

WarmBeforeBath



José Neves
Miguel Fazenda



BitChallenge 2018

Conceito

Na nossa proposta de projeto, falámos do quanto desagradável era em dias de frio acordar, sair da cama que estava quente para o resto da casa que estava frio e ter de terminar aquele banho que nos estava a saber tão bem. Dissemos que era muito bom que a casa de banho estivesse já quente quando lá chegássemos e que esse seria o objetivo do nosso trabalho. Nessa altura, o projeto consistia então numa aplicação para telemóvel que comunicava a um dispositivo na casa de banho a hora de despertar do utilizador e a temperatura a que a ele queria, utilizando um aquecedor para o fazer.

A ideia sofreu, no entanto, algumas alterações resultantes tanto de dificuldades como de tentativas de satisfazer melhor o utilizador. A eliminação da aplicação mobile é porventura a que salta mais à vista. Problemas na utilização simultânea do módulo ESP8266 com o Arduino, aliados a uma insuficiência de memória no microcontrolador levaram-nos a descartar esta funcionalidade, que consideramos integrar numa versão já mais avançada do projeto. Todavia, esta decisão não se revelou em todos os aspetos prejudicial: ela permitiu-nos também perceber que o nosso projeto podia ter outras aplicações. Embora possa continuar a utilizar o dispositivo para aquecer a sua casa de banho quando acorda e vai tomar banho, ele pode servir para fazer o mesmo com a sua cozinha quando for tomar o pequeno almoço ou o seu quarto quando se for deitar. Outras funcionalidades não foram implementadas: escolha de idioma, que, apesar de importante num dispositivo comercial, não mostraria capacidades enquanto concorrentes, e um modo automático de desligar, que se revelou inexequível quer através de medições de humidade ou incidência luminosa, como inicialmente pensámos.

Funcionalidades

A ideia, contudo, continua a ser bastante simples e, portanto, prática, basta dizer ao dispositivo quando quer o local quente e durante quanto tempo e ele fá-lo á por si, podendo escolher horas diferentes para cada dia da semana. Se quiser, pode até trocá-lo de lugar: de manhã aquece-lhe a casa de banho e à noite o quarto. Por outro lado, decidimos tornar o aparelho partilhável, isto é, o mais provável é que não haja apenas um utilizador mas vários. Assim, imagine-se uma família, o pai pode acordar, utilizá-lo e, passado algum tempo, vir o filho mais novo que se queixa sempre do frio na casa de banho e que passa a não poder fazê-lo. Até cinco utilizadores são bem-vindos e cada um poderá escolher uma temperatura e um intervalo de tempo que considere adequados, para além da óbvia hora por dia da semana.

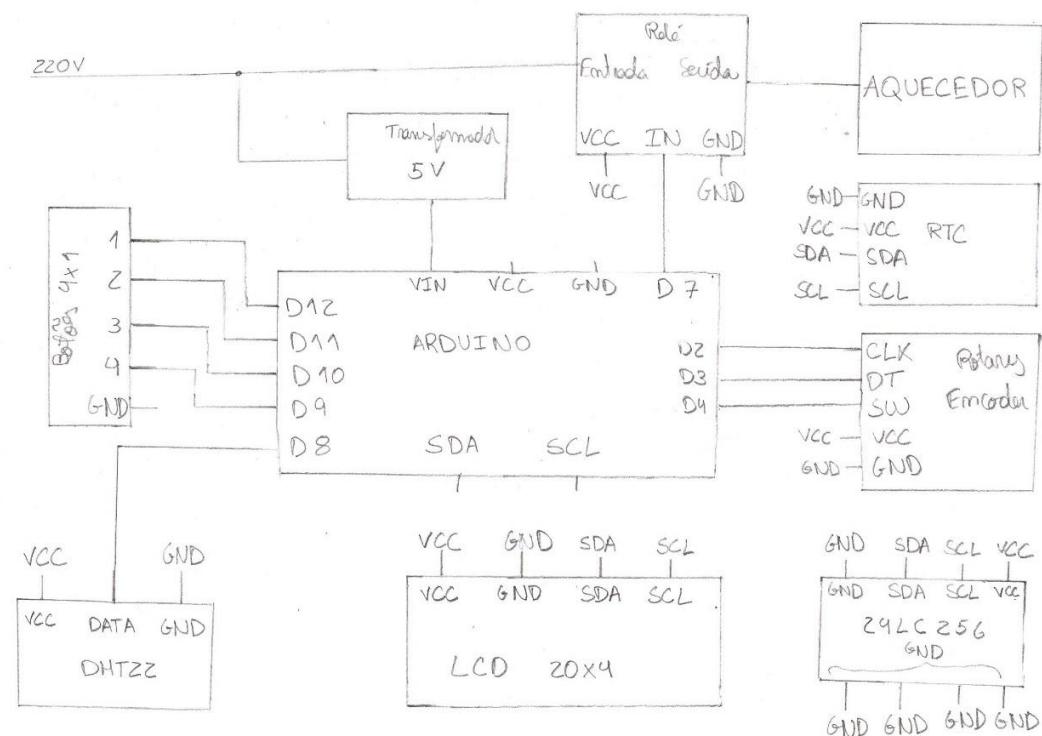
A interação com o dispositivo segue também o mesmo pressuposto. Um menu Inicial, onde data, hora, temperatura, humidade e estado do dispositivo são apresentados, recebe o utilizador que passa a poder navegar pelos diferentes menus: “Utilizadores”, “Config Data/Hora” e “Reset”, entre outros que pretendemos incluir no futuro. Do menu “Utilizadores” à configuração das suas

opções pessoais, basta clicar no seu nome ou, se quiser criar um novo utilizador, selecionar “NOVO UTILIZADOR”. Para selecionar estas opções terá ao seu dispor um LCD, 4 botões (Cima, Baixo, OK e Cancelar) e um rotary encoder como mostrado na figura abaixo.



Uma nova funcionalidade foi também criada, o modo temporizador. Neste modo acessível a partir do menu Inicial, permite ligar o aquecedor durante algum tempo, o dispositivo simplifica-se, e passa a funcionar como uma tomada temporizada que pode ser usada com qualquer dispositivo.

Esquema



No esquema atrás, estão apresentados os principais componentes deste projeto. O relé permite ao microcontrolador ligar ou desligar a tomada onde estaria ligado o. O módulo DS3231 é um RTC (Real Time Clock), que serve para que a hora não seja perdida sempre que se desliga o dispositivo ou há um corte de energia. O circuito integrado 24LC256 é uma memória EEPROM, onde são guardadas as configurações para cada utilizador. Decidimos utilizá-lo, pois, embora o Arduino tenha também uma, os seus 1024 bytes não seriam suficiente para versões posteriores do projeto. O RTC, a EEPROM e o LCD comunicam com o microcontrolador através da interface I²C (razão das ligações SDA e SCL). O DHT22 é um sensor de humidade e temperatura, que foi escolhido por ser relativamente barato e preciso. O rotary encoder serve para incrementar/decrementar valores ao ser rodado, para além de ter um botão embutido (que utilizamos como botão de confirmação).

Instruções

Preparação do dispositivo

1. Ligue ao dispositivo um aquecedor (ou outro aparelho no modo temporizador)
2. Ligue o dispositivo a uma tomada

Criar um novo utilizador

1. No menu Inicial, clique em “Opcoes”
2. No menu Opções, utilizando as setas ou o rotary encoder, clique em “Utilizadores”



3. No menu Utilizadores, clique em “NOVO UTILIZADOR” (esta opção pode não surgir caso tenha sido atingido o número máximo de utilizadores, sendo nesse caso apresentado “CHEIO”)



4. No menu Criar Utilizador, introduza o seu nome de acordo com os seguintes comandos:
 - a. Premir setas Cima/Baixo: alterna entre posições
 - b. Rodar o rotary encoder: alterna entre caracteres
 - c. Premir OK: grava e volta ao menu Utilizadores
 - d. Premir Cancelar: não grava e volta ao menu Utilizadores
 - e. Premir o rotary encoder: passa à posição seguinte, exceto quando num espaço em branco caso no qual grava e volta ao menu Utilizadores



Configurar hora para um determinado dia da semana

1. No menu Utilizadores, clique no seu nome de utilizador
2. No seu menu Utilizador, selecione o dia da semana que pretende



3. Introduza a hora pretendida de acordo com os seguintes comandos e tendo em conta as notas indicadas:
 - a. Premir seta Baixo: muda para campo Minuto
 - b. Premir seta Cima: muda para campo Hora
 - c. Rodar o rotary encoder: alterna entre valores de cada campo
 - d. Premir OK: grava e volta ao seu menu Utilizador
 - e. Premir Cancelar: Elimina o alarme e volta ao seu menu Utilizador
 - f. Premir o rotary encoder: quando no campo Hora, passa para o campo Minuto e, quando no campo Minuto, grava e volta ao menu Utilizador



Configurar temperatura desejada

A temperatura desejada pode variar entre os 15 e os 40°C, tendo como valor predefinido os 20°C.

1. No seu menu Utilizador, clique na temperatura apresentada



2. No menu Temperatura Desejada, rodando o rotary encoder, selecione a temperatura que deseja



3. Clicando no rotary encoder ou em OK, grava e volta ao menu Utilizador e, clicando em Cancelar, não grava e volta ao menu Utilizador

Configurar período de funcionamento pretendido

O período de funcionamento pretendido pode variar entre os 5 e os 120 minutos, tendo como valor predefinido os 30 minutos.

1. No seu menu Utilizador, clique no período de funcionamento apresentado
2. No menu Período de Funcionamento Desejado, rodando o rotary encoder, selecione o período que pretende



3. Clicando no rotary encoder ou em OK, grava e volta ao menu Utilizador e, clicando em Cancelar, não grava e volta ao menu Utilizador

Eliminar utilizador

1. No seu menu Utilizador, clique em “ELIMINAR”

Acertar data e hora

2. No menu Opções, selecione “Config. Data/Hora”
3. Introduza a data e hora pretendidas de acordo com os seguintes comandos:
 - a. Premir seta Cima/Baixo: alterna entre campos
 - b. Rodar o rotary encoder: alterna entre valores de cada campo
 - c. Premir OK: grava e volta ao seu menu Opções
 - d. Premir Cancelar: não grava e volta ao menu Opções
 - e. Premir o rotary encoder: passa ao campo seguinte, exceto no último campo no qual grava e volta ao menu Opções



Interromper aquecimento programado

- No menu Inicial, utilizando as setas ou o rotary encoder, clique em “Cancelar” (esta opção não está disponível quando o dispositivo não está neste modo)

Modo rápido

- No menu Inicial, utilizando as setas ou o rotary encoder, selecione “Modo rapido” (esta opção não está disponível quando o dispositivo está já neste modo ou em modo de aquecimento programado)
- Selecione o número de minutos que pretende tal como nos passos 2 e 3 de “Configurar período de funcionamento pretendido”

Interromper modo rápido

- No menu Inicial, clique em “Desligar” (esta opção não está disponível quando o dispositivo não está neste modo)

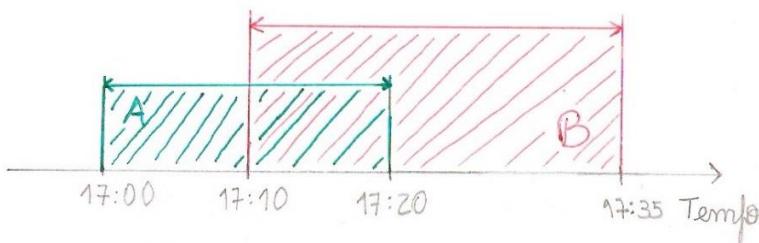
Restaurar dispositivo

- No menu Opções, selecione “RESET”
- No menu Reset, utilizando as setas ou o rotary encoder, escolha a opção que pretende
- Clicando no rotary encoder ou no botão OK, selecione essa opção

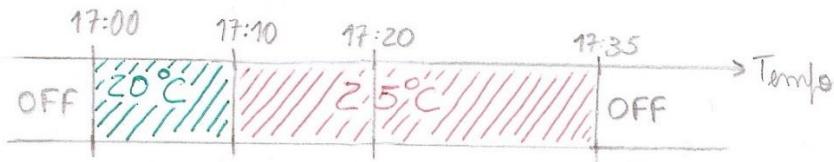
Situações problemáticas

Sobreposição de horários

Imagine-se que o utilizador A (cor verde) define para as 17:00 de segunda-feira um aquecimento a 20°C durante 20 minutos e que o utilizador B (cor vermelha) define para as 17:10 desse mesmo dia outro aquecimento a 25°C durante 25 minutos, como ilustrado no diagrama temporal abaixo.



Assim, entre as 17:10 (hora de início do aquecimento de B) e as 17:20 (hora de final do aquecimento de A) os horários de ambos sobrepõem-se. Tendo em conta exemplos semelhantes a este, foi decidido que, em momentos de sobreposição, o dispositivo deve implementar as condições do horário que termina mais tarde. Desta forma, a situação anterior daria origem ao ciclo de aquecimento esquematizado abaixo.



Outras opções como o impedimento do utilizador de definir um horário que se sobreponha a outro já existente foram também consideradas, tendo-se, no entanto, optado por esta pois respeita os horários de todos os utilizadores e é simples.

Inícios de aquecimento no dia anterior ao qual ele se aplica

Para a divisão já estar à temperatura pretendida à hora que o utilizador deseja, é necessário que o início do aquecimento se dê antes dessa hora. Imagine-se então que o utilizador A pretende ter a casa de banho quente às 00:02. Tendo em conta que o aquecedor se liga 5 minutos antes, o início do aquecimento deveria ser às 23:57 do dia anterior. Assim, o dispositivo, a partir das 23:55 (5 minutos antes do dia seguinte), começa a verificar se existem horários, em vez de no dia em que está, no dia seguinte.

