

```
1 //=====//
2 //                                           //
3 //                                     Atributos de classe                       //
4 //                                           //
5 //=====//
6
7 //Entradas digitais
8 int sensorPortaEsquerdaAberta = 22;
9 int sensorPortaDireitaAberta = 23;
10 int sensorPortaEsquerdaFechada = 24;
11 int sensorPortaDireitaFechada = 25;
12 int sensorAndar1 = 26;
13 int sensorAndar2 = 27;
14 int botaoChamadaCima = 28;
15 int botaoChamadaBaixo = 29;
16 int botaoCabineEmergencia = 30;
17 int botaoCabineBaixo = 31;
18 int botaoCabineCima = 32;
19 int botaoAbrePorta = 33;
20 int botaoFechaPorta = 34;
21
22 //Saídas digitais
23
24 int habilitaMotorPortaEsquerda = 45;
25 int habilitaMotorPortaDireita = 46;
26 int abrePortaEsquerdaA = 47; //Para abrir setar em 1, para fechar setar em 0
27 int abrePortaEsquerdaB = 48; //Para abrir setar em 0, para fechar setar em 1
28 int abrePortaDireitaA = 49;
29 int abrePortaDireitaB = 50;
30 int motorSobe = 51;
31 int motorDesce = 52;
32 int motorPara = 53;
33
34 //Entradas polarizadas (lógica no método setup)z
35 boolean sensorPEA_EP;
36 boolean sensorPEF_EP;
37 boolean sensorPDA_EP;
38 boolean sensorPDF_EP;
39
40
41
42 //=====//
43 //                                           //
44 //                                     Método setup                               //
45 //                                           //
46 //=====//
47
48 void setup() {
49     //Variáveis de classe
50
51     //Seta os pinos como entradas digitais e seta-os como falso.
52     for (int i = 22; i <= 34; i++) {
53         pinMode(i, INPUT);
54         digitalWrite(i, LOW);
55     }
56
57     //Seta os pinos como saídas digitais e seta-os como falso.
58     for (int i = 45; i <= 53; i++) {
59         pinMode(i, OUTPUT);
60         digitalWrite(i, LOW);
61     }
62     Serial.begin(9600);
63     int matrizSensores[] = {22, 23, 24, 25, 26, 27};
64     int matrizBotoes[] = {28, 29, 30, 31, 32, 33, 34};
65
66 }
67 //=====//
68 //                                           //
69 //                                     Método polariza                             //
70 //                                           //
71 //=====//
72
73 void polariza() {
74
75     delay(1000);
76
77     //Exibe o status de cada sensor - físico
78     Serial.print("Esquerda aberta= " + String(digitalRead(sensorPortaEsquerdaAberta)) + " ");
79     Serial.print("Esquerda fechada= " + String(digitalRead(sensorPortaEsquerdaFechada)) + " ");
80     Serial.print("Direita aberta= " + String(digitalRead(sensorPortaDireitaAberta)) + " ");
81     Serial.println("Direita fechada= " + String(digitalRead(sensorPortaDireitaFechada)) + " ");
82
83     //Exibe o status de cada sensor - polarizado
84     Serial.print("Esquerda aberta= " + String(digitalRead(sensorPEA_EP)) + " ");
85     Serial.print("Esquerda fechada= " + String(digitalRead(sensorPEF_EP)) + " ");
86     Serial.print("Direita aberta= " + String(digitalRead(sensorPDA_EP)) + " ");
87     Serial.println("Direita fechada= " + String(digitalRead(sensorPDF_EP)) + " ");
88
89
90     //Lógica de polarização de entradas
91     //Sensores das portas
92
93     //Sensor de porta totalmente aberta. Este sensor é NF. O ponto esperado é 1. Quando comuta para 0 ele foi pressionado.
94     sensorPEA_EP = digitalRead(sensorPortaEsquerdaAberta) ^ 1;
95     //Sensor de porta totalmente aberta. Este sensor é NF. O ponto esperado é 1. Quando comuta para 0 ele foi pressionado.
96     sensorPDA_EP = digitalRead(sensorPortaDireitaAberta) ^ 1;
97     //Sensor de porta totalmente fechada. Este sensor é NA. O ponto esperado é 0. Quando comuta para 1 ele foi pressionado.
