

UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

SENSORIAMENTO DE TEMPERATURA BASEADO EM RTOS NO

ARDUINO

RELATÓRIO DA DISCIPLINA DE PROJETO DE SISTEMAS

EMBARCADOS

Prof. Carlos Henrique Barriquello

Franciúne Barbosa da Silva de Almeida

Victor Eugenio Mainardi Fritz

Santa Maria, RS

2021

RESUMO

O presente documento tem como objetivo relatar o processo de desenvolvimento do projeto final da disciplina, que consiste na simulação de um sensor de temperatura baseado em um sistema operacional em tempo real. Para este, foi utilizado o software de simulações elétricas e eletrônicas *Proteus*, integrando o uso da simulação de uma placa *Arduino UNO* e dos demais componentes do circuito. Além da própria IDE Arduino para a implementação da mesma.

ÍNDICE

1 INTRODUÇÃO	3
1.1 FUNDAMENTAÇÃO TEÓRICA	3
2 IMPLEMENTAÇÃO EMBARCADA	3
2.1 MATERIAIS UTILIZADOS	4
2.2 HARDWARE	4
2.3 SOFTWARE	6
3 IMPLEMENTAÇÃO	7
REFERÊNCIAS BIBLIOGRÁFICAS.....	12
APÊNDICE A	13

1 INTRODUÇÃO

Através de três tarefas executadas independentemente, temos o controle de um ar condicionado através do sensoramento de temperatura. A temperatura lida como entrada define o estado do ar, assim como suas saídas.

1.1 FUNDAMENTAÇÃO TEÓRICA

O Arduino Uno é uma placa de Arduino que tem como microcontrolador principal o ATmega328P da fabricante Atmel. Tem 14 pinos digitais que podem ser utilizados como entrada e/ou saída, sendo que desses 14 pinos, 6 deles podem ser utilizados como saída PWM que é um tipo de sinal elétrico para controle de motor por largura de pulso ainda tem mais 6 pinos de entrada para sinais analógicos. Para o clock do microcontrolador é utilizado um cristal oscilador de 16Mhz, tem também conexão USB e um conector para ligação da fonte de energia, um conector para programação e um botão de reset para reiniciar a placa.

A biblioteca de código livre FreeRTOS oferece um sistema operacional em tempo real para sistemas embarcados, Por ser de código aberto, isso permitiu o surgimento de várias versões do FreeRTOS que suportam vários dispositivos. Atualmente, existem mais de 35 versões de sistemas operacionais da série de processadores, incluindo Atmel AVR, cujo microcontrolador ATmega328 é o principal componente do Arduino ONU.

O sensor LM35 é um sensor de precisão que apresenta uma saída de tensão linear em relação à temperatura quando é alimentado. Seu terminal de saída emite um sinal de 10mV por graus Celsius. Ele se sobressai em relação a outros sensores quando consideramos esta medida de temperatura, já que a maioria dos dispositivos de sensoramento trabalham com a escala Kelvin, assim, o LM35 tem uma saída mais precisa, visto que nenhuma variável é subtraída, além de ter um custo reduzido para o sistema.

2 IMPLEMENTAÇÃO EMBARCADA

Um sistema embarcado, no entendimento básico e objetivo dos conhecimentos adquiridos ao longo da disciplina, ele é definido como a integração de um software, modelado para o funcionamento de um hardware que tem como objetivo fazer toda a leitura desse código que recebe e executar funções físicas, como por exemplo sensores e atuadores

para um problema específico. O uso de sistemas embarcados, cresce diariamente devido a necessidade de sistema auxiliar para um determinado problema, que, humanamente, é inviável e ineficiente fazer essa observação. E para essa integração, é necessário a modelagem da lógica de um programa computacional “software” que seja embutido em um dispositivo físico “hardware”.