Continuação do projeto

Antes de propormos o nosso projeto para o concurso, tivemos bastantes ideias e, à medida que o fomos realizando, ainda mais surgiram. Algumas implementámos, outras pensámos que conseguíamos mas não conseguimos e algumas nem pensámos em implementar. Todas elas, no entanto, alimentaram a nossa vontade de continuar este projeto depois da sua entrega.

Das que inicialmente considerámos, ligar o aquecedor tendo em conta um despertador no telemóvel é uma das que pretendemos vir a integrar. Para isso, ter-se-ia de substituir o Arduino por um microcontrolador ESP8266. As alterações ao código seriam mínimas, pois as bibliotecas associadas ao primeiro têm suporte em ambos. O código para a aplicação móvel comunicar com o microcontrolador seria relativamente simples da parte do segundo, tendo sido feito em parte num dos protótipos. A aplicação móvel que foi desenvolvida para Android estava ainda muito rudimentar, mas já lia a data e a hora do próximo despertador agendado e enviava-a ao microcontrolador através da plataforma Thinger.io (escolhida por simplificar o processo). Numa fase mais avançada do desenvolvimento, poderíamos permitir que a aplicação tentasse perceber se o utilizador vai dormir em casa (por exemplo, através do GPS), de modo a determinar se é realmente necessário aquecer a casa de banho.

Outra funcionalidade a acrescentar é a utilização de uma expressão matemática para, a partir de aquecimentos passados, o dispositivo prever o tempo de aquecimento necessário para elevar a casa de banho de uma temperatura pretendida. Analisando o tempo que demora a aquecer a casa de banho num dia, e tendo em conta a temperatura inicial e final, estimar-se-ia o tempo antes da hora agendada a que o aquecedor deve ligar e evitar-se-iam gastos desnecessários de energia. No entanto, esta tarefa pode ser dificultada devido a acontecimentos aleatórios que

corromperiam os dados, tais como a porta ou janela da casa de banho estarem abertas. A forma de cálculo deveria então ter em conta a sua possibilidade de ocorrência e saber contorná-los.

Mais ainda, gostaríamos de aprofundar o tema de casas inteligentes e integrar o nosso projeto numa rede com outros sensores, dispositivos e sistemas de domótica que já existem, bem como criarmos nós alguns que consideremos úteis. Arres condicionados ou sistemas de piso radiante são algumas ideias de passos seguintes que podemos tomar, de modo a aumentar o conforto das pessoas e tornar o uso de energia mais eficiente. Por exemplo, poder-se-ia fazer com que uma aplicação determine se o utilizador está a voltar a casa e arrefece-la num dia quente de verão.

```

1  /*
2   *                               WarmBeforeBath
3   *     José Neves      89683
4   *     Miguel Fazenda  90146
5   *     BitChallenge 2018
6  */
7
8 #include <dht.h>
9 #include <extEEPROM.h>
10 #include "estado.h"
11 #include "input.h"
12 #include "menus.h"
13
14 #define DHT_PIN 8
15 #define RELAY_PIN 7
16
17 #define BACKLIGHT_TIMEOUT_MILLIS 30000
18 #define MENU_TIMEOUT_MILLIS 120000
19
20 LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
21 DS3231 rtc;
22 extEEPROM memEEPROM(kbits_256, 2, 64);
23 dht DHT;
24
25 Menus menus;
26 Estado estado;
27 Input input;
28
29 int temp;
30 unsigned long lastUpdateTempMillis;
31 bool lcdState;
32
33 void rotaryEncoderInterrupt() {
34     input.rotaryEncoderInterrupt();
35 }
36
37 void setup() {
38     lcd.begin(LCD_COMP, LCD_ALT);
39
40     // Inicialização dos botões e rotary encoder
41     input.init();
42     attachInterrupt(digitalPinToInterruption(2), rotaryEncoderInterrupt, CHANGE);
43
44     // Inicialização do relay
45     pinMode(RELAY_PIN, OUTPUT);
46     digitalWrite(RELAY_PIN, HIGH);
47
48     menus.init();
49
50     lastUpdateTempMillis = 0;
51     lcdState = true; // A backlight deve estar ligada ao inicio
52 }
53
54 void loop() {
55     unsigned long millisTime = millis();
56     bool backlightShouldBeOn;
57
58     input.update(&lcd);
59
60     // De dois em dois segundos
61     if (millisTime - lastUpdateTempMillis > 2000) {
62         // Lê a temperatura e a umidade no sensor
63         DHT.read22(DHT_PIN);
64
65         // Atualiza o estado do aquecedor
66         estado.loop(float(DHT.temperature), &rtc);
67
68         // Liga ou desliga o relé

```

```
69     digitalWrite(RELAY_PIN, !estado.isAquecedorLigado);
70
71     // Atualiza lastUpdateMillis
72     lastUpdateTempMillis = millisTime;
73 }
74
75 // Atualizar os menus
76 menus.loop(&DHT);
77
78 // Verifica se passou o tempo necessário para desligar a backlight desde a
79 // ultima interação do utilizador
80 backlightShouldBeOn =
81     millisTime < BACKLIGHT_TIMEOUT_MILLIS + input.lastInputChangeMillis;
82
83 // Liga ou desliga a backlight se for necessário
84 if (lcdState != backlightShouldBeOn) {
85     lcdState = backlightShouldBeOn;
86     lcd.setBacklight(lcdState);
87 }
88
89 // Passado algum tempo sem interação
90 if (menus.menuId != MENU_INICIAL &&
91     (millisTime > (input.lastInputChangeMillis + MENU_TIMEOUT_MILLIS)))
92     menus.setMenu(MENU_INICIAL); // Volta ao menu inicial
93 }
```

```

1  /*
2   * menus.h - Comunicação usuário-dispositivo
3   * Criada por José Neves e Miguel Fazenda, 5 de fevereiro de 2018
4   * BitChallenge 2018
5  */
6
7 #ifndef MENUS_H
8 #define MENUS_H
9
10 #include <dht.h>
11 #include "estado.h"
12 #include "memoria.h"
13 #include "opcoes.h"
14
15 #define MENU_INICIAL 0
16 #define MENU_OPCOES 1
17 #define MENU_TEMP_DESEJADA 2
18 #define MENU_SET_USER_NUM_MIN 3
19 #define MENU_HORA_DESEJADA 4
20 #define MENU_UTILIZADORES 5
21 #define MENU_UTILIZADOR 6
22 #define MENU_CONFIG_DATA_HORA 7
23 #define MENU_CRIAR_UTILIZADOR 8
24 #define MENU_MODO_RAPIDO 9
25 #define MENU_RESET_CONFIRMAR 10
26
27 #define OPCOES 0
28 #define MODO_RAPIDO 1
29 #define MUDAR_USER_TEMP 8
30 #define MUDAR_USER_NUM_MIN 9
31 #define ELIMINAR_USER 10
32
33 #define DELAY_TIME 1000
34 #define PISCAR_CAMPO millis() % 800 < 700
35 // O campo acenderá durante 700ms por cada 800ms
36
37 class Menus {
38 public:
39     uint8_t menuId;
40     uint8_t lastMenuItem;
41     uint8_t campoAtual;
42
43     DATE data;
44     HORA hora;
45     uint8_t dow;
46
47     Opcoes opcoes;
48
49     char* strIns = NULL;
50
51     // Gerais
52     void init();
53     void loop(dht* DHT);
54     void setMenu(int menu);
55     void setMenuBack();
56     void printFuncionalidadeFutura();
57
58     // Home
59     void menuInicialInit();
60     void menuInicialLoop(dht* DHT);
61
62     // Opções Principais
63     void menuOpcoesInit();
64     void menuOpcoesLoop();
65
66     // Seleção de Variáveis
67     void menuTempDesejadaInit();
68     void menuTempDesejadaLoop();

```

```
69
70     void menuNumMinutosDesejadoInit();
71     void menuNumMinutosDesejadoLoop();
72
73     void menuHoraDesejadaInit();
74     void menuHoraDesejadaLoop();
75
76 // Utilizadores
77     void menuUtilizadoresInit();
78     void menuUtilizadoresLoop();
79
80     uint8_t utilizadorSelecionado;
81     void menuUtilizadorInit();
82     void menuUtilizadorLoop();
83
84     void menuCriarUtilizadorInit();
85     void menuCriarUtilizadorLoop();
86
87 // Acerta Relógio
88     void menuConfigDataHoraInit();
89     void menuConfigDataHoraLoop();
90
91 // Reset
92     void menuResetConfirmarInit();
93     void menuResetConfirmarLoop();
94 };
95
96 extern LiquidCrystal_I2C lcd;
97 extern Menus menus;
98 extern DS3231 rtc;
99 extern Estado estado;
100
101 #endif
102
```

```

1  /*
2   * menus.cpp - Comunicação usuário-dispositivo
3   * Criada por José Neves e Miguel Fazenda, 5 de fevereiro de 2018
4   * BitChallenge 2018
5  */
6
7 #include "menus.h"
8
9 extern LiquidCrystal_I2C lcd;
10 extern Menus menus;
11 extern DS3231 rtc;
12 extern Estado estado;
13 extern Input input;
14
15 // Gerais
16 void Menus::init() {
17     uint8_t setaVerticalMap[] = SETA_VERTICAL_MAP;
18     lcd.createChar(SETA_VERTICAL, setaVerticalMap);
19
20     opcoes = Opcoes();
21     lastMenuItem = MENU_INICIAL;
22
23     setMenu(MENU_INICIAL);
24 }
25 void Menus::loop(dht* DHT) {
26     switch (menuId) {
27         case MENU_INICIAL:
28             menus.menuInicialLoop(DHT);
29             break;
30         case MENU_OPCOES:
31             menus.menuOpcoesLoop();
32             break;
33         case MENU_HORA_DESEJADA:
34             menus.menuHoraDesejadaLoop();
35             break;
36         case MENU_UTILIZADORES:
37             menuUtilizadoresLoop();
38             break;
39         case MENU_UTILIZADOR:
40             menuUtilizadorLoop();
41             break;
42         case MENU_TEMP_DESEJADA:
43             menus.menuTempDesejadaLoop();
44             break;
45         case MENU_CRIAR_UTILIZADOR:
46             menuCriarUtilizadorLoop();
47             break;
48         case MENU_CONFIG_DATA_HORA:
49             menuConfigDataHoraLoop();
50             break;
51         case MENU_SET_USER_NUM_MIN:
52             menuNumMinutosDesejadoLoop();
53             break;
54         case MENU_MODO_RAPIDO:
55             menuNumMinutosDesejadoLoop();
56             break;
57         case MENU_RESET_CONFIRMAR:
58             menuResetConfirmarLoop();
59     }
60 }
61 void Menus::setMenu(int menu) {
62     lcd.clear();
63
64     lastMenuItem = menuItem;
65     menuItem = menu;
66
67     switch (menu) {
68         case MENU_INICIAL:

```

```

69     menuInicialInit();
70     break;
71 case MENU_OPCOES:
72     menuOpcoesInit();
73     break;
74 case MENU_HORA_DESEJADA:
75     menuHoraDesejadaInit();
76     break;
77 case MENU_TEMP_DESEJADA:
78     menuTempDesejadaInit();
79     break;
80 case MENU_UTILIZADORES:
81     menuUtilizadoresInit();
82     break;
83 case MENU_UTILIZADOR:
84     menuUtilizadorInit();
85     break;
86 case MENU_CRIAR_UTILIZADOR:
87     menuCriarUtilizadorInit();
88     break;
89 case MENU_CONFIG_DATA_HORA:
90     menuConfigDataHoraInit();
91     break;
92 case MENU_SET_USER_NUM_MIN:
93     menuNumMinutosDesejadoInit();
94     break;
95 case MENU_MODO_RAPIDO:
96     menuNumMinutosDesejadoInit();
97     break;
98 case MENU_RESET_CONFIRMAR:
99     menuResetConfirmarInit();
100 }
101 }
102 void Menus::setMenuBack() {
103     setMenu(lastMenuItem);
104 }
105 void Menus::printFuncionalidadeFutura() {
106     lcd.clear();
107     lcd.setCursor(0, 1);
108     lcd.print(F("Funcionalidade"));
109     lcd.setCursor(0, 2);
110     lcd.print(F("Futura"));
111     delay(DELAY_TIME);
112     setMenu(MENU_OPCOES);
113 }
114
// Menu Inicial
116 void menuInicialDrawEstado() {
117     lcd.setCursor(0, 2); // Coloca o cursor no início da 3a fila
118     switch (estado.estadoCodigo) {
119         case ESTADO_NADA:
120             lcd.print(F("          "));
121             lcd.setCursor(9, 3); // Coloca o cursor no lugar da segunda opção
122             lcd.print(F("Modo Rapido"));
123             break;
124         case ESTADO_TEMPORIZADOR:
125             lcd.print(F("Ate as "));
126             printHora(estado.horaDeParagem.hour(), estado.horaDeParagem.minute(),
127                         NAO_PISCAR, false);
128             lcd.print(F("      "));
129             lcd.setCursor(9, 3); // Coloca o cursor no lugar da segunda opção
130             lcd.print(F("Desligar   "));
131             break;
132         case ESTADO_AQUECE_E_MANTEM:
133             lcd.print(F("Ate as "));
134             printHora(estado.horaDeParagem.hour(), estado.horaDeParagem.minute(),
135                         NAO_PISCAR, false);
136             lcd.print(F(" a "));