98     sensorPEF_EP = digitalRead(sensorPortaEsquerdaFechada) ^ 0;
99     //Sensor de porta totalmente fechada. Este sensor é NA. O ponto esperado é 0. Quando comuta para 1 ele foi pressionado.
```

```
100     sensorPDF_EP = digitalRead(sensorPortaDireitaFechada) ^ 0;
101 }
102
103 //=====//
104 //                                     //
105 //                                     Método abrePortas                                     //
106 //                                     //
107 //=====//
108 void abrePortas(String lado) {
109
110     //Escreve no monitor qual porta foi pedida para ser aberta
111     String str = "Pedido de abertura da porta " + lado;
112     Serial.println(str);
113
114     //Lê os sensores fim-de-curso
115     //Primeiro testa se a porta solicitada foi a da esquerda
116     if (lado.equals("Esquerda")) {
117         //Agora verifica se a porta já se encontra aberta
118         if (sensorPEA_EP) {
119             Serial.println("Porta esquerda totalmente aberta");
120             //Certifica-se de parar o motor de abertura da porta esquerda
121             digitalWrite(abrePortaEsquerdaA, LOW); //Retira pulso para abertura
122             digitalWrite(abrePortaEsquerdaB, LOW); //Retira pulso para abertura
123             digitalWrite(habilitaMotorPortaEsquerda, LOW); //Inibe ponte-h da porta da esquerda
124         }
125         else {
126             //Se a porta da esquerda não se encontrar totalmente aberta o controlador solicita sua abertura
127             Serial.println("Abrindo porta esquerda");
128             //Certifica-se de mandar abrir a porta esquerda
129             digitalWrite(abrePortaEsquerdaA, HIGH); //Ativa pulso para abertura
130             digitalWrite(abrePortaEsquerdaB, LOW); //Retira pulso para abertura
131             digitalWrite(habilitaMotorPortaEsquerda, HIGH); //Habilita ponte-h da porta da esquerda
132         }
133
134         //Depois testa se a porta solicitada foi a da direita
135     } else if (lado.equals("Direita")) {
136         //Agora verifica se a porta já se encontra aberta
137         if (sensorPDA_EP) {
138             Serial.println("Porta direita totalmente aberta");
139             //Certifica-se de parar o motor de abertura da porta esquerda
140             digitalWrite(abrePortaDireitaA, LOW); //Retira pulso para abertura
141             digitalWrite(abrePortaDireitaB, LOW); //Retira pulso para abertura
142             digitalWrite(habilitaMotorPortaDireita, LOW); //Inibe ponte-h da porta da direita
143         }
144         else {
145             //Se a porta da esquerda não se encontrar totalmente aberta o controlador solicita sua abertura
146             Serial.println("Abrindo porta direita");
147             //Certifica-se de mandar abrir a porta esquerda
148             digitalWrite(abrePortaDireitaA, HIGH); //Ativa pulso para abertura
149             digitalWrite(abrePortaDireitaB, LOW); //Retira pulso para abertura
150             digitalWrite(habilitaMotorPortaDireita, HIGH); //Habilita ponte-h da porta da direita
151         }
152     }
153 }
154
155
156 //=====//
157 //                                     //
158 //                                     Método fechaPortas                                     //
159 //                                     //
160 //=====//
161 void fechaPortas(String lado) {
162
163     //Escreve no monitor qual porta foi pedida para ser aberta
164     String str = "Pedido de fechamento da porta " + lado;
165     Serial.println(str);
166
167     //Lê os sensores fim-de-curso
168     //Primeiro testa se a porta solicitada foi ada esquerda
169     if (lado.equals("Esquerda")) {
170         //Agora verifica se a porta já se encontra fechada
171         if (sensorPEF_EP) {
172             Serial.println("Porta esquerda totalmente fechada");
173             //Certifica-se de parar o motor de fechamento da porta esquerda
174             digitalWrite(abrePortaEsquerdaA, LOW); //Retira pulso para fechamento
175             digitalWrite(abrePortaEsquerdaA, LOW); //Retira pulso para fechamento
176             digitalWrite(habilitaMotorPortaEsquerda, LOW); //Inibe ponte-h da porta da esquerda
177         }
178         else {
179             //Se a porta da esquerda não se encontrar totalmente fechada o controlador solicita seu fechamento
180             Serial.println("Fechando porta esquerda");
181             //Certifica-se de mandar fechar a porta esquerda