2.1 MATERIAIS UTILIZADOS

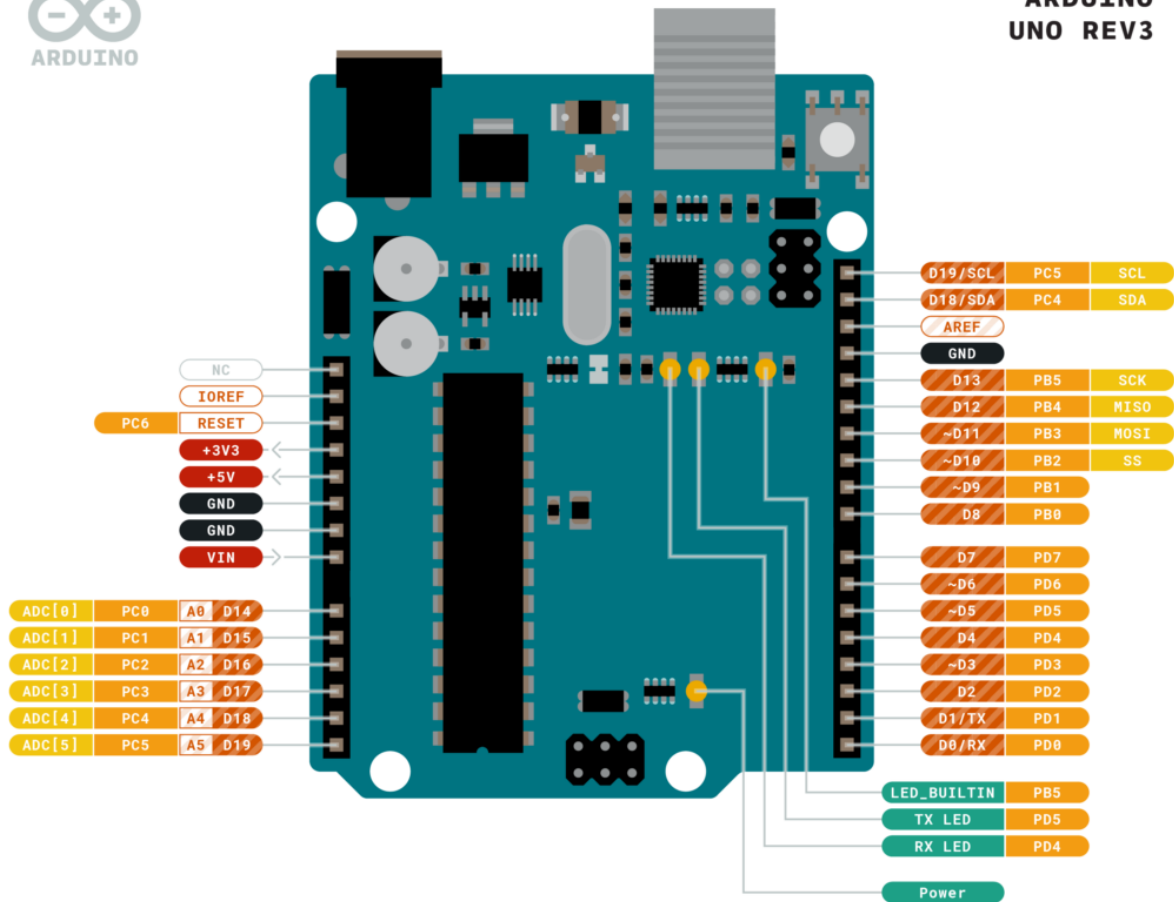
- Placa Arduino UNO;
- LCD;
- LED's verde e vermelho;
- Sensor LM35.











2.2 HARDWARE

Para complementar a simulação em software e realizar a adaptação do projeto para um projeto de sistema embarcado, é necessário e basicamente obrigatório termos um sistema de hardware para comportar e servir de alicerce para todo o software desenvolvido, bem como funções específicas de um sistema de tempo real, que é o FreeRTOS. Foi utilizado então, a placa Arduino UNO, que faz leitura do input analógico de um sensor de temperatura LM35, e também , um output de LED, indicando a situação da leitura atual.



ARDUINO UNO REV3



 Ground	 Internal Pin	 Digital Pin	 Microcontroller's Port
 Power	 SWD Pin	 Analog Pin	
 LED	 Other Pin	 Default	

ARDUINO.CC



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Figura 1 - Diagrama Arduino Mega 2560

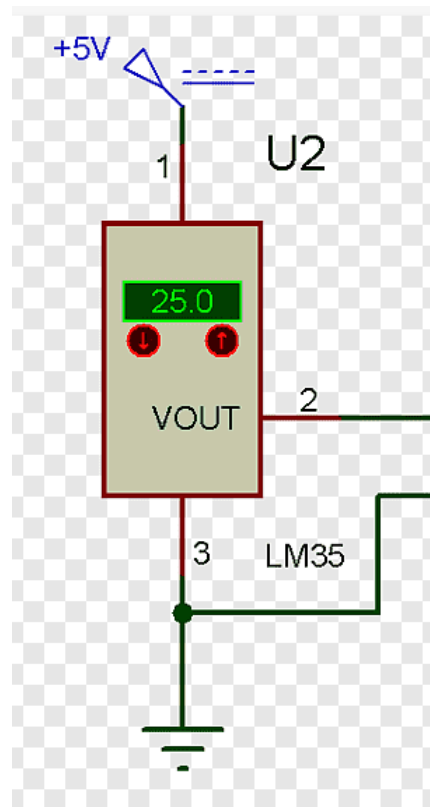


Figura 2.2 - Diagrama Sensor LM35 no Proteus

2.3 SOFTWARE

Para a implementação do projeto, foi utilizado a plataforma de desenvolvimento Arduino IDE, para a implementação de funções do FreeRTOS. Integrando o arquivo compilado *projeto_final.hex* com o simulador Proteus.