```

```

137     printTemperature(estado.tempAManter);
138     lcd.setCursor(9, 3); // Coloca o cursor no lugar da segunda opção
139     lcd.print(F("Cancelar    "));
140     break;
141 }
142 }
143 void Menus::menuInicialInit() {
144     campoAtual = OPCOES;
145
146     menuInicialDrawEstado();
147 }
148 void Menus::menuInicialLoop(dht* DHT) {
149     bool a;
150
151     lcd.setCursor(2, 0);
152     printDoW(rtc.getDoW(), NAO_PISCAR, false);
153     lcd.print(F(" "));
154     printData(rtc.getYear(), rtc.getMonth(a), rtc.getDate(), NAO_PISCAR, false);
155     lcd.setCursor(0, 1);
156     printTemperature(DHT->temperature);
157     lcd.setCursor(7, 1);
158     printHora(rtc.getHour(a, a), rtc.getMinute(), NAO_PISCAR, false);
159     lcd.setCursor(17, 1);
160     printHumidity(DHT->humidity);
161     lcd.setCursor(campoAtual == OPCOES ? 0 : 8, 3);
162     lcd.write(SETA_HORIZONTAL);
163     lcd.setCursor(1, 3);
164     lcd.print(F("Opcoes"));
165
166     if (estado.estadoMudou) {
167         menuInicialDrawEstado();
168         estado.estadoMudou = false;
169     }
170
171     if (input.checkCliqueBotao(BOTA0_BAIXO) ||
172         input.rotationEvent == ROTATION_RIGHT) {
173         campoAtual = MODO_RAPIDO;
174         lcd.setCursor(0, 3);
175         lcd.print(F(" ")); // Apagar a seta
176     }
177     if (input.checkCliqueBotao(BOTA0_CIMA) ||
178         input.rotationEvent == ROTATION_LEFT) {
179         campoAtual = OPCOES;
180         lcd.setCursor(8, 3);
181         lcd.print(F(" ")); // Apagar a seta
182     }
183     if (input.checkCliqueBotao(BOTA0_OK) ||
184         input.checkCliqueBotao(BOTA0_ROTARY)) {
185         if (campoAtual == OPCOES)
186             setMenu(MENU_OPCOES);
187         else {
188             if (estado.estadoCodigo == ESTADO_NADA)
189                 setMenu(MENU_MODO_RAPIDO);
190             else {
191                 estado.estadoCodigo = ESTADO_NADA;
192                 estado.estadoMudou = true;
193             }
194         }
195     }
196 }
197
198 // Opções principais
199 void Menus::menuOpcoesInit() {
200     opcoes.setOpcoes(6);
201
202     opcoes.setConstOption(0, "Utilizadores");
203     opcoes.setConstOption(1, "Config Data/Hora");
204     opcoes.setConstOption(2, "Configurar Wi-fi");

```

```

205     opcoes.setConstOption(3, "Idioma");
206     opcoes.setConstOption(4, "Modo de desligar");
207     opcoes.setConstOption(5, "RESET");
208
209     opcoes.printOpcoes();
210 }
211
212 void Menus::menuOpcoesLoop() {
213     opcoes.slideThroughOpcoes();
214
215     if (input.checkCliqueBotao(BOTA0_OK) || input.checkCliqueBotao(BOTA0_ROTARY))
216         switch (opcoes.opcaoAtual) {
217             case 0:
218                 setMenu(MENU_UTILIZADORES);
219                 break;
220             case 1:
221                 setMenu(MENU_CONFIG_DATA_HORA);
222                 break;
223             case 5:
224                 setMenu(MENU_RESET_CONFIRMAR);
225                 break;
226             default:
227                 printFuncionalidadeFutura();
228                 break;
229         }
230     if (input.checkCliqueBotao(BOTA0_CANCELAR))
231         setMenu(MENU_INICIAL);
232 }
233
234 // Seleção de Variáveis
235 void Menus::menuTempDesejadaInit() {
236     // Reutilização de variáveis para poupança de memória
237     campoAtual = getSelectedUser(utilizadorSelecionado).temperature;
238 }
239 void Menus::menuTempDesejadaLoop() {
240     lcd.setCursor(7, 1);
241     printTemperature(campoAtual);
242
243     input.rotaryAlteraVar(&campoAtual, MAX_TEMP, MIN_TEMP);
244
245     if (input.checkCliqueBotao(BOTA0_OK) ||
246         input.checkCliqueBotao(BOTA0_ROTARY)) {
247         EEPROM_USER_DATA user = getSelectedUser(utilizadorSelecionado);
248         user.temperature = campoAtual;
249         setUserData(user);
250         setMenuBack();
251     }
252
253     if (input.checkCliqueBotao(BOTA0_CANCELAR))
254         setMenuBack();
255 }
256 void Menus::menuNumMinutosDesejadoInit() {
257     // Reutilização de variáveis para poupança de memória
258     if (menuId == MENU_SET_USER_NUM_MIN)
259         campoAtual = getSelectedUser(utilizadorSelecionado).numMin;
260     else if (menuId == MENU_MODE_RAPIDO)
261         campoAtual = 30;
262 }
263 void Menus::menuNumMinutosDesejadoLoop() {
264     lcd.setCursor(7, 1);
265     print_Int(campoAtual, 3);
266     lcd.print(F("min"));
267
268     input.rotaryAlteraVar(&campoAtual, 120, 5);
269
270     if (input.checkCliqueBotao(BOTA0_OK) ||
271         input.checkCliqueBotao(BOTA0_ROTARY)) {
272         if (menuId == MENU_SET_USER_NUM_MIN) {

```

```

273     EEPROM_USER_DATA user = getSelectedUser(utilizadorSelecionado);
274     user.numMin = campoAtual;
275     setUserData(user);
276     setMenuBack();
277 } else if (menuId == MENU_MODO_RAPIDO) {
278     estado.setEstadoTemporizador(&rtc, campoAtual);
279     setMenuBack();
280 }
281 }
282
283 if (input.checkCliqueBotao(BOTA0_CANCELAR))
284     setMenuBack();
285 }
286
287 void Menus::menuHoraDesejadaInit() {
288     hora = getSelectedUser(utilizadorSelecionado).horaPorDiaSemana[dow];
289
290     if (hora.hora == 25) {
291         hora.hora = 0;
292         hora.minuto = 0;
293     }
294
295     campoAtual = CAMPO_HORA;
296 }
297 void leaveMenuHoraIniciar(HORA hora) {
298     EEPROM_USER_DATA user = getSelectedUser(menus.utilizadorSelecionado);
299     user.horaPorDiaSemana[menus.dow] = hora;
300     setUserData(user);
301     menus.setMenu(MENU_UTILIZADOR);
302 }
303 void Menus::menuHoraDesejadaLoop() {
304     bool aceso = PISCAR_CAMPO;
305
306     lcd.setCursor(8, 0);
307     printHora(hora.hora, hora.minuto, campoAtual, aceso);
308     lcd.setCursor(0, 1);
309     lcd.print(F("OK - Sai e Guarda"));
310     lcd.setCursor(0, 2);
311     lcd.print(F("C - Sai e elimina"));
312     lcd.setCursor(4, 3);
313     lcd.print(F("alarme"));
314
315     if (input.checkCliqueBotao(BOTA0_BAIXO))
316         campoAtual = CAMPO_MINUTO;
317     if (input.checkCliqueBotao(BOTA0_CIMA))
318         campoAtual = CAMPO_HORA;
319     if (input.checkCliqueBotao(BOTA0_OK))
320         leaveMenuHoraIniciar(hora);
321     if (input.checkCliqueBotao(BOTA0_CANCELAR))
322         leaveMenuHoraIniciar(HORA{25, 25});
323
324     if (input.checkCliqueBotao(BOTA0_ROTARY)) {
325         if (campoAtual == CAMPO_HORA)
326             campoAtual = CAMPO_MINUTO;
327         else if (campoAtual == CAMPO_MINUTO)
328             leaveMenuHoraIniciar(hora);
329     }
330     if (campoAtual == CAMPO_HORA)
331         input.rotaryAlteraVar(&hora.hora, 23, 0);
332     else if (campoAtual == CAMPO_MINUTO)
333         input.rotaryAlteraVar(&hora.minuto, 59, 0);
334 }
335
336 // UTILIZADORES
337 void Menus::menuUtilizadoresInit() {
338     EEPROM_USER_DATA users[MAX_USERS];
339     uint8_t usersCount = getAllUsersData(users);
340 }
```

```

341     opcoes.setOpcoes(usersCount + 1);
342
343     for (uint8_t i = 0; i < usersCount; i++)
344         opcoes.setDynOptionCpy(i, users[i].userName, strlen(users[i].userName));
345
346     opcoes.setConstOption(usersCount,
347                           usersCount == MAX_USERS ? "CHEIO" : "NOVO UTILIZADOR");
348
349     opcoes.printOpcoes();
350 }
351
352 void Menus::menuUtilizadoresLoop() {
353     opcoes.slideThroughOpcoes();
354
355     if (input.checkCliqueBotao(BOTA0_OK) ||
356         input.checkCliqueBotao(BOTA0_ROTARY)) {
357         // Caso seja a última opção a selecionada (NOVO UTILIZADOR)
358         if (opcoes.opcaoAtual == opcoes.numOpcoes - 1) {
359             EEPROM_USER_DATA users[MAX_USERS];
360             uint8_t usersCount = getAllUsersData(users);
361
362             if (usersCount < MAX_USERS)
363                 setMenu(MENU_CRIAR_UTILIZADOR);
364         } else {
365             selecionado = opcoes.opcaoAtual;
366             setMenu(MENU_UTILIZADOR);
367         }
368     }
369     if (input.checkCliqueBotao(BOTA0_CANCELAR))
370         setMenu(MENU_OPCOES);
371 }
372
373 void Menus::menuUtilizadorInit() {
374     String buf;
375
376     EEPROM_USER_DATA user = getSelectedUser(selecionado);
377
378     opcoes.setOpcoes(11);
379
380     // Cria as strings para a opção da temperatura e dos minutos
381     buf = String(user.temperature);
382     buf.concat(char(223));
383     buf.concat("C");
384     opcoes.setDynOptionCpy(MUDAR_USER_TEMP, (char*)buf.c_str(), buf.length());
385
386     buf = String(user.numMin);
387     buf.concat("min");
388     opcoes.setDynOptionCpy(MUDAR_USER_NUM_MIN, (char*)buf.c_str(), buf.length());
389
390     opcoes.setDynOptionCpy(0, user.userName, strlen(user.userName));
391     for (uint8_t d = 0; d < 7; d++) {
392         int8_t hora = user.horaPorDiaSemana[d].hora;
393         int8_t minuto = user.horaPorDiaSemana[d].minuto;
394
395         buf = getMes_ou_Dow(d, DOW);
396         if (hora == 25) {
397             buf.concat(" Sem alarme");
398         } else {
399             buf.concat(" ");
400             buf.concat(Int_To_Str(hora, 2));
401             buf.concat(":");
402             buf.concat(Int_To_Str(minuto, 2));
403         }
404
405         opcoes.setDynOptionCpy(d + 1, (char*)buf.c_str(), buf.length());
406     }
407     opcoes.setConstOption(ELIMINAR_USER, "ELIMINAR");
408 }
```

```

409     opcoes.printOpcoes();
410     dow = 0;
411 }
412 void Menus::menuUtilizadorLoop() {
413     opcoes.slideThroughOpcoes();
414
415     if (input.checkCliqueBotao(BOTA0_OK) ||
416         input.checkCliqueBotao(BOTA0_ROTARY)) {
417         switch (opcoes.opcaoAtual) {
418             // As outras opções que existem no menu são dias da semana
419             case 0:
420                 break;
421             default:
422                 dow = opcoes.opcaoAtual - 1;
423                 setMenu(MENU_HORA_DESEJADA);
424                 break;
425             case MUDAR_USER_TEMP:
426                 setMenu(MENU_TEMP_DESEJADA);
427                 break;
428             case MUDAR_USER_NUM_MIN:
429                 setMenu(MENU_SET_USER_NUM_MIN);
430                 break;
431             case ELIMINAR_USER:
432                 removeUserData(opcoes.listaOpcoes[0]);
433                 setMenu(MENU_UTILIZADORES);
434                 break;
435         }
436     }
437     if (input.checkCliqueBotao(BOTA0_CANCELAR))
438         setMenu(MENU_UTILIZADORES);
439 }
440
441 void Menus::menuCriarUtilizadorInit() {
442     if (strIns != NULL) {
443         delete[] strIns;
444         strIns = NULL;
445     }
446     if (strIns == NULL)
447         strIns = new char[DIM_USERNAME + 1];
448
449     for (uint8_t i = 0; i <= DIM_USERNAME + 1; i++)
450         strIns[i] = '\0';
451
452     strIns[0] = 'A';
453
454     campoAtual = 0;
455
456     lcd.setCursor(0, 3);
457     lcd.print(F("Nao pode ter espacos"));
458 }
459 void leaveMenuCriarUtilizador(char* strIns[]) {
460     delete[](*strIns);
461     *strIns = NULL;
462     menus.setMenu(MENU_UTILIZADORES);
463 }
464 // Guarda o novo utilizador. Caso já exista devolve false
465 bool saveNovoUtilizador(char strIns[]) {
466     EEPROM_USER_DATA users[MAX_USERS];
467     uint8_t usersCount = getAllUsersData(users);
468
469     for (uint8_t i = 0; i < usersCount; i++) {
470         if (strcmp(users[i].userName, strIns) == 0) {
471             lcd.clear();
472             lcd.home();
473             lcd.print(F("Nome ja utilizado"));
474             delay(Delay_Time);
475             lcd.clear();
476             return false;

