

# Detector de ocupação máxima

...

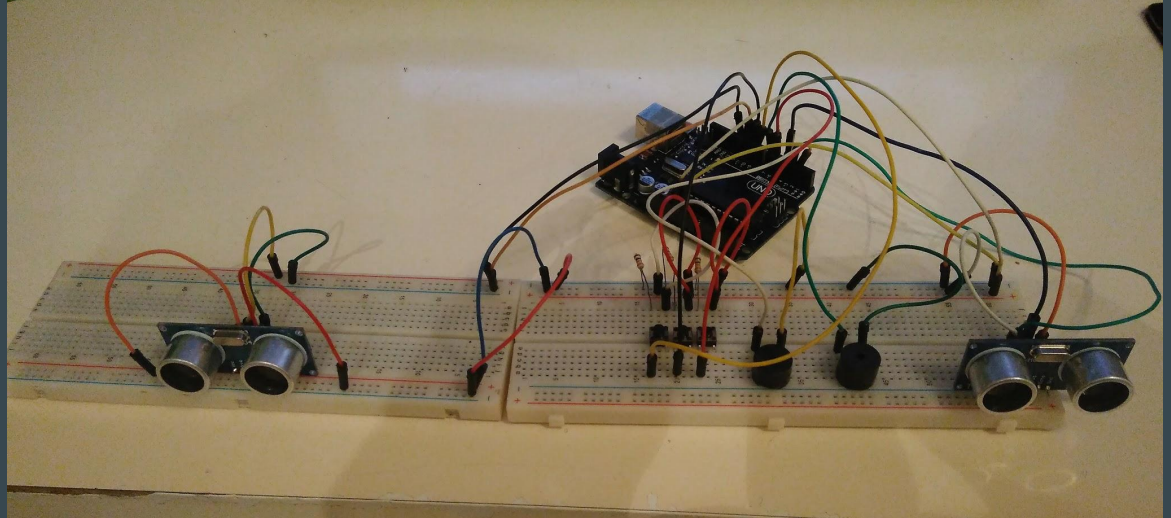
# Propósito

- Controle de pessoas numa sala
- Permite configuração do número máximo
- Acrescenta contador quando alguém entra
- Decrementa quando sai



# Hardware

- 3 botões
- 2 sensores de distância
- 2 beepers
- Não pude usar a LCD



# Código

- Reage a eventos
- Máquina de estados

```
void initialStateButtonChanged(int pin, int value) {  
  if (value == HIGH) {  
    switch (pin) {  
      case addButton:  
        tone(beeperPin, NOTE_A);  
        maxOccupants++;  
        printMaxOccupants();  
        break;  
      case subButton:  
        if (maxOccupants > 0) {
```

```
40 void loop_() {  
41   switch (currentState) {  
42     case initial:  
43       initialStateLoop();  
44       break;  
45     case counting:  
46       countingStateLoop();  
47       break;  
48   }  
49 }  
50  
51 void buttonChanged(int pin, int value) {  
52   switch (currentState) {  
53     case initial:  
54       initialStateButtonChanged(pin, value);  
55       break;  
56     case counting:  
57       countingStateButtonChanged(pin, value);  
58       break;  
59   }  
60 }  
61
```

# Dificuldades

- Distâncias diferentes
- Sensores entram em conflito
- Edge cases
- Trial and error

```
14 16 v 17 14 20 vo. 20 voi 22 void countingStateLoop() {  
15 17 18 21 : 21 : 23 int sampleDistance1 = getDistance(trigPin1, echoPin1);  
16 18 19 22 : 22 : 24 delayMicroseconds(20);  
17 19 20 23 : 23 : 25 int sampleDistance2 = getDistance(trigPin2, echoPin2);  
18 20 21 24 : 24 : 26 if (currentSample < numSamples) {  
19 21 22 25 25 27 distance1 += sampleDistance1/numSamples;  
20 22 23 26 26 28 distance2 += sampleDistance2/numSamples;  
21 23 24 27 27 29 currentSample++;  
22 24 25 28 28 30 return;  
23 25 26 29 29 31 }  
24 26 27 30 30 32 }  
25 27 28 31 31 33 if (debug) {  
26 28 29 32 32 34 Serial.print("Distance 1 ");  
27 29 30 33 33 35 Serial.println(distance1);  
28 30 31 34 34 36 Serial.print("Distance 2 ");  
29 31 32 35 35 37 Serial.println(distance2);  
30 32 33 36 36 38 }  
31 33 34 37 37 39 // Beep when over capacity  
32 34 35 38 38 40 if (currentOccupants > maxOccupants) {  
33 35 36 39 39 41 tone(errorBeeperPin, NOTE_F);  
34 36 37 40 40 42 } else {  
35 37 38 41 41 43 noTone(errorBeeperPin);  
36 38 39 42 42 44 }  
37 39 40 43 43 45 if (shouldChange(lastDistance1, distance1)) {  
38 40 41 44 44 46 sensorChanged(1);  
39 41 42 45 45 47 } else if (shouldChange(lastDistance2, distance2)) {  
40 42 43 46 46 48 sensorChanged(2);  
41 43 44 47 47 49 }  
42 44 45 48 48 50 lastDistance1 = distance1;  
43 45 46 49 49 51 lastDistance2 = distance2;  
44 46 47 50 50 52 distance1 = 0;  
45 47 48 51 51 53 distance2 = 0;  
46 48 49 52 52 54 currentSample = 0;  
47 49 50 53 53 55 }  
48 50 51 54 54 56 }
```

# Soluções

- Solução mais simples foi a melhor
- Usa a média de samples consecutivos
- Reseta os sensors após um intervalo
- Permite configuração de distância mínima