3 IMPLEMENTAÇÃO

Para a implementação das tarefas, foram utilizadas 4 funções na representação, como leitura dos dados analógicos, temperatura do sensor, cálculo da temperatura média e uma função do atuador usando LED.

3.1 Tarefa do atuador: *void task_led*

```
154 //funcao led
155 void task_led(void *pvParameters __attribute__((unused))){
156     while(){
157         //intensidade de da luz
158         int intensidade;
159         //guarda oq foi consumido do buffer, a temperatura
160         int aux;
161         if(i>0){
162             //consome o buffer
163             aux = temp_media[i];
164             if(aux>29){
165                 intensidade = map(aux, 30, 80, 0, 2500);
166                 tone(pinled, intensidade);
167             }
168         }
169         else{
170             i=0;
171             noTone(pinled);
172         }
173     }
174 }
```

Figura 3.1 - Função da tarefa do LED

3.2 Tarefa que calcula a temperatura média: *void mediaTemperatura*

```
122 //funcao temperatura media
123 void task_mediaTemperatura(void *pvParameters __attribute__((unused)){
124     while(){
125         //media da temperatura
126         float media;
127         //acumulador do buffer
128         float acumulador;
129         //se a flag é 1, calcula a media
130         if(flag==1){
131             for(int j=0; j<10; j++){
132                 acumulador = acumulador + temp_media[j];
133             }
134             //divide pelo tamanho do buffer pra calcular a media
135             media = acumulador/10;
136             //reseta a flag que indica se o buffer ta cheio ou n
137             flag=0;
138             //reseta a var de cont. do buffer
139             k=0;
140
141             if(xSemaphoreTake(xSerialSemaphore, (TickType_t)5)==pdTRUE){
142                 Serial.print("Media: ");
143                 Serial.println(media);
144                 media=0;
145                 acumulador=0;
146                 xSemaphoreGive(xSerialSemaphore);
147             }
148         }
149         //caso contrario n faz leitura
150         else{flag=0; i=k}
151     }
152 }
```

Figura 3.2 - Função da média de temperatura

3.3 Tarefa que recebe a temperatura atual: *void task_temperatura*

```
95 //funcao temperatura
96 void task_temperatura(void *pvParameters __attribute__((unused)){
97     while(){
98         struct pino pinoatual;
99         if(xQueueReceive(structQueue, &pinoatual, portMAX_DELAY)==pdPASS){
100             temp_media[k]=pinoatual.valor;
101             //checa se encheu o vetor
102             if(k<10){
103                 i = k;
104                 flag = 0;
105                 if(xSemaphoreTake(xSerialSemaphore, (TickType_t)5) == pdTRUE){
106                     //prints da serial
107                     Serial.print("Temperatura lida atual: ");
108                     Serial.println(pinoatual.valor);
109                     //printa posicao do buffer
110                     Serial.println(k);
111                     xSemaphoreGive(xSerialSemaphore);
112                     k=k+1;
113                 }
114             } else{
115                 i = 0;
116                 flag = 1;
117             }
118         }
119     }
120 }
```

Figura 3.3 - Função da tarefa de temperatura atual

3.4 Tarefa da leitura dos dados do sensor

```
81 //funcao da task do input analogico, sensor LM35
82 void task_inputAnalogico(void *pvParameters __attribute__((unused)){
83     while(){
84         //acessando struct
85         struct lerpino pinoatual;
86         pinoatual.pino = 0;
87         pinoatual.valor = (float(inputAnalogico(A0))*5 / (1023))/0.01;
88         //inserir na pilha
89         xQueueSend(structQueue, &pinoatual, portMAX_DELAY);
90         //delay da leitura
91         vTaskDelay(1);
92     }
93 }
```