```

```

477     }
478 }
479
480 HORA a = HORA{25, 25};
481 EEPROM_USER_DATA userData = {"", {a, a, a, a, a, a}, 20, 30};
482
483 strcpy(userData.userName, strIns);
484 setUserData(userData);
485 return true;
486 }
487 void Menus::menuCriarUtilizadorLoop() {
488 lcd.setCursor(2, 0);
489 printUserName(strIns);
490 lcd.setCursor(2 + campoAtual, 1);
491 lcd.write(SETA_VERTICAL);
492
493 // Se se estiver na primeira posição
494 if (campoAtual == 0)
495     // Altera a letra selecionada de A a Z
496     input.rotaryAlteraVar((uint8_t*)(strIns + campoAtual), 'Z', 'A');
497 else {
498     // Altera a letra selecionada de A a Z e espaço branco
499     input.rotaryAlteraVar((uint8_t*)(strIns + campoAtual), 'Z', 'A' - 1);
500     // Se passou para cima do espaço vazio
501     if (strIns[campoAtual] == '\0' + 1)
502         strIns[campoAtual] = 'A'; // Coloca um A
503     // Se chegou ao caracter depois do Z ou antes do A
504     else if (strIns[campoAtual] < 'A')
505         strIns[campoAtual] = '\0'; // Mete um espaço vazio
506 }
507
508 if (input.checkCliqueBotao(BOTA_O_ROTARY)) {
509     // Se estiver num espaço em branco ou na última posição
510     if (strIns[campoAtual] == '\0' || campoAtual == DIM_USERNAME - 1) {
511         // Se conseguir criar um novo utilizador
512         if (saveNovoUtilizador(strIns))
513             leaveMenuCriarUtilizador(&strIns);
514     } else {
515         campoAtual = strIns[campoAtual] == '\0' || campoAtual == DIM_USERNAME - 1
516             ? campoAtual
517             : campoAtual + 1; // Avança a seta uma posição
518         lcd.setCursor(2 + campoAtual - 1,
519                     1); // Coloca o cursor na posição anterior da seta
520         lcd.print(" "); // Apaga a seta
521     }
522 }
523
524 if (input.checkCliqueBotao(BOTA_O_CIMA)) {
525     campoAtual = campoAtual == 0 ? 0 : campoAtual - 1;
526     lcd.setCursor(2 + campoAtual + 1, 1);
527     lcd.print(" ");
528 }
529 if (input.checkCliqueBotao(BOTA_O_BAIXO)) {
530     campoAtual = strIns[campoAtual] == '\0' || campoAtual == DIM_USERNAME - 1
531         ? campoAtual
532         : campoAtual + 1;
533     lcd.setCursor(2 + campoAtual - 1, 1);
534     lcd.print(" ");
535 }
536
537 if (input.checkCliqueBotao(BOTA_O_CANCELAR))
538     leaveMenuCriarUtilizador(&strIns);
539
540 if (input.checkCliqueBotao(BOTA_O_OK)) {
541     if (saveNovoUtilizador(strIns)) {
542         // Caso o utilizador tenha sido criado com sucesso sai do menu
543         leaveMenuCriarUtilizador(&strIns);
544     }

```

```

545     }
546 }
547
548 // Acerto do Relógio
549 void Menus::menuConfigDataHoraInit() {
550     bool a;
551
552     data = {rtc.getYear(), rtc.getMonth(a), rtc.getDate()};
553     hora = {rtc.getHour(a, a), rtc.getMinute()};
554     dow = rtc.getDoW();
555
556     campoAtual = CAMPO_DOW;
557 }
558 void leaveSaveConfigDataHora(DATE* data, HORA* hora, uint8_t dow) {
559     if (verificaDataHora(data, hora)) {
560         rtc.setYear(data->ano);
561         rtc.setMonth(data->mes);
562         rtc.setDate(data->dia);
563         rtc.setHour(hora->hora);
564         rtc.setMinute(hora->minuto);
565         rtc.setDoW(dow);
566         rtc.setSecond(0);
567
568         menus.setMenu(MENU_OPCOES);
569     } else {
570         lcd.clear();
571         lcd.setCursor(0, 1);
572         lcd.print(F("Data e Hora"));
573         lcd.setCursor(0, 2);
574         lcd.print(F("Impossíveis"));
575         delay(DELAY_TIME);
576         lcd.clear();
577     }
578 }
579 void Menus::menuConfigDataHoraLoop() {
580     bool aceso = PISCAR_CAMPO;
581
582     lcd.setCursor(2, 1);
583     printDoW(dow, campoAtual, aceso);
584     lcd.print(F(" "));
585     printData(data.ano, data.mes, data.dia, campoAtual, aceso);
586     lcd.setCursor(6, 2);
587     printHora(hora.hora, hora.minuto, campoAtual, aceso);
588
589     switch (campoAtual) {
590     case CAMPO_ANO:
591         input.rotaryAlteraVar(&data.ano, 50, 18);
592         break;
593     case CAMPO_MES:
594         input.rotaryAlteraVar(&data.mes, 12, 1);
595         break;
596     case CAMPO_DIA:
597         input.rotaryAlteraVar(&data.dia, 31, 1);
598         break;
599     case CAMPO_HORA:
600         input.rotaryAlteraVar(&hora.hora, 23, 0);
601         break;
602     case CAMPO_MINUTO:
603         input.rotaryAlteraVar(&hora.minuto, 59, 0);
604         break;
605     case CAMPO_DOW:
606         input.rotaryAlteraVar(&dow, 6, 0);
607     }
608     if (input.checkCliqueBotao(BOTAO_ROTARY)) {
609         if (campoAtual != CAMPO_MINUTO)
610             campoAtual++;
611         else
612             leaveSaveConfigDataHora(&data, &hora, dow);

```

```

613 }
614
615 if (input.checkCliqueBotao(BOTA0_BAIXO))
616     campoAtual = campoAtual < CAMPO_MINUTO ? campoAtual + 1 : CAMPO_MINUTO;
617 if (input.checkCliqueBotao(BOTA0_CIMA))
618     campoAtual = campoAtual > 0 ? campoAtual - 1 : CAMPO_DOW;
619 if (input.checkCliqueBotao(BOTA0_OK))
620     leaveSaveConfigDataHora(&data, &hora, dow);
621 if (input.checkCliqueBotao(BOTA0_CANCELAR))
622     setMenu(MENU_OPCOES);
623 }
624
625 // Menu Reset
626 void Menus::menuResetConfirmarInit() {
627     campoAtual = OPCOES; // opcao "NAO" selecionada por predefinicao
628     lcd.setCursor(0, 3);
629     lcd.write(SETA_HORIZONTAL);
630 }
631
632 void Menus::menuResetConfirmarLoop() {
633     lcd.setCursor(3, 0);
634     lcd.print(F("Tem a certeza?"));
635     lcd.setCursor(0, 1);
636     lcd.print(F("(Apagar as config.)"));
637
638     // Imprimir as opcoes
639     lcd.setCursor(1, 3);
640     lcd.print(F("Nao"));
641
642     lcd.setCursor(9, 3);
643     lcd.print(F("Sim"));
644
645     if (input.checkCliqueBotao(BOTA0_CIMA) ||
646         input.rotationEvent == ROTATION_LEFT) {
647         campoAtual = MODO_RAPIDO;
648         lcd.setCursor(0, 3);
649         lcd.print(F(" ")); // Apagar a seta
650         lcd.setCursor(8, 3);
651         lcd.write(SETA_HORIZONTAL);
652     }
653     if (input.checkCliqueBotao(BOTA0_BAIXO) ||
654         input.rotationEvent == ROTATION_RIGHT) {
655         campoAtual = OPCOES;
656         lcd.setCursor(8, 3);
657         lcd.print(F(" ")); // Apagar a seta
658         lcd.setCursor(0, 3);
659         lcd.write(SETA_HORIZONTAL);
660     }
661     if (input.checkCliqueBotao(BOTA0_OK) ||
662         input.checkCliqueBotao(BOTA0_ROTARY)) {
663         if (campoAtual == MODO_RAPIDO) {
664             // Caso seja selecionado a opcao SIM
665             setMenu(MENU_INICIAL);
666             resetEEPROM();
667         } else {
668             setMenu(MENU_OPCOES);
669         }
670     }
671 }

```

```

1  /*
2   estado.h - Gestão do funcionamento do aquecedor
3   Criada por José Neves e Miguel Fazenda, 18 de março de 2018
4   BitChallenge 2018
5  */
6
7 #ifndef ESTADO_H
8 #define ESTADO_H
9
10 #include <DS3231.h>
11 #include "memoria.h"
12 #include "relogio.h"
13
14 #define ESTADO_NADA 1
15 #define ESTADO_TEMPORIZADOR 2
16 #define ESTADO_AQUECE_E_MANTEM 3
17
18 // Metade da dimensão do intervalo de temperaturas centrado na tempAManter e no
19 // qual o aquecedor se encontra desligado
20 #define PRECISAO_TEMP 2
21
22 class Estado {
23 public:
24     bool isAquecedorLigado = false;
25     bool estadoMudou = false;
26     uint8_t estadoCodigo = ESTADO_NADA;
27     uint8_t tempAManter;
28     uint8_t minutoAnterior;
29     DateTime horaDeParagem;
30
31     void setEstadoTemporizador(DS3231* rtc, uint8_t minutos);
32     void comparaHoraELigaAquecedor(DS3231* rtc,
33                                     HORA* rtcHora,
34                                     HORA* hora,
35                                     uint8_t minutos,
36                                     uint8_t temperatura);
37     void checkHoraDeAquecer(DS3231* rtc);
38
39     void estadoNada(DS3231* rtc);
40     void estadoTemporizador(DS3231* rtc);
41     void estadoAqueceMantem(float temp, DS3231* rtc);
42     void loop(float temp, DS3231* rtc);
43
44 private:
45     void mantemTemp(float temp);
46 };
47
48 #endif
49

```

```

1  /*
2   estado.cpp - Gestão do funcionamento do aquecedor
3   Criada por José Neves e Miguel Fazenda, 5 de fevereiro de 2018
4   BitChallenge 2018
5 */
6
7 #include "estado.h"
8
9 #define MINUTOS_ANTES_LIGAR_AQUECEDOR 5
10
11 DateTime calculaHoraDeParagem(DS3231* rtc, uint8_t minutos) {
12     // 1 - Converte a hora e data da classe DS3231 na classe DateTime
13     // 2 - Retira o respetivo tempo UNIX
14     // 3 - Soma-lhe o número de segundos correspondente aos minutos
15     // 4 - Constrói uma nova classe DateTime a partir do novo tempo UNIX
16     return DateTime(converte_DS3231(rtc).unixtime() +
17                     (minutos + MINUTOS_ANTES_LIGAR_AQUECEDOR) * 60);
18 }
19
20 void Estado::setEstadoTemporizador(DS3231* rtc, uint8_t minutos) {
21     estadoCodigo = ESTADO_TEMPORIZADOR;
22     estadoMudou = true;
23     horaDeParagem = calculaHoraDeParagem(rtc, minutos);
24 }
25
26 void Estado::comparaHoraELigaAquecedor(DS3231* rtc,
27                                         HORA* rtcHora,
28                                         HORA* hora,
29                                         uint8_t numMinutos,
30                                         uint8_t temperatura) {
31     if (rtcHora->hora == hora->hora && rtcHora->minuto == hora->minuto) {
32         DateTime novaHoraDeParagem = calculaHoraDeParagem(rtc, numMinutos);
33
34         // Se se está no estado NADA (o aquecedor não está ligado) ou
35         // se o aquecedor está ligado, chegou à hora de início do utilizador i e a
36         // hora de paragem deste é posterior à que estava agendada, define a nova
37         // hora de paragem do aquecedor
38         if (estadoCodigo == ESTADO_NADA ||
39             (novaHoraDeParagem.unixtime() > horaDeParagem.unixtime())) {
40             estadoCodigo = ESTADO_AQUECE_E_MANTEM;
41             estadoMudou = true;
42             tempAManter = temperatura;
43             horaDeParagem = novaHoraDeParagem;
44         }
45     }
46 }
47
48 // Verifica se chegou a alguma hora agendada para ligar o aquecedor
49 void Estado::checkHoraDeAquecer(DS3231* rtc) {
50     // Se o minutoAnterior for diferente do atual, mudou de minuto, assim o corpo
51     // do if só corre uma vez em cada minuto
52
53     if (minutoAnterior != rtc->getSecond()) {
54         bool a;
55         HORA rtcHora(rtc->getHour(a, a), rtc->getMinute());
56         EEPROM_USER_DATA users[MAX_USERS];
57         uint8_t usersCount = getAllUsersData(users);
58
59         // Percorre todos os utilizadores
60         for (uint8_t i = 0; i < usersCount; i++) {
61             // Obtem cada um a hora de aquecer naquele dia
62             // (se a hora for 25 significa que não tem alarme agendado)
63             HORA hora = users[i].horaPorDiaSemana[rtc->getDoW()];
64             bool mudouDeDia;
65             if (hora.hora < 25) {
66                 HORA horaAIricular =
67                     hora.subtraiMinutos(MINUTOS_ANTES_LIGAR_AQUECEDOR, &mudouDeDia);
68

