

# PiLarm

**By Tyler Vergin and Leo Liang**

Have you ever  
slept through an  
alarm?

Never Again!

# Inspired By

Nintendo's Alarmo



**Have a partner you  
sleep in the same bed  
as?**

Nintendo says screw you!

---

**Also very  
expensive...**

Nintendo Sound Clock: Alarmo™

\$99.99



---

One thing  
nintendo didn't  
think about...

# You have arms!

(At least you probably do...)





Meet PiLarm!

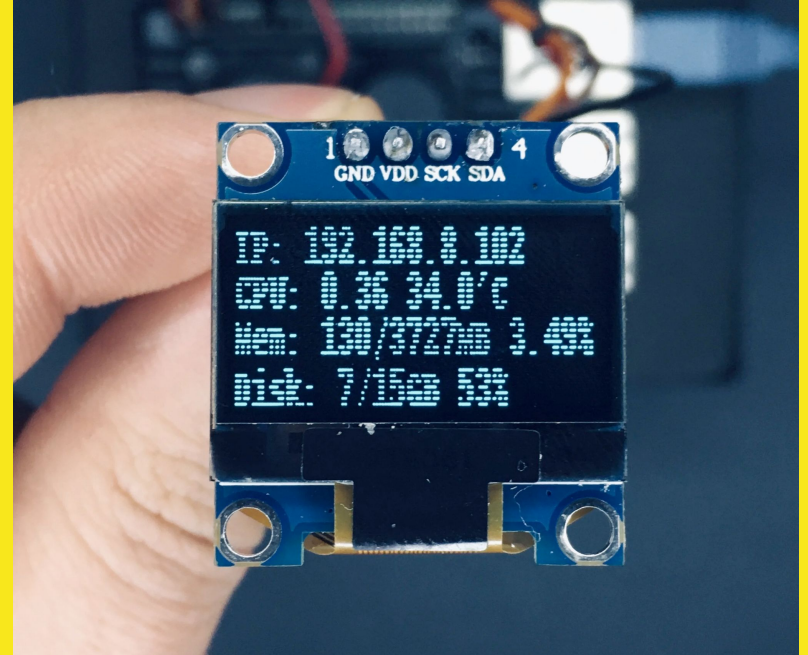
# Just a distance sensor

(14.19 Measuring Distance Using Ultrasound)



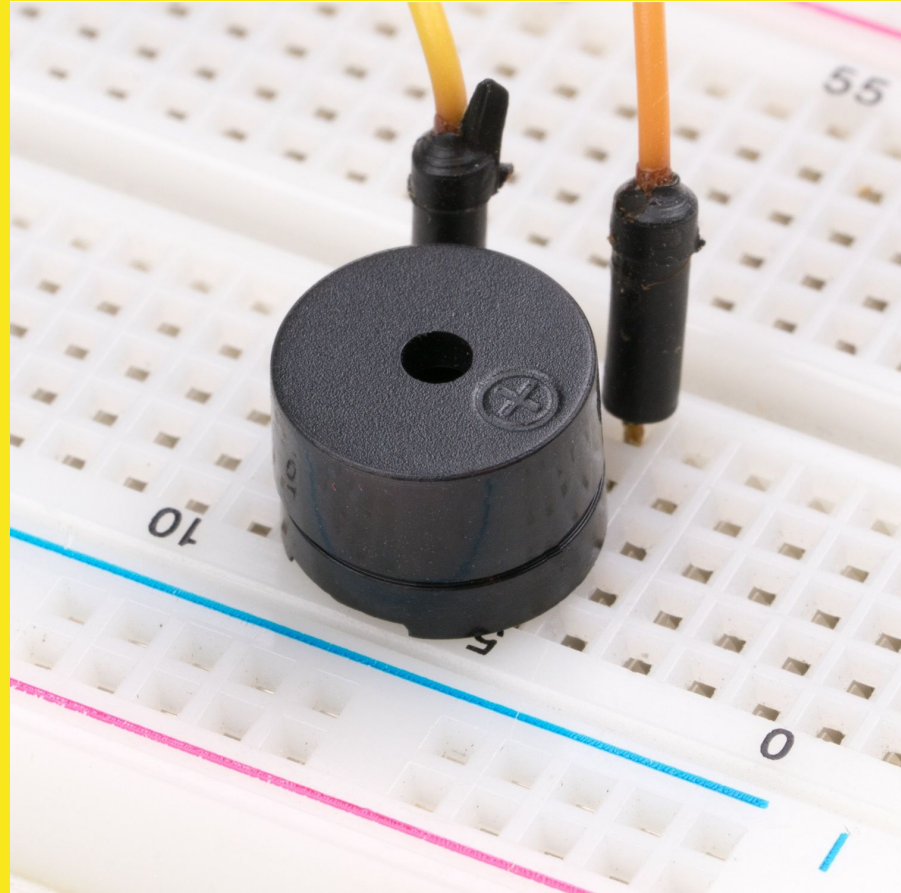
# An OLED Display

(15.4 Using an OLED Graphical Display)