Figura 3.4 - Função do input analógico do sensor

3.4 Diagrama da estrutura do projeto

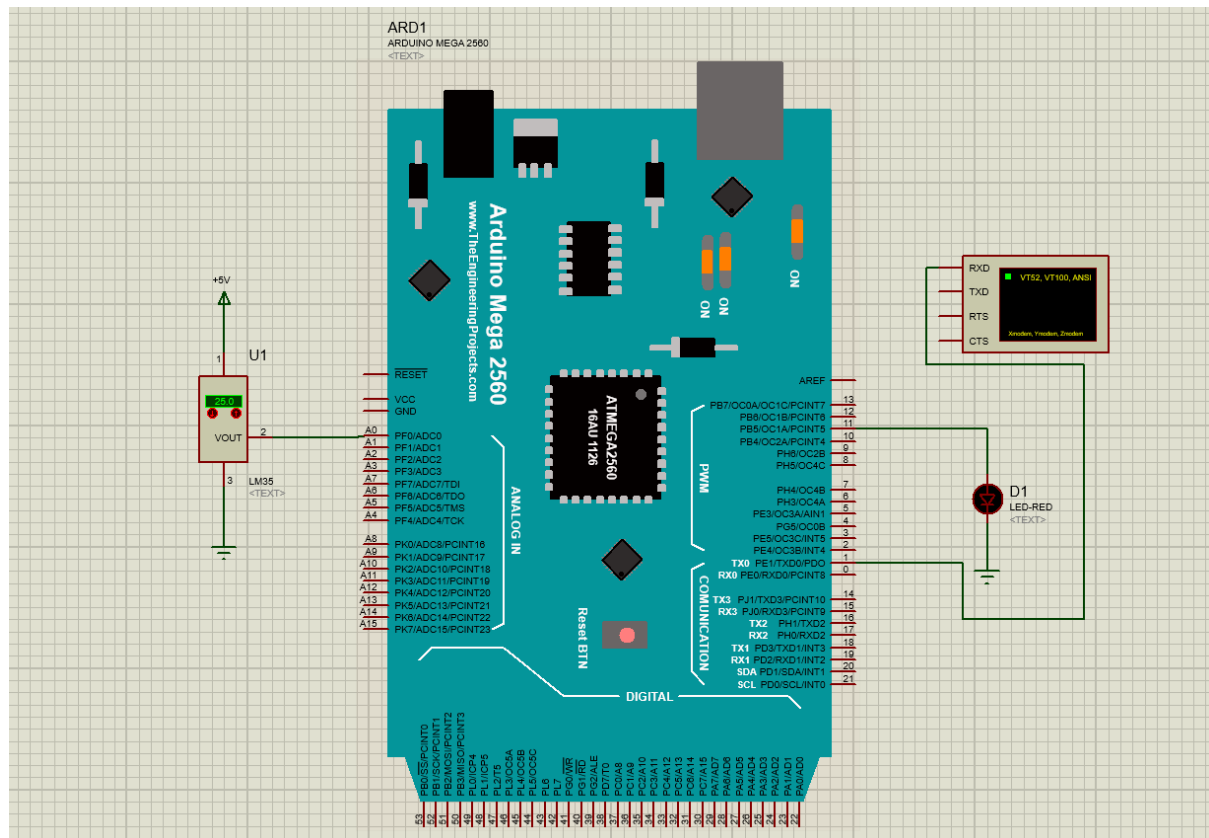


Figura 3.4 - Diagrama do projeto montado no Proteus

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] How to use FreeRTOS structure Queue to Receive Data from Multiple Tasks. Microcontrollers Lab. Disponível em: <<https://microcontrollerslab.com/arduino-freertos-structure-queue-receive-data-multiple-resources/>>. Acesso em: 21 Aug. 2021.
- [2] BERTOLETI, Pedro. Principais conceitos de RTOS para iniciantes com Arduino e FreeRTOS. Embarcados - Sua fonte de informações sobre Sistemas Embarcados. Disponível em: <<https://www.embarcados.com.br/rtos-para-iniciantes-com-arduino-e-freertos/>>. Acesso em: 21 Aug. 2021.
- [3] Arduino - AnalogRead. Arduino.cc. Disponível em: <<https://www.arduino.cc/en/Reference/AnalogRead>>. Acesso em: 21 Aug. 2021.
- [4] SENSOR DE TEMPERATURA LM35. Sensor de Temperatura LM35. Bau Eletrônica. Disponível em: <<https://www.baudaeletronica.com.br/sensor-de-temperatura-lm35.html#:~:text=O%20Sensor%20de%20Temperatura%20LM35,cada%20grau%20celsius%20de%20temperatura.>>. Acesso em: 21 Aug. 2021.
- [5] How to “Multithread” an Arduino (Protothreading Tutorial). Arduino Project Hub. Disponível em: <<https://create.arduino.cc/projecthub/reanimationxp/how-to-multithread-an-arduino-prototyping-tutorial-dd2c37>>. Acesso em: 25 Aug. 2021.