```

```

69     if (!mudouDeDia) {
70         // Se a hora coincide com a atual
71         comparaHoraELigaAquecedor(rtc, &rtcHora, &horaAIricular,
72                                     users[i].numMin, users[i].temperature);
73     }
74 }
75
76 if (rtcHora.hora == 23 &&
77     rtcHora.minuto >= 60 - MINUTOS_ANTES_LIGAR_AQUECEDOR) {
78     // Verifica se no dia atual é suposto ligar o aquecedor para estar
79     // quente no dia seguinte
80     // (ex: se estiver programado para estar quente à 00:03 do dia seguinte
81     // pode ter de ligar às 23:58 do dia atual) Apenas é relevante a partir
82     // das 23 e 55 (com MINUTOS_ANTES_LIGAR_AQUECEDOR a 5 minutos)
83     hora = users[i].horaPorDiaSemana[(rtc->getDoW() + 1) % 7];
84     if (hora.hora < 25) {
85         HORA horaAIricular =
86             hora.subtraiMinutos(MINUTOS_ANTES_LIGAR_AQUECEDOR, &mudouDeDia);
87
88         if (mudouDeDia) {
89             // Se a hora coincide com a atual
90             comparaHoraELigaAquecedor(rtc, &rtcHora, &horaAIricular,
91                                         users[i].numMin, users[i].temperature);
92         }
93     }
94 }
95 }
96 }
97 minutoAnterior = rtc->getSecond();
98 }
99
100 void Estado::estadoNada(DS3231* rtc) {
101     isAquecedorLigado = false;
102 }
103 void Estado::estadoTemporizador(DS3231* rtc) {
104     // Se ainda não se tiver atingido a hora de paragem
105     if (!DateTime_anteres_DS3231(rtc, &horaDeParagem))
106         isAquecedorLigado = true;
107     else {
108         estadoCodigo = ESTADO_NADA;
109         estadoMudou = true;
110     }
111 }
112 void Estado::estadoAqueceMantem(float temp, DS3231* rtc) {
113     // Se ainda não se tiver atingido a hora de paragem
114     if (!DateTime_anteres_DS3231(rtc, &horaDeParagem))
115         mantemTemp(temp);
116     else {
117         estadoCodigo = ESTADO_NADA;
118         estadoMudou = true;
119     }
120 }
121 void Estado::loop(float temp, DS3231* rtc) {
122     checkHoraDeAquecer(rtc);
123
124     switch (estadoCodigo) {
125         case ESTADO_NADA:
126             estadoNada(rtc);
127             break;
128         case ESTADO_TEMPORIZADOR:
129             estadoTemporizador(rtc);
130             break;
131         case ESTADO_AQUECE_E_MANTEM:
132             estadoAqueceMantem(temp, rtc);
133             break;
134     }
135 }
136

```

```
137 void Estado::mantemTemp(float temp) {
138     // Se o aquecedor está desligado
139     if (!isAquecedorLigado) {
140         // e a temperatura atual estiver abaixo da mínima
141         if (temp <= tempAManter - PRECISAO_TEMP)
142             // Liga o aquecedor
143             isAquecedorLigado = true;
144     }
145     // Se o aquecedor está ligado e a temperatura acima da máxima
146     else if (temp > tempAManter + PRECISAO_TEMP)
147         // Desliga o aquecedor
148         isAquecedorLigado = false;
149 }
```

```
1  /*
2   *         memoria.h - Gestão da informação armazenada na EEPROM externa
3   *         Criada por José Neves e Miguel Fazenda, 26 de março de 2018
4   *         BitChallenge
5   */
6
7 #ifndef MEMORIA_H
8 #define MEMORIA_H
9
10 #include <extEEPROM.h>
11 #include "structs.h"
12
13 #define MAX_USERS 8
14 #define PAGE_SIZE 64
15 #define MAX_TEMP 40
16 #define MIN_TEMP 15
17
18 extern extEEPROM memEEPROM;
19
20 int8_t getAllUsersData(EEPROM_USER_DATA* userData);
21 EEPROM_USER_DATA getSelectedUser(uint8_t userSelecionado);
22 EEPROM_USER_DATA_OPTIONAL getUserData(String user);
23
24 short findPageForUser(String name);
25 void setData(EEPROM_USER_DATA data);
26 void removeUserData(String user);
27 void resetEEPROM();
28
29 #endif
```

```

1  /*
2   memoria.cpp - Gestão da informação armazenada na EEPROM externa
3   Criada por José Neves e Miguel Fazenda, 26 de março de 2018
4   BitChallenge 2018
5  */
6
7 #include "memoria.h"
8
9 #define PAGE_NETWORK_CONNECTION_SETTINGS 128
10 #define PAGE_USER_FIRST_CONNECTION_SETTINGS 1024
11
12 extern extEEPROM memEEPROM;
13
14 // Retorna os dados de todos os utilizadores na EEPROM
15 int8_t getAllUsersData(EEPROM_USER_DATA* userData) {
16     int8_t counter = 0;
17     EEPROM_USER_DATA data;
18
19     // Percorre todos os espaços da EEPROM que possam corresponder a utilizadores
20     for (uint8_t userNum = 0; userNum < MAX_USERS; userNum++) {
21         // Guarda em data o que lá está
22         memEEPROM.read(PAGE_USER_FIRST_CONNECTION_SETTINGS + userNum * PAGE_SIZE,
23                         (byte*)&data, sizeof(EEPROM_USER_DATA));
24
25         // Se estiver lá um utilizador
26         if (data.userName[0] != 0) {
27             // e o apontador dado apontar para algum endereço
28             if (userData != NULL)
29                 userData[counter] = data; // Guarda-o no próximo lugar do vetor
30
31             // Atualiza o número de utilizadores encontrados
32             counter++;
33         }
34     }
35
36     return counter;
37 }
38 // Retorna os dados de um utilizador a partir do seu número
39 EEPROM_USER_DATA getSelectedUser(uint8_t userSelecionado) {
40     EEPROM_USER_DATA users[MAX_USERS];
41
42     getAllUsersData(users);
43
44     return users[userSelecionado];
45 }
46 // Retorna os dados de um utilizador a partir do seu nome ou um utilizador vazio
47 EEPROM_USER_DATA_OPTIONAL getUserData(String user) {
48     EEPROM_USER_DATA data;
49
50     // Percorre todos os espaços da EEPROM que possam corresponder a utilizadores
51     for (uint8_t userPageNum = 0; userPageNum < MAX_USERS; userPageNum++) {
52         // Guarda em data o que lá está
53         memEEPROM.read(
54             PAGE_USER_FIRST_CONNECTION_SETTINGS + userPageNum * PAGE_SIZE,
55             (byte*)&data, sizeof(EEPROM_USER_DATA));
56
57         // Se os nomes corresponderem
58         if (user.equalsIgnoreCase(data.userName))
59             // Retorna os seus dados
60             return EEPROM_USER_DATA_OPTIONAL{true, userPageNum, data};
61     }
62
63     // Se não tiver encontrado um utilizador com esse nome, retorna um vazio
64     return EEPROM_USER_DATA_OPTIONAL{
65         false, -1,
66         EEPROM_USER_DATA{
67             "", {HORA(), HORA(), HORA(), HORA(), HORA(), HORA(), HORA()}, 0, 0}};
68 }

```

```

69
70 // Devolve um sitio na EEPROM para escrever um utilizador, o sítio onde ele
71 // está se já estiver na EEPROM ou -1 se não existir espaço
72 short findPageForUser(String user) {
73     EEPROM_USER_DATA_OPTIONAL foundUser = getUserData(user);
74
75     // Se o utilizador existir
76     if (foundUser.userExists)
77         // Retorna a sua página
78         return foundUser.userPageNum;
79     else {
80         EEPROM_USER_DATA data;
81
82         // Percorre todos os espaços da EEPROM que possam corresponder a
83         // utilizadores
84         for (uint8_t userPageNum = 0; userPageNum < MAX_USERS; userPageNum++) {
85             // Guarda em data o que lá está
86             memEEPROM.read(
87                 PAGE_USER_FIRST_CONNECTION_SETTINGS + userPageNum * PAGE_SIZE,
88                 (byte*)&data, sizeof(EEPROM_USER_DATA));
89
90             // Se não estiver lá um utilizador
91             if (strlen(data.userName) == 0)
92                 // Retorna a página
93                 return userPageNum;
94         }
95     }
96
97     // Se estiver cheio, retorna -1
98     return -1;
99 }
100
101 // Escreve os dados de um utilizador já existente ou de um novo
102 void setData(EEPROM_USER_DATA data) {
103     uint8_t pageForUser = findPageForUser(data.userName);
104
105     // Se se tiver encontrado uma página
106     if (pageForUser != -1)
107         // Escreve lá os dados do utilizador
108         memEEPROM.write(
109             PAGE_USER_FIRST_CONNECTION_SETTINGS + pageForUser * PAGE_SIZE,
110             (byte*)&data, sizeof(EEPROM_USER_DATA));
111 }
112
113 void removeUserData(String user) {
114     EEPROM_USER_DATA_OPTIONAL foundUser = getUserData(user);
115
116     // Se se tiver encontrado o utilizador pretendido
117     if (foundUser.userPageNum != -1) {
118         // Declara dados de utilizador vazios
119         EEPROM_USER_DATA userEmptyData = EEPROM_USER_DATA{
120             "", {HORA(), HORA(), HORA(), HORA(), HORA(), HORA(), HORA()}, 0, 0};
121         // Escreve-os no espaço do utilizador
122         memEEPROM.write(
123             PAGE_USER_FIRST_CONNECTION_SETTINGS + foundUser.userPageNum * PAGE_SIZE,
124             (byte*)&userEmptyData, sizeof(EEPROM_USER_DATA));
125     }
126 }
127
128 // Remove todos os utilizadores
129 void resetEEPROM() {
130     // Declara dados de utilizador vazios
131     EEPROM_USER_DATA userEmptyData = EEPROM_USER_DATA{
132         "", {HORA(), HORA(), HORA(), HORA(), HORA(), HORA(), HORA()}, 0, 0};
133
134     // Percorre todos os espaços da EEPROM que possam corresponder a
135     // utilizadores
136     for (uint8_t userPageNum = 0; userPageNum < MAX_USERS; userPageNum++) {

```

```
137     // Escreve os dados vazios no espaço em que está
138     memEEPROM.write(
139         PAGE_USER_FIRST_CONNECTION_SETTINGS + userPageNum * PAGE_SIZE,
140         (byte*)&userEmptyData, sizeof(EEPROM_USER_DATA));
141 }
142 }
```

```
1  /*
2   *      opcoes.h - Gestão da navegação num menu de várias opções
3   *      Criada por José Neves e Miguel Fazenda, 5 de fevereiro de 2018
4   *      BitChallenge 2018
5  */
6
7 #ifndef OPCOES_H
8 #define OPCOES_H
9
10 #include "input.h"
11 #include "printLCD.h"
12
13 class Opcoes {
14 public:
15     Opcoes();
16
17     uint8_t opcaoAtual = 0;
18     uint8_t opcaoMaisAcima = 0;
19     uint8_t numOpcoes = 0;
20
21     uint16_t needsFreeBoolArray;
22     char** listaOpcoes = NULL;
23
24     void printOpcoes();
25     void slideThroughOpcoes();
26     void setOpcoes(int numeroOpcoes);
27
28     void setConstOption(uint8_t num, const char* str);
29     void setDynOptionCpy(uint8_t num, char* src, uint8_t len);
30 };
31
32 #endif
```

```

1  /*
2   * opcoes.cpp - Gestão da navegação num menu de várias opções
3   * Criada por José Neves e Miguel Fazenda, 5 de fevereiro de 2018
4   * BitChallenge 2018
5  */
6
7 #include "opcoes.h"
8
9 extern Input input;
10
11 Opcoes::Opcoes() {
12     uint8_t setaHorizontalMap[] = SETA_HORIZONTAL_MAP;
13     lcd.createChar(SETA_HORIZONTAL, setaHorizontalMap);
14     // Criam uma seta horizontal
15
16     listaOpcoes = NULL;
17 }
18 void Opcoes::printOpcoes() {
19     // Percorre os índices das opções desde aquela que está mais acima até à que
20     // fica mais abaixo ou à última
21     for (uint8_t i = opcaoMaisAcima; i < opcaoMaisAcima + 4 && i < numOpcoes;
22          i++) {
23         lcd.setCursor(1, i - opcaoMaisAcima); // Vai descendo o cursor pelo ecrã
24                                     // sempre na coluna 1
25         lcd.print(listaOpcoes[i]);           // Imprime a opção
26     }
27
28     lcd.setCursor(0, opcaoAtual - opcaoMaisAcima);
29     lcd.write(SETA_HORIZONTAL);
30 }
31 void Opcoes::slideThroughOpcoes() {
32     if (input.checkCliqueBotao(BOTA_O_CIMA) ||
33         input.rotationEvent == ROTATION_LEFT) {
34         if (opcaoAtual > 0) {
35             lcd.setCursor(0, opcaoAtual - opcaoMaisAcima);
36             lcd.print(" ");
37             opcaoAtual--;
38
39             /*Se a opção selecionada for a primeira que se vê no lcd, diminuir o
40              opcaoView para passar a ser a segunda, caso não seja a primeira de todas
41              as opções*/
42             if (opcaoAtual == opcaoMaisAcima && opcaoAtual > 0) {
43                 opcaoMaisAcima--;
44                 lcd.clear();
45                 printOpcoes();
46             } else {
47                 lcd.setCursor(0, opcaoAtual - opcaoMaisAcima);
48                 lcd.write(SETA_HORIZONTAL);
49             }
50         }
51     } else if (input.checkCliqueBotao(BOTA_O_BAIXO) ||
52                input.rotationEvent == ROTATION_RIGHT) {
53         if (opcaoAtual < numOpcoes - 1) {
54             lcd.setCursor(0, opcaoAtual - opcaoMaisAcima);
55             lcd.print(" ");
56             opcaoAtual++;
57
58             /*Se a opção selecionada for a ultima que se vê no lcd, aumentar o
59              opcaoView para passar a ser a terceira, caso não seja a ultima de todas as
60              opções*/
61             if (opcaoAtual - opcaoMaisAcima == 3 && opcaoAtual < numOpcoes - 1) {
62                 opcaoMaisAcima++;
63                 lcd.clear();
64                 printOpcoes();
65             } else {
66                 lcd.setCursor(0, opcaoAtual - opcaoMaisAcima);
67                 lcd.write(SETA_HORIZONTAL);
68             }
69     }
70 }