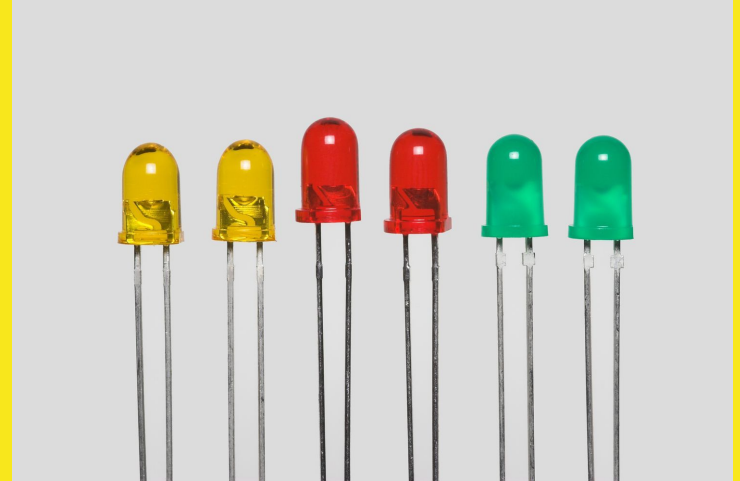
# A Buzzer

(16.7 Making a Buzzing Sound)



