd( void *pvParameters ){
00118     float distancia = 0.0;
00119     int leitura_MQ2 = 0;
00120     char linha_str[16] = {0x00};  //formata a linha a ser escrita no display
00121     int distancia_cm = 0;         //mostra somente a parte inteira da distancia,
porém o decimal é transportado para fila
00122
00123     while(1){
00124       // xQueuePeek -> "espia" a fila do sensor
00125         if( xQueuePeek(fila_ultrassonico, &distancia, TIMER_QUEUE_WAIT) ){
//escreve a ultima leitura do sensor
00126             lcd.setCursor(0,0);
//posiciono no começo da linha no display
00127             lcd.print(LCD_16X2_CLEAN_LINE);
//escrevo uma linha "em branco"
00128             lcd.setCursor(0,0);
//reposiciono no começo da linha
00129
```

```
00130             distancia_cm = (int)distancia;
00131             sprintf (linha_str, "Dist: %d cm", distancia_cm);
//formato a escrita no displaY
00132             lcd.print(linha_str);
00133         }
00134 
00135 
00136         if( xQueuePeek(fila_MQ2, &leitura_MQ2, TIMER_QUEUE_WAIT) ){
00137             lcd.setCursor(0,1);
00138             lcd.print(LCD_16X2_CLEAN_LINE);
00139             lcd.setCursor(0,1);
00140 
00141             sprintf (linha_str, "MQ2: %d", leitura_MQ2);
00142             lcd.print(linha_str);
00143         }
00144 
00145 
00146         vTaskDelay( LCD_TIMER_VERIFIC / portTICK_PERIOD_MS ); // tempo de
verificação de atualização do display
00147         // porTICK_PERIOD_MS converte o tempo setado em ms no LCD_TIMER_VERIFIC em
ticks de processador.
00148     }
00149 }
00150 
00151 void task_MQ2( void *pvParameters ){
00152     int leitura_analogica = 0;
00153     int Pinbuzzer = 8; //PINO UTILIZADO PELO BUZZER
00154     int pinled = 7; //PINO UTILIZADO PELO LED
00155 
00156     int leitura_sensor = 400;//DEFININDO UM VALOR LIMITE (NÍVEL DE GÁS NORMAL)
00157 
00158     while(1){
00159         leitura_analogica = analogRead(ANALOG_A0);
00160 
00161         //Insere leitura na fila
00162         xQueueOverwrite(fila_MQ2, (void *)&leitura_analogica);
00163 
00164        /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
00165          até o tempo definido em TIME_SEMPH_WAIT*/
00166 
00167        if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
00168            Serial.print("- Leitura MQ-2: ");
00169            Serial.println(leitura_analogica);
00170            xSemaphoreGive(semaforo_serial);
00171        }
00172 
00173         vTaskDelay( MQ2_TIMER_LEITURA/ portTICK_PERIOD_MS );  //aguarda tempo
determinado em MQ2_TIMER_LEITURA para realizar prox leitura;
00174 
00175         pinMode(ANALOG_A0, INPUT); //DEFINE O PINO COMO ENTRADA
00176         pinMode(Pinbuzzer, OUTPUT); //DEFINE O PINO COMO SAÍDA
00177         pinMode(pinled, OUTPUT); //DEFINE O PINO COMO SAIDA
00178         Serial.begin(115200);//INICIALIZA A SERIAL
00179 
00180         int valor_analogico = analogRead(ANALOG_A0); //VARIÁVEL RECEBE O VALOR LIDO
NO PINO ANALÓGICO
00181 
00182         if (valor_analogico > leitura_sensor){//SE VALOR LIDO NO PINO ANALÓGICO FOR
MAIOR QUE O VALOR LIMITE, FAZ
00183             digitalWrite(Pinbuzzer, HIGH); //ATIVA O BUZZER E O MESMO EMITE O SINAL
SONORO
00184             digitalWrite(pinled, HIGH); //ativa o led
00185          }
00186          else{ //SENÃO, FAZ
00187             digitalWrite(Pinbuzzer, LOW);//BUZZER DESLIGADO
00188             digitalWrite(pinled, LOW);//LED DESLIGADO
00189          }
00190          delay(100); //INTERVALO DE 100 MILISSEGUNDOS
00191     }
00192 
00193 }
00194 void task_ultrassonico( void *pvParameters ){
00195     float distancia_cm = 0.0;
00196     long microsec = 0;
00197 
00198     while(1){
00199        //mede distância em CM
```

```
00200         microsec = ultrasonic.timing();
00201         distancia_cm = ultrasonic.convert(microsec, Ultrasonic::CM);
00202 
00203         //Insere leitura na fila
00204         xQueueOverwrite(fila_ultrassonico, (void *)&distancia_cm);
00205 
00206         /*escreve a leiturana serial. Tentativa de controle do semafoto é feita
00207           até o tempo definido em TIME_SEMPH_WAIT*/
00208 
00209         if ( xSemaphoreTake(semaforo_serial, TIMER_SEMPH_WAIT ) == pdTRUE ){
00210             Serial.print("- Distancia: ");
00211             Serial.print(distancia_cm);
00212             Serial.println("cm");
00213             xSemaphoreGive(semaforo_serial);
00214         }
00215 
00216         vTaskDelay( ULTRASSONICO_TIMER_LEITURA / portTICK_PERIOD_MS );
00217     }
00218 }
```

Sumário

INDEX