```

```

69      }
70  }
71 }
72 void Opcoes::setOpcoes(int numeroOpcoes) {
73     // Se listaOpcoes guardar algo
74     if (listaOpcoes != NULL) {
75         // Cria uma máscara de bits com o bit mais à direita a 1, que serve para
76         // selecionar o bit
77         // correspondente à opção i do needsFreeBoolArray
78         uint16_t mask = 1;
79
80         // Para cada opção na lista
81         for (int i = 0; i < numOpcoes; i++) {
82             // Se ela precisar de ser libertada
83             if (needsFreeBoolArray & mask) {
84                 delete[] listaOpcoes[i]; // Liberta-a
85                 listaOpcoes[i] = NULL; // e assegura-se que fica a NULL
86             }
87
88             // Desloca a máscara para a esquerda
89             mask = mask << 1;
90         }
91
92         // Liberta o vetor
93         delete[] listaOpcoes;
94     }
95
96     numOpcoes = numeroOpcoes;
97     opcaoAtual = 0;
98     opcaoMaisAcima = 0;
99
100    listaOpcoes = new char*[numeroOpcoes];
101 }
102
103 void Opcoes::setConstOption(uint8_t num, const char* str) {
104     // Coloca o bit correspondente a esta opção a 0 (não necessita free, porque
105     // não foi alocada memória)
106     needsFreeBoolArray &= ~(1 << num);
107
108     listaOpcoes[num] = (char*)str;
109 }
110
111 // Cria uma opção com string alocada dinamicamente
112 // Aloca memória com len+1 bytes para a string e copia a string src
113 void Opcoes::setDynOptionCpy(uint8_t num, char* src, uint8_t len) {
114     // Coloca o bit correspondente a esta opção a 1 (necessita free, porque foi
115     // alocada memória)
116     needsFreeBoolArray |= (1 << num);
117
118     // Aloca memória e copia a string
119     char* ptr = new char[(len + 1) * sizeof(char)];
120     strcpy(ptr, src);
121     listaOpcoes[num] = ptr;
122 }
```

```

1  /*
2  	input.h - Gestão dos inputs dados pelo utilizador
3  	Criada por José Neves e Miguel Fazenda, 26 de março de 2018
4  	BitChallenge 2018
5  */
6
7 #ifndef INPUT_H
8 #define INPUT_H
9
10 #include <LiquidCrystal_I2C.h>
11 #include <stdint.h>
12
13 #define NUM_BOTOES 5
14
15 #define BOTAO_CIMA 1
16 #define BOTAO_BAIXO 2
17 #define BOTAO_OK 3
18 #define BOTAO_CANCELAR 4
19 #define BOTAO_ROTARY 5
20
21 #define BOTAO0_PIN 12
22 #define BOTAO1_PIN 11
23 #define BOTAO2_PIN 10
24 #define BOTAO3_PIN 9
25 #define BOTAO_ROTARY_PIN 4
26
27 #define ROTATION_NONE 0
28 #define ROTATION_RIGHT 1
29 #define ROTATION_LEFT -1
30
31 #define RE_A_PIN 2
32 #define RE_B_PIN 3
33 #define PRECISAO 4
34
35 typedef struct botao_stru {
36     uint8_t estadoBotaoAnt, estadoBotao;
37 } Botao;
38
39 class Input {
40 public:
41     int8_t rotationCounter = 0;
42     int8_t rotationEvent = ROTATION_NONE;
43     Botao botoes[NUM_BOTOES];
44
45     // Guarda o tempo em que um botão foi premido (ou evento de rotacao) pela
46     // ultima vez
47     // para depois desligar a backlight passado X tempo
48     unsigned long lastInputChangeMillis;
49
50     void init();
51     void update(LiquidCrystal_I2C* lcd);
52     void refresh();
53
54     void updateBacklightTimeout(LiquidCrystal_I2C* lcd);
55
56     bool checkCliqueBotao(uint8_t botao);
57
58     void rotaryEncoderInterrupt();
59     bool rotaryAlteraVar(uint8_t* var, uint8_t varMax, uint8_t varMin);
60 };
61
62 #endif

```

```

1  /*
2  	input.cpp - Gestão dos inputs dados pelo utilizador
3  	Criada por José Neves e Miguel Fazenda, 26 de março de 2018
4  	BitChallenge 2018
5  */
6
7 #include "input.h"
8 #include <Arduino.h>
9
10 void Input::init() {
11     // Inicialização dos botões
12     pinMode(BOTA00_PIN, INPUT_PULLUP);
13     pinMode(BOTA01_PIN, INPUT_PULLUP);
14     pinMode(BOTA02_PIN, INPUT_PULLUP);
15     pinMode(BOTA03_PIN, INPUT_PULLUP);
16     pinMode(BOTAOROTARY_PIN, INPUT);
17     // Inicialização do rotary encoder
18     pinMode(2, INPUT);
19     pinMode(3, INPUT);
20
21     for (unsigned int i = 0; i < NUM_BOTOES; i++) {
22         botoes[i].estadoBotaoAnt = 0;
23         botoes[i].estadoBotao = 0;
24     }
25
26     lastInputChangeMillis = millis();
27 }
28
29 // Atualiza o estado dos botões
30 void Input::update(LiquidCrystal_I2C* lcd) {
31     // Caso se tenha rodado o rotary encoder
32     if (rotationEvent != ROTATION_NONE)
33         updateBacklightTimeout(lcd);
34
35     // Guarda o estado anterior dos botões
36     this->refresh();
37
38     // Lê o estado de cada botão
39     botoes[0].estadoBotao = !digitalRead(BOTA00_PIN);
40     botoes[1].estadoBotao = !digitalRead(BOTA01_PIN);
41     botoes[2].estadoBotao = !digitalRead(BOTA02_PIN);
42     botoes[3].estadoBotao = !digitalRead(BOTA03_PIN);
43     botoes[4].estadoBotao = !digitalRead(BOTAOROTARY_PIN);
44
45     // Para cada botão
46     for (unsigned int i = 0; i < NUM_BOTOES; i++)
47         // Se tiver sido premido ou largado
48         if (botoes[i].estadoBotaoAnt != botoes[i].estadoBotao)
49             updateBacklightTimeout(lcd);
50 }
51
52 void Input::updateBacklightTimeout(LiquidCrystal_I2C* lcd) {
53     // Guarda o tempo em milissegundos no momento
54     lastInputChangeMillis = millis();
55 }
56
57 bool Input::checkCliqueBotao(uint8_t botao) {
58     // Se o botão tiver sido clicado e no momento já se encontrar largado
59     if (botoes[botao - 1].estadoBotao == true &&
60         botoes[botao - 1].estadoBotaoAnt == false) {
61         botoes[botao - 1].estadoBotaoAnt = 1;
62         // Atualiza o estado anterior do botão
63         return true;
64     }
65
66     return false;
67 }
68

```

```
69 // Guarda os estados dos botoes no momento anterior, para depois se poder
70 // comparar com o
71 // estado atual
72 void Input::refresh() {
73     for (unsigned int i = 0; i < NUM_BOTOES; i++) {
74         botoes[i].estadoBotaoAnt = botoes[i].estadoBotao;
75     }
76
77     rotationEvent = ROTATION_NONE;
78 }
79
80 void Input::rotaryEncoderInterrupt() {
81     bool p2 = digitalRead(2);
82
83     // Conta o numero de rotações. Serve para diminuir a sensibilidade do rotary
84     // encoder
85     rotationCounter += (digitalRead(3) != p2) ? 1 : -1;
86
87     if (rotationCounter >= 4) {
88         rotationEvent = ROTATION_RIGHT;
89         rotationCounter = 0;
90     } else if (rotationCounter <= -4) {
91         rotationEvent = ROTATION_LEFT;
92         rotationCounter = 0;
93     }
94 }
95
96 bool Input::rotaryAlteraVar(uint8_t* var, uint8_t varMax, uint8_t varMin) {
97     if (rotationEvent == ROTATION_RIGHT) {
98         // Aumenta-se a variável em 1, a não ser que ela seja igual ao máximo
99         // (nesse caso, passa a ser o mínimo)
100        *var = (*var < varMax) ? *var + 1 : varMin;
101        return true;
102    } else if (rotationEvent == ROTATION_LEFT) {
103        // Diminui-se a hora em 1, a não ser que ela seja igual ao mínimo
104        // (nesse caso, passa a ser o máximo)
105        *var = (*var > varMin) ? *var - 1 : varMax;
106        return true;
107    }
108
109    return false;
110 }
```

```

1  /*
2   * printLCD.h - Apresentação de dados frequentes
3   * Criada por José Neves e Miguel Fazenda, 7 de fevereiro de 2018
4   * BitChallenge 2018
5  */
6
7 #ifndef PRINT_H
8 #define PRINT_H
9
10 #include <EEPROM.h>
11 #include <LiquidCrystal_I2C.h>
12 #include <stdint.h>
13 #include "relogio.h"
14
15 #define LCD_COMP 20
16 #define LCD_ALT 4
17
18 #define DOW 0
19 #define MES 1
20
21 #define CAMPO_DOW 0
22 #define CAMPO_DIA 1
23 #define CAMPO_MES 2
24 #define CAMPO_ANO 3
25 #define CAMPO_HORA 4
26 #define CAMPO_MINUTO 5
27 #define NAO_PISCAR -1
28
29 #define SETA_HORIZONTAL_MAP \
30     { 0x00, 0x04, 0x02, 0x1F, 0x02, 0x04, 0x00, 0x00 }
31 #define SETA_VERTICAL_MAP \
32     { 0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x00, 0x00 }
33
34 #define SETA_HORIZONTAL 1
35 #define SETA_VERTICAL 2
36
37 extern LiquidCrystal_I2C lcd;
38
39 String multiplicaCaracter(uint8_t numVezes, char ch);
40 String Int_To_Str(uint16_t numero, uint8_t expectedNumDigits);
41 void print_Int(uint8_t numero, uint8_t expectedNumDigits);
42 void print_Blinking_Str(String str, bool vazio);
43
44 void printTemperature(uint8_t temp);
45 void printHumidity(uint8_t hum);
46
47 String getMes_ou_Dow(uint8_t valor, byte grandeza);
48 void printData(uint8_t ano, uint8_t mes, uint8_t dia, int8_t modo, bool aceso);
49 void printHora(uint8_t hora, uint8_t minuto, int8_t modo, bool aceso);
50 void printDow(uint8_t dow, int8_t modo, bool aceso);
51
52 void printUserName(char strIns[]);
53
54 #endif
55

```

```

1  /*
2   * printLCD.cpp - Apresentação de dados frequentes
3   * Criada por José Neves e Miguel Fazenda, 7 de fevereiro de 2018
4   * BitChallenge 2018
5  */
6
7 #include "printLCD.h"
8
9 extern LiquidCrystal_I2C lcd;
10
11 String multiplicaCaracter(uint8_t numVezes, char ch) {
12     String res = "";
13
14     for (uint8_t i = 0; i < numVezes; i++)
15         res += ch;
16
17     return res;
18 }
19 String Int_To_Str(uint16_t numero, uint8_t expectedNumDigits) {
20     String numeroStr = String(numero);
21     uint8_t numZeros = expectedNumDigits - numeroStr.length();
22
23     return multiplicaCaracter(numZeros, '0') + numeroStr;
24 }
25 void print_Int(uint8_t numero, uint8_t expectedNumDigits) {
26     lcd.print(Int_To_Str(numero, expectedNumDigits));
27 }
28 void print_Blinking_Str(String str, bool vazio) {
29     // Se se pretender apagar
30     if (vazio)
31         lcd.print(multiplicaCaracter(str.length(), ' '));
32     // Se se pretender escrever
33     else
34         lcd.print(str);
35 }
36
37 void printTemperature(uint8_t temp) {
38     print_Int(temp, 2);
39     lcd.print((char)223);
40     lcd.print("C");
41 }
42 void printHumidity(uint8_t hum) {
43     print_Int(hum, 2);
44     lcd.print("%");
45 }
46
47 // Grandeza pode tomar os valores 0 ou 1, sendo que 0 corresponde a dow e 1 a
48 // mês
49 String getMes_ou_Dow(uint8_t valor, byte grandeza) {
50     char str[4];
51
52     EEPROM.get(valor * 3 + grandeza * 7 * 3, str);
53     str[3] = '\0';
54
55     return String(str);
56 }
57 void printDow(uint8_t dow, int8_t modo, bool aceso) {
58     print_Blinking_Str(getMes_ou_Dow(dow, DOW), modo == CAMPO_DOW && !aceso);
59 }
60
61 void printData(uint8_t ano, uint8_t mes, uint8_t dia, int8_t modo, bool aceso) {
62     print_Blinking_Str(Int_To_Str(dia, 2), modo == CAMPO_DIA && !aceso);
63     lcd.print(" ");
64     print_Blinking_Str(getMes_ou_Dow(mes - 1, MES), modo == CAMPO_MES && !aceso);
65     lcd.print(" ");
66     print_Blinking_Str(Int_To_Str(ano + 2000, 4), modo == CAMPO_ANO && !aceso);
67 }
68 void printHora(uint8_t hora, uint8_t minuto, int8_t modo, bool aceso) {