182             digitalWrite(abrePortaEsquerdaA, LOW); //Ativa pulso para fechamento
183             digitalWrite(abrePortaEsquerdaB, HIGH); //Retira pulso para fechamento
184             digitalWrite(habilitaMotorPortaEsquerda, HIGH); //Habilita ponte-h da porta da esquerda
185         }
186
187         //Depois testa se a porta solicitada foi a da direita
188     } else if (lado.equals("Direita")) {
189         //Agora verifica se a porta já se encontra fechada
190         if (sensorPDA_EP) {
191             Serial.println("Porta direita totalmente fechada");
192             //Certifica-se de parar o motor de abertura da porta esquerda
193             digitalWrite(abrePortaDireitaA, LOW); //Retira pulso para abertura
194             digitalWrite(abrePortaDireitaB, LOW); //Retira pulso para abertura
195             digitalWrite(habilitaMotorPortaDireita, LOW); //Inibe ponte-h da porta da direita
196         }
197         else {
198             //Se a porta da direita não se encontrar totalmente aberta o controlador solicita sua abertura
```

```
199     Serial.println("Fechando porta direita");
200     //Certifica-se de mandar abrir a porta esquerda
201     digitalWrite(abrePortaDireitaA, LOW); //Ativa pulso para abertura
202     digitalWrite(abrePortaDireitaB, HIGH); //Retira pulso para abertura
203     digitalWrite(habilitaMotorPortaDireita, HIGH); //Habilita ponte-h da porta da direita
204 }
205 }
206
207 }
208
209 //=====//
210 //                                           //
211 //                               Método botoes                               //
212 //                                           //
213 //=====//
214
215 void botoes() {
216
217     //Botoes internos da cabine
218     /*Abertura e fechamento das portas
219     Se o botão de abertura da cabine for acionado ele solicita a abertura ou fechamento
220     de ambas as portas até que os sensores sejam todos acionados.*/
221     String lado;
222     if (digitalRead(botaoAbrePorta)) {
223         Serial.println("Botao de abertura aciondao - cabine");
224         do {
225             lado = "Esquerda";
226             abrePortas(lado);
227             lado = "Direita";
228             abrePortas(lado);
229             polariza();
230         }
231         while (!sensorPEA_EP && !sensorPDA_EP);
232     }
233
234     else if (digitalRead(botaoFechaPorta)) {
235         Serial.println("Botao de fechamento aciondao - cabine");
236         do {
237             lado = "Esquerda";
238             fechaPortas(lado);
239             lado = "Direita";
240             fechaPortas(lado);
241             polariza();
242         }
243         while (!sensorPEA_EP && !sensorPDA_EP);
244     }
245     else {
246         //Nenhum botão acionado. Os motores devem parar e deixar o sistema automático os posicionar
247         Serial.println("Nao ha pedido de abertura ou fechamento");
248         digitalWrite(abrePortaEsquerdaA, LOW);
249         digitalWrite(abrePortaEsquerdaB, LOW);
250         digitalWrite(abrePortaDireitaA, LOW);
251         digitalWrite(abrePortaDireitaB, LOW);
252         digitalWrite(habilitaMotorPortaEsquerda, LOW); //Inibe ponte-h da porta da esquerda
253         digitalWrite(habilitaMotorPortaDireita, LOW); //Inibe ponte-h da porta da direita
254     }
255 }
256 }
257
258 //=====//
259 //                                           //
260 //                               Método cronometro                               //
261 //                                           //
262 //=====//
263
264 /* Este método retorna o tempo que um evento demorou, recebendo um valor de tempo como argumento e a
265 condição do evento como booleano*/
266
267 long cronometro(long tempo, boolean evento) {
268
269     long tempoPassado;
270
271     do {
272         tempoPassado = millis() - tempo;
273     }
274     while (evento);
275
276     return tempoPassado;
277 }
278
279 //=====//
280 //                                           //
281 //                               Método loop                               //
282 //                                           //
283 //=====//
284
285 void loop() {
286
287     do {
288
289         //Exibe no monitor o tempo em que o programa está rodando
290         Serial.println("-----");
291         Serial.println(String((millis() / 1000)) + " segundos ativo" );
292         int tempo = 0;
293
294         polariza();
295         botoes();
296     }
297 }
```

```
298     while (Serial.available() > 0);  
299  
300 }  
301  
302  
303
```