```

```
69     print_Blinking_Str(Int_To_Str(hora, 2), modo == CAMPO_HORA && !aceso);
70     lcd.print(":");
71     print_Blinking_Str(Int_To_Str(minuto, 2), modo == CAMPO_MINUTO && !aceso);
72 }
73
74 void printUserName(char strIns[]) {
75     uint8_t posFinal = strlen(strIns);
76
77     lcd.print(strIns);
78     lcd.print(multiplicaCaracter(LCD_COMP - 2 - posFinal, ' '));
79 }
```

```
1  /*
2   relogio.h - Gestão da informação relacionada com o RTC
3   Criada por José Neves e Miguel Fazenda, 18 de março de 2018
4   BitChallenge 2018
5 */
6
7 #ifndef RELOGIO_H
8 #define RELOGIO_H
9
10 #include <stdint.h>
11
12 #include <DS3231.h>
13 #include "structs.h"
14
15 bool verificaDataHora(DATE* data, HORA* hora);
16
17 bool DateTime_anteres_DS3231(DS3231* rtc, DateTime* dataHora);
18
19 DateTime converte_DS3231(DS3231* rtc);
20
21 #endif
```

```

1  /*
2   relogio.cpp - Gestão da informação relacionada com o RTC
3   Criada por José Neves e Miguel Fazenda, 18 de março de 2018
4   BitChallenge 2018
5 */
6
7 #include "relogio.h"
8
9 bool verificaDataHora(DATE* data, HORA* hora) {
10    bool bissexto = data->ano % 4 == 0;
11
12    if (data->mes == 2) {
13        if (data->dia > 28 && !bissexto)
14            return false;
15        if (data->dia > 29 && bissexto)
16            return false;
17    }
18
19    if (data->mes == 4 || data->mes == 6 || data->mes == 9 || data->mes == 11)
20        if (data->dia > 30)
21            return false;
22
23    return true;
24}
25
26 bool DateTime_anteres_DS3231(DS3231* rtc, DateTime* dataHora) {
27    bool a = false;
28
29    if (dataHora->year() - 2000 < rtc->getYear())
30        return true;
31    else if (dataHora->year() - 2000 == rtc->getYear()) {
32        if (dataHora->month() < rtc->getMonth(a))
33            return true;
34        else if (dataHora->month() == rtc->getMonth(a)) {
35            if (dataHora->day() < rtc->getDate())
36                return true;
37            else if (dataHora->day() == rtc->getDate()) {
38                if (dataHora->hour() < rtc->getHour(a, a))
39                    return true;
40                else if (dataHora->hour() == rtc->getHour(a, a)) {
41                    if (dataHora->minute() < rtc->getMinute())
42                        return true;
43                    else if (dataHora->minute() == rtc->getMinute()) {
44                        if (dataHora->second() < rtc->getSecond())
45                            return true;
46                    }
47                }
48            }
49        }
50    }
51
52    return false;
53}
54
55 DateTime converte_DS3231(DS3231* rtc) {
56    bool a;
57
58    return DateTime(rtc->getYear() + 2000, rtc->getMonth(a), rtc->getDate(),
59                   rtc->getHour(a, a), rtc->getMinute(), rtc->getSecond());
60}
61

```

```
1  /*
2   strcucts.h - Registo do conjunto de estruturas
3   Criada por José Neves e Miguel Fazenda, 18 de março de 2018
4   BitChallenge 2018
5 */
6
7 #ifndef STRUCTS_H
8 #define STRUCTS_H
9
10 #include <stdint.h>
11
12 #define DIM_USERNAME 15
13
14 typedef struct {
15     uint8_t ano, mes, dia; // ano = anoReal - 2000 para ser de 8 bits
16 } DATE;
17
18 class HORA {
19 public:
20     HORA(void);
21     HORA(uint8_t hora, uint8_t minuto);
22     uint8_t hora, minuto;
23     HORA subtraiMinutos(uint8_t minutes, bool* mudouDeDia);
24 };
25
26 typedef struct {
27     char userName[DIM_USERNAME + 1];
28     HORA horaPorDiaSemana[7]; // 25 para não aquecer nesse dia da semana
29     uint8_t temperature;
30     uint8_t numMin;
31 } EEPROM_USER_DATA;
32
33 typedef struct horario_stru_optional {
34     bool userExists;
35     int userPageNum;
36     EEPROM_USER_DATA data;
37 } EEPROM_USER_DATA_OPTIONAL;
38
39 #endif
40
```

```
1  /*
2   strcucts.cpp - Registo do conjunto de estruturas
3   Criada por José Neves e Miguel Fazenda, 6 de setembro de 2018
4   BitChallenge 2018
5 */
6
7 #include "structs.h"
8 #include <Arduino.h>
9
10 HORA::HORA(void) {
11     this->hora = 0;
12     this->minuto = 0;
13 }
14
15 HORA::HORA(uint8_t hora, uint8_t minuto) {
16     this->hora = hora;
17     this->minuto = minuto;
18 }
19
20 // Apenas funciona para subtrair entre 0 e 59 minutos
21 HORA HORA::subtraiMinutos(uint8_t minutos, bool* mudouDeDia) {
22     int8_t hora = this->hora - minutos / 60;
23     int8_t minuto = this->minuto - minutos % 60;
24
25     if (minuto < 0) {
26         hora--;
27         minuto = 60 + minuto;
28     }
29
30     if (mudouDeDia != NULL) {
31         // Se der uma hora negativa significa que passou para o dia anterior
32         *mudouDeDia = hora < 0;
33     }
34
35     if (hora < 0) {
36         hora = 24 + hora;
37     }
38     return HORA(hora, minuto);
39 }
```

Bibliotecas Incluídas

- Arduino External EEPROM Library
 - <https://github.com/JChristensen/extEEPROM>
- New LiquidCrystal
 - <https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home>
- DHTlib
 - <https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib>
- DS3231
 - <https://github.com/NorthernWidget/DS3231>

Custos

Artigos (por ordem alfabética)	Custo Conjunto	Custo Unitário	Unidades Utilizadas	Custo final
Botões 4x1	0,89	0,30	1	0,30
Cabo FVV 3x1 (m)	0,713	0,24	1	0,24
Cabo USB	1,50	1,50	1	1,50
Caixa	4,55	4,55	1	4,55
DHT	2,64	2,64	1	2,64
EEPROM 24LC256	2,15	0,43	1	0,43
Ficha Schuk Macho	0,90	0,90	1	0,90
Fios Macho-Fêmea	0,82	0,82	1	0,82
LCD 20x4	4,03	4,03	1	4,03
Ligador Rápido 3x4	1,09	0,36	2	0,72
NEECduino	20,00	20,00	1	20,00
Papel Autocolante	1,40	1,40	1	1,40
Parafusos M3	0,96	0,16	6	0,96
PCB	1,64	0,55	1	0,55
Pilha 3.3V	3,38	1,13	1	1,13
Pinos Macho	1,33	0,07	1	0,07
Porcas	0,48	0,08	6	0,48
Relé	1,10	1,10	1	1,10
Rotary encoder	1,18	0,59	1	0,59
RTC DS3231	1,07	1,07	1	1,07
Suporte IC 8 pinos	0,30	0,15	1	0,15
Tomada	2,69	2,69	1	2,69
Transformador 5V	3,00	3,00	1	3,00
			Total	49,32

Comprovativos de compra



Good-Module

2Pcs 1x4 Matrix Array 4 Key Membrane Switch Keypad Keyboard 1x4 Keys for Arduino

Condition: New	Quantity: 1	5 available <small>26 sold / See feedback</small>
Price: C \$1.35 Approximately EUR 0.89	Buy It Now	Add to cart
Add to watch list		
100% buyer satisfaction	26 Sold	More than 83% Sold

Top-rated seller
good-module (61488) 
99.1% Positive feedback

-  Consistently receives highest buyers' ratings
-  Ships items quickly
-  Has earned a track record of excellent service

[Save this Seller](#)
[Contact seller](#)
[Visit store](#)

Shipping: FREE Economy Int'l Shipping | [See details](#)
International items may be subject to customs processing and additional charges. 
Item location: 深圳, 广东省, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Fri. Sep. 21 and Fri. Oct. 26
Seller ships within 1 day after receiving cleared payment. 

DAED-Distribuidora Acessorios Elect. Lda
Rua Jose Falcão 66-A
Lisboa
1000-185 LISBOA
Contribuinte nº 501272909
Capital Social. 5.000,00
C.R.C.Lisboa Matricula nº 1961
Tel.213535747 Tlm.934918990

Fatura-recibo nº FR 2018A12/4008
Data: 2018-06-27 Via: ORIGINAL

Consumidor Final

N.Contrib: 999999990

Referência	Design.	Pr.Unit.	Qt.	Tx.	Total
80-101671	CABO FVV 3x1 BR		0,713	1,0	23,00% 0,71
19-221413	LIGADOR RAPIDO 3x4 FLEX WAG		0,362	3,0	23,00% 1,09

Apenas utilizámos dois ligadores

Software PHC - Dmg-Processado por programa certificado nº 0006/AT (20180513)

Taxa	Base Incid.	Valor IVA
23,00%	1,46	0,34
Base Incid.	1,46	
Desconto.		
Total Iva	0,34	
Total	1,80	

Os artigos/serviços constantes deste documento foram colocados à disposição do adquirente na data 27.06.2018 alínea f) do n.º 4 do art.36 do CIVA

27.06.2018 16:30:29

**** Minimercado ****
Samjhana L. Pathak
Rua Ponta Delgada, 64 A/B
Tel.920478845
N. Contrib. 287758710
Registo na Cons. n.
Capital Social

NIF: Consumidor final

Fatura simplificada FS A18/6291
2018-06-27 16:19

Qt	Descrição	P.Unit	Valor	IVA
1.00	Diversos Acessorios	€ 1.50	€ 1.50	23
Total			€ 1.50	
<i>Cabo USB no sistema informático</i>				

Pagamento

Numerário	Valor
	€ 1.50

Taxa **Base** **Valor** **Total**

23	€ 1.22	€ 0.28	€ 1.50
----	--------	--------	--------

IVA Incluido

mJTM-Processado por programa certificado nº 2326/AT

DAED-Distribuidora Acessorios Elect. Lda
Rua Jose Falcão 66-A
Lisboa
1000-185 LISBOA
Contribuinte nº 501272909
Capital Social. 5.000,00
C.R.C.Lisboa Matrícula nº 1961
Tel.213535747 Tlm.934918990

Fatura-recibo nº FR 2018A12/5021
Data: 2018-09-11 Via: ORIGINAL

Consumidor Final

N.Contrib: 999999990

Referência	Design.	Pr.Unit.	Qt.	Tax.	Total
23-150102	CX DERIVAÇÃO EXT 160x120 C53	4,551	1,0	23,00%	4,55

Software PHC - tzkM-Processado por
programa certificado nº 0006/AT (2018051
3)

Taxa	Base Incid.	Valor IVA
23,00%	3,70	0,85

Base Incid. 3,70

Desconto.

Total Iva . 0,85

Total 4,55

Os artigos/serviços constantes deste documento foram colocados à disposição do adquirente na data 11.09.2018 alínea f) do n.º 4 do art.36 do CIVA

11.09.2018 17:08:48

AM2302 DHT22 Digital Temperature And Humidity Sensor Module Replace SHT15 SHT11

★★★★★ 1 product rating

Condition: New	More than 10 available	
Quantity: 1	222 sold / See feedback	
Price: US \$3.05	Buy it Now	
Approximately EUR 2.64	Add to cart	
Add to watch list		
100% buyer satisfaction	222 Sold	More than 67% Sold
Shipping: FREE Economy International Shipping see details		
International items may be subject to customs processing and additional charges. ⓘ		
Item location: Shen Zhen, China		
Ships to: Worldwide See exclusions		
Delivery: Estimated between Fri, Sep. 21 and Mon. Nov. 12		
Seller ships within 1 day after receiving cleared payment. ⓘ		

Top-rated seller
elecbuy (16875 ⭐)
99.4% Positive feedback

- Consistently receives highest buyers' ratings
- Ships items quickly
- Has earned a track record of excellent service

[Save this Seller](#)
[Contact seller](#)
[Visit store](#)



5PCS IC MICROCHIP DIP-8 24LC256-I/P 24LC256 NEW

Condition: New More than 10 available / 15 sold

Quantity: 1 Price: US \$2.48
Approximately EUR 2.15

Buy It Now Add to cart Add to watch list

Free Shipping 30-day Returns Longtime Member

Shipping: FREE Economy International Shipping | [See details](#)
International items may be subject to customs processing and additional charges. [Learn more](#)
Item location: Yancheng, Jiangsu, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Fri, Sep. 21 and Mon, Nov. 12
Seller ships within 1 day after receiving cleared payment. [Learn more](#)

Payments:

Returns: 30 day returns. Buyer pays for return shipping | [See details](#)

Guarantee: [Learn details](#)

DAED-Distribuidora Acessorios Elect. Lda
Rue Jose Falcão 66-A
Lisboa
1000-185 LISBOA
Contribuinte nº 501272909
Capital Social. 5.000,00
C.R.C.Lisboa Matricula nº 1961
Tel.213535747 Tlm.934918990

Fatura-recibo nº FR 2018A12/3990
Data: 2018-06-27 Vias: ORIGINAL

Consumidor Final

N.Contrib: 234479921

Referência	Design.	Pr.Unit.	Qty.	Tx.	Total
------------	---------	----------	------	-----	-------

70-782420	TOM SCHUK AP BR FORIX LEG	2,690	1,0	23,00%	2,69
35-000152	FICHA SCHUK MACHO DIR BR 152	0,904	1,0	23,00%	0,90
80-101677	CABO FVV 3x1.5 BR	0,085	1,0	23,00%	0,89

Não usado

Software PHC - ZgNq-Processado por
programa certificado nº 0006/AT (2018051
3)

Taxa	Base Incid.	Valor IVA
23,00%	3,64	0,84

Base Incid.	3,64
Desconto.	
Total Iva	0,84
Total	4,48

Os artigos/serviços constantes deste documento foram colocados à disposição do adquirente na data 27.06.2018
alínea f) do n.º 4 do art.36 do CIVA

27.06.2018 09:39:39

40PCS Dupont 10CM Male To Female Jumper Wire Ribbon Cable for Arduino

13 viewed per day

Condition: New
Quantity: 1 More than 10 available
1,671 sold / See feedback

Price: AU \$1.32
Approximately EUR 0.82

Buy It Now **Add to cart** **Add to watch list**

100% buyer satisfaction	1,671 Sold	More than 98% Sold
-------------------------	------------	--------------------

Shipping: FREE International Economy : tracked-no signature (11 to 35 business days) | [See details](#)
International items may be subject to customs processing and additional charges. ⓘ
Item location: 深圳, 广东省, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Fri. Sep. 21 and Fri. Oct. 26
Seller ships within 1 day after receiving cleared payment. ⓘ

Payments:

Yellow Green Serial IIC/I2C/TWI 2004 LCD 20X4 Character LCD Module For Arduino

Condition: New
Quantity: 1 More than 10 available
236 sold / See feedback

Price: US \$4.66
Approximately EUR 4.03

Buy It Now **Add to cart** **Add to watch list**

100% buyer satisfaction	236 Sold	Free Shipping
-------------------------	----------	---------------

Shipping: FREE Economy Shipping from China/Hong Kong/Taiwan to worldwide | [See details](#)
International items may be subject to customs processing and additional charges. ⓘ
Item location: 深圳市, 广东省, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Mon. Sep. 24 and Wed. Nov. 14
Seller ships within 3 days after receiving cleared payment. ⓘ

Payments:

Returns: 30 day returns. Buyer pays for return shipping | [See details](#)



PAPELARIA REGALO
Celestino Marques dos Santos, Unip., Lda
R. Cavaleiro Oliveira, 25-1170-086 Lisboa
NIF: 500060070
Tel. 218 149 570

NIF: Consumidor final

Fatura Simplificada
Natureza: Fatura Simplificada
2018-09-11 17:23 GERAL
FS 001/303690 Original Ter: 1

Quant.	Pr.	Unit.	IVA	Desc.	Total
D- Jornais Revistas Diversos					
1	1,40	6%			1,40
Marcador EPENE Cd M	1	1,20	23%		1,20
Itens: 2					
Total				2,60 EUR	
Forma Pag.	DINHEIRO			3,00	
Troco:				0,40	

Iva Incluido a Taxa Indicada
Taxa Valor Iva Base Imp. Valor Total
6% 0,08 1,32 1,40
23% 0,22 0,98 1,20

*** Muito Obrigado p/ preferencia ***
PAPELARIA REGALO
Powered by KIOS KUBE

Ylsj-Processado por programa certificado
nº 2/AT - Sage
Este documento não constitui documento de transporte, nos termos do Decreto-Lei nº 147/2003

Carvalho, Batista, Lda
SEDE: Rua D. Estefânia, 191-B
LOJA E SHOWROOM: Loja: Rua D. Estefânia, 98
1000-154 LISBOA
Tel: 213 157 838 | Fax: 213 153 786
NIF: 500537380
Email: carvalho.baptista.lda@gmail.com
Site: www.carvalhobatista.com

Fatura/RC POS - Num. 6385 - 17
Fatura FT 2018A17/6385 - 2018-09-12/09:54
Op: Jorge Custódio

NOME: Consumidor Final
MORADA:
NIF: Consumidor Final
LOCAL.:
Software PHC - eWeF-Processado por
programa certificado nº 0006/AT (20160728)

QTD	P.UN. PRODUTO	IVA	TOTAL
6,00	x 0,08 Porcas Sextu M-3 Inox	23% 0,48	
6,00	x 0,16 Paraf.InoxC/C. ou Cx.M	23% 0,96	
1,00	x 2,36 FITA ISOLAD.TESA 53948	23% 2,36	
			Total líquido 3,80
			IVA total 0,87
			TOTAL EURO 4,67

TAXA	B.INC.	IVA	TAXA	B.INC.	IVA
6%	0,00	0,00	13%	0,00	0,00
23%	3,80	0,87			

Documento indispensável para troca no prazo de 10 dias.
Não efectuamos devolução de dinheiro.

Os artigos foram colocados à disposição do adquirente
em 12.09.2018

Obrigado pela visita, volte sempre!

5Pcs Double Side Printed Circuit PCB Vero Prototyping Track Strip Board 5x7cm

★★★★★ Be the first to write a review.

Condition: New

Quantity: 1 More than 10 available / 4 sold

Price: AU \$2.64
Approximately EUR 1.64

Buy It Now **Add to cart**

Best Offer: **Make Offer** Add to watch list

Free Shipping **30-day Returns** **Longtime Member**

Shipping: FREE International Economy : tracked-no signature (11 to 35 business days) | See details
International items may be subject to customs processing and additional charges. ⓘ
Item location: Hong Kong, China
Online Mandirada Com Andriod

Top-rated seller
happyness8966 (580186) ★★★★★ 98.4% Positive feedback
✓ Consistently receives highest bu
✓ Ships items quickly
✓ Has earned a track record of exc
service
.....
♥ Save this Seller
Contact seller
Visit store

5 pcs New Maxell CR2032 3V Coin Cell Button Battery (FREE SHIPPING) EXP: 2027

4 sold in last 24 hours

Condition: New
Quantity: 1 More than 10 available
429 sold / See feedback

Price: US \$3.90 Approximately EUR 3.38
Buy It Now
Add to cart

Best Offer:
Make Offer
Add to watch list

100% buyer satisfaction Limited quantity remaining More than 87% Sold

Shipping: FREE Economy Shipping from China/Hong Kong/Taiwan to worldwide | [See details](#)
International items may be subject to customs processing and additional charges. ⓘ
Item location: Hong Kong, Hong Kong
Ships to: Worldwide

Delivery: Estimated between Thu, Sep. 20 and Wed, Oct. 31

Top-rated seller
deepimpex99 (933 ⭐)
99.7% Positive feedback
Consistently receives highest buyers' ratings
Ships items quickly
Has earned a track record of excellent service

Save this Seller
Contact seller
See other items

20PCS 2.54mm 40 Pin Male Single Row Pin Header Strip GOOD QUALITY

★★★★★ 5 product ratings | Write a review

Condition: New
Quantity: 1 More than 10 available
3,453 sold / See feedback

Price: US \$1.55 Approximately EUR 1.33
Buy It Now
Add to cart

Best Offer:
Make Offer
Add to watch list

100% buyer satisfaction 3,453 Sold More than 96% Sold

Shipping: FREE Economy International Shipping | [See details](#)
International items may be subject to customs processing and additional charges. ⓘ
Item location: Shenzhen, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Tue, Sep. 25 and Wed, Nov. 14
Seller ships within 1 day after receiving cleared payment. ⓘ

Top-rated seller
surv2014 (214806 ⭐)
99.3% Positive feedback
Consistently receives highest buyers' ratings
Ships items quickly
Has earned a track record of excellent service

Save this Seller
Contact seller
Visit store

5V 1/2/4/6/8 Channel Relay Board Module Optocoupler LED F

Condition: New
select: 1 CH Without Optocoupler(Black...)

Quantity: 1 5 available
86 sold / See feedback

Price: GBP 0.99 Approximately EUR 1.10
Buy It Now
Add to cart
Add to watch list

100% buyer satisfaction 86 Sold More than 67% Sold

Shipping: FREE Economy Shipping | [See details](#)
International items may be subject to customs processing and additional charges. ⓘ
Item location: Shenzhen, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Fri, Sep. 21 and Mon, Nov. 12
Seller ships within 1 day after receiving cleared payment. ⓘ

Payments:



Module-ME

2pcs

2x New KY-040 Rotary Encoder Module for Arduino AVR PIC

Condition: New
Sale ends in: 02d 13h 36m
Quantity: 1 More than 10 available

Was: C \$1.99
You save: C \$0.20 (10% off)
Price: C \$1.79 Approximately EUR 1.18

Buy It Now **Add to cart** **Make Offer**

Free Shipping 30-day Returns Longtime Member

Top-rated seller utopianliv 98.5% Positive feedback
Consistently ships items
Has earned a track record of excellent service

Save Contact seller Visit store

Arduino DS3231 ZS042 AT24C32 IIC Module Precision RTC Real time Clock Memory

Condition: New
Quantity: 1 More than 10 available 5,420 sold / See feedback

Price: US \$1.24 Approximately EUR 1.07

Buy It Now **Add to cart** **Add to watch list**

100% buyer satisfaction 5,420 Sold More than 95% Sold

Shipping: FREE Economy International Shipping | [See details](#)
International items may be subject to customs processing and additional charges.
Item location: ShenZhen, China
Ships to: Worldwide [See exclusions](#)

Delivery: Estimated between Fri. Sep. 21 and Mon. Nov. 12
Seller ships within 1 day after receiving cleared payment.

Top-rated seller alice1101983 (458199) 99.2% Positive feedback
Consistently receives highest buyers' rating
Ships items quickly
Has earned a track record of excellent service

Save this Seller Contact seller Visit store

ARABOT HI-FI Largo Emílio Infante da Câmara, 5 - 6 2000-051 Santarém / Tel.:243324676 NIF:500959196/C.R.C.Santarém nº1233 Capital Social :5.000€ NIF: Consumidor final	LOJA DO EURO MOHAMMAD IMRAN SARKER CONTRIB.276408527 TEL.920117521 AV.ALMIRANTE REIS,134-B 1150-023 LISBOA M.FISCAL: RUA JORGE SENA,30-10 ESQ. 2675-391 ODIVELAS
Fatura Simplificada Natureza:Fatura Simplificada 2018-08-02 11:39 Loja FS EL2/4429 Original Ter:2 Quant. Pr. Unit. IVA Desc. Total Barra de 22 pinos macho/macho - TIM022 3 0,55 23% 1,65 Suporte IC 8p 2 0,15 23% 0,30 Itens: 5 Apenas usámos 1, pois era para Total 1,95 EUR Forma Pag. DINHEIRO 1,95 Iva Incluido a Taxa Indicada Taxa Valor Iva Base Imp. Valor Total 23% 0,36 1,59 1,95 jWZ-Processado por programa certificado nº 2648/AT - Sage Este documento não constitui documento de transporte, nos termos do Decreto-Lei n.º 147/2003	FATURA SIMPLIFICADA ORIGINAL PST:1 DATA 2018/06/27 - 9:46 Nº D/14936 Cliente: Consumidor final Morada: Cont.Nº: N/A Artigo Qtd Preço € Valor € ADAPTADOR 23% 1 un x 3,00 = 3,00 #001- TOTAL..... 3,00 Númerário..... 3,00 Troca..... 0,00 IVA incluido à taxa indicada Taxa % Sujeito IVA Total 23,00 2,44 0,56 3,00 BTy0-Processado por programa certificado Nº 88/AT - 1 D/14936 Opr: EMPREGADO OBRIGADO - VOLTE SEMPRE

Comprovativos de aproveitamento curricular

Aluno: José Afonso De Oliveira Moreno Neves (89683) - Bilhete de Identidade (14471622)

Informação do apuramento

A Matrícula do Aluno ainda não foi submetida ao Apuramento Final.

Aprovações

Resumo

Aprovações	Total Créditos ECTS	Média	Ano Curricular
11	60.0	17.95	2

Notas

Inscrições	Nota	Peso	Ano Lectivo	
Álgebra Linear	18	6.0	2017/2018	1 Sem.
Cálculo Diferencial e Integral I	17	6.0	2017/2018	1 Sem.
Desenho e Modelação Geométrica I	16	4.5	2017/2018	1 Sem.
Programação	17	6.0	2017/2018	1 Sem.
Química	19	6.0	2017/2018	1 Sem.
Seminário Aeroespacial I	16	1.5	2017/2018	1 Sem.
Cálculo Diferencial e Integral II	17	7.5	2017/2018	2 Sem.
Ciência de Materiais	19	6.0	2017/2018	2 Sem.
Gestão	19	4.5	2017/2018	2 Sem.
Mecânica e Ondas	19	6.0	2017/2018	2 Sem.
Sistemas Digitais	19	6.0	2017/2018	2 Sem.

Aluno: Miguel De Sousa Fazenda (90146) - Bilhete de Identidade (15772581)

Informação do apuramento

A Matrícula do Aluno ainda não foi submetida ao Apuramento Final.

Aprovações

Resumo

Aprovações	Total Créditos ECTS	Média	Ano Curricular
10	60.0	17.23	2

Notas

Inscrições	Nota	Peso	Ano Lectivo	
Álgebra Linear	17	6.0	2017/2018	1 Sem.
Cálculo Diferencial e Integral I	15	6.0	2017/2018	1 Sem.
Portfólio Meec	17	6.0	2017/2018	1 Sem.
Química	17	6.0	2017/2018	1 Sem.
Sistemas Digitais	19	6.0	2017/2018	1 Sem.
Arquitectura de Computadores	18	6.0	2017/2018	2 Sem.
Cálculo Diferencial e Integral II	15	7.5	2017/2018	2 Sem.
Gestão	18	4.5	2017/2018	2 Sem.
Mecânica e Ondas	18	6.0	2017/2018	2 Sem.
Programação	19	6.0	2017/2018	2 Sem.