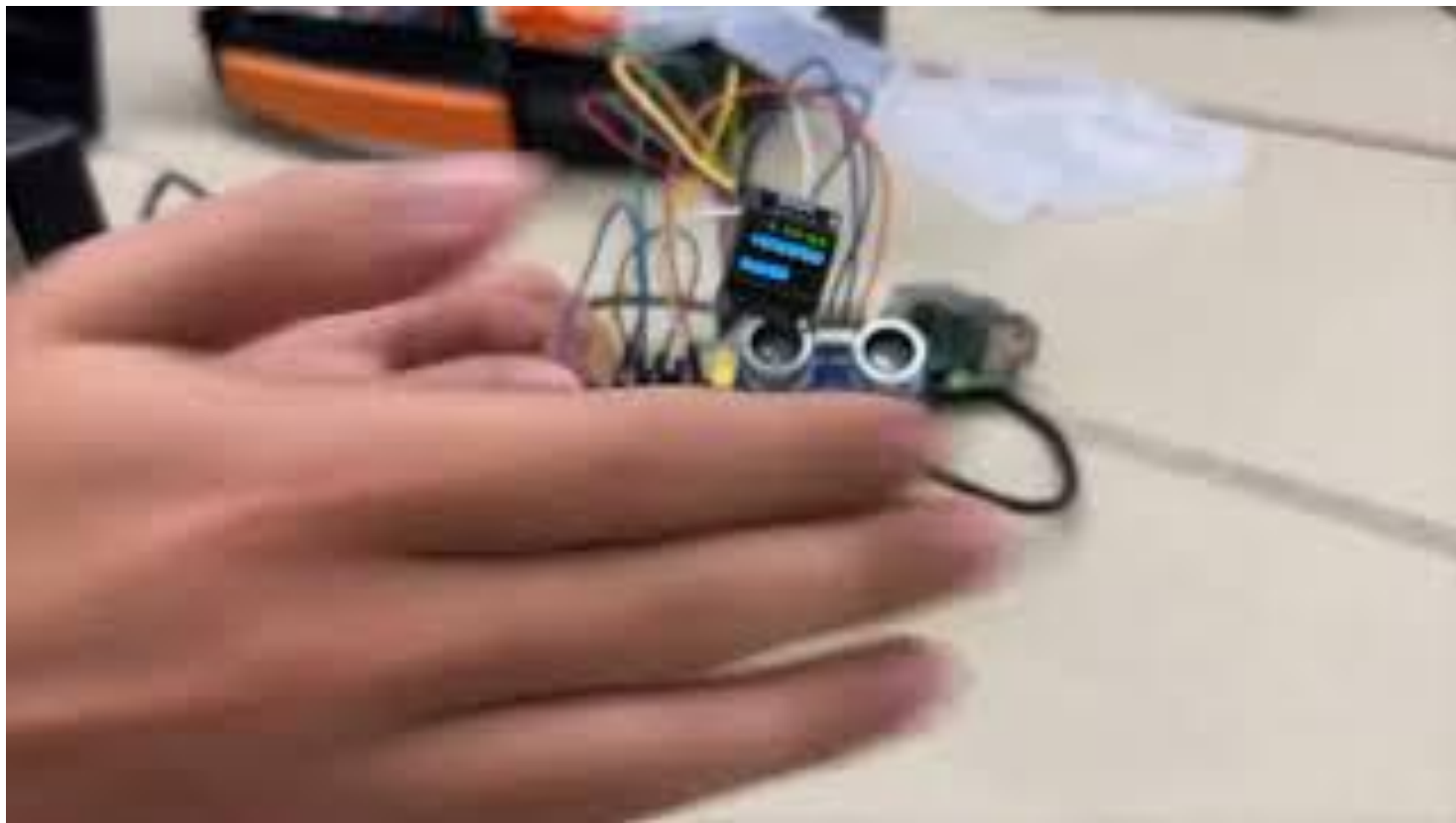
# A Button and an LED

(11.1 Connecting an LED && 13.1 Connecting a Push Switch)



**Come Together...**

---



For PiLarm!

```
Code File Edit Selection View Go Run Terminal Window Help

◆ _final-script.py ×
1 from PiAlarm import PiAlarm
2
3
4
5 pialarm.set_alarm(7, 0) Add to chat (Ctrl+) | Edit highlighted code (Cmd+I).
6 pialarm.run()

◆ PiAlarm.py 3 X
1 import board
2 from PIL import Image, ImageDraw, ImageFont
3 import adafruit_ssd1306
4 from time import sleep
5 from clock import Clock
6 # Add to chat (Ctrl+) | Edit highlighted code (Cmd+I).
7 # button 18, buzzer17, LED27, Ultrasonic(echo22, trigger23)
8
9 class PiAlarm:
10     def __init__(self):
11         self.display = adafruit_ssd1306.SSD1306_I2C(128, 64, board.SCL, board.SDA)
12         self.small_font = ImageFont.truetype('fonts.ttf', 22)
13         self.large_font = ImageFont.truetype('fonts.ttf', 33)
14         self.clock = Clock()
15
16     def setup_display(self):
17         self.display.fill(0)
18         self.display.show()
19
20     def set_alarm(self, h, m):
21         self.clock.set_alarm(h, m)
22
23     def display_message(self, top_line, line_2):
24         width = self.display.width
25         height = self.display.height
26         image = Image.new('1', (width, height))
27         draw = ImageDraw.Draw(image)
28         draw.rectangle((0, 0, width, height), outline=1, fill=0)
29         draw.text((0, 0), top_line, font=self.large_font, fill=255)
30         draw.text((0, 50), line_2, font=self.small_font, fill=255)
31         self.display.image(image)
32         self.display.show()
33
34     def run(self):
35         self.setup_display()
36         while True:
37             self.clock.update()
38             self.display_message(self.clock.get_time_message(), self.clock.get_alarm_message())
39             self.clock.alarming()
40             sleep(0.2)

◆ clock.py 1 X
1 # Clock
2 from alarm import Alarm
3 from gpiozero import Button
4
5 class Clock:
6     def __init__(self):
7         self.time = datetime.now()
8         self.hour = self.time.hour
9         self.minute = self.time.minute
10         self.second = self.time.second
11         self.alarm_time = (0, 0)
12         self.alarm = Alarm()
13         self.armed = True
14         self.is_snoozed = False
15         self.snooze_time = None
16
17     def set_alarm(self, hour, minute):
18         self.alarm_time = (hour, minute)
19         print('Alarm set to: ' + str(hour) + ':' + str(minute))
20
21     def set_waving_time(self, waving_time):
22         self.waving_time = waving_time
23
24     def alarm_check(self):
25         if self.is_snoozed:
26             return self.hour == self.snooze_time[0] and self.minute == self.snooze_time[1]
27         else:
28             return self.hour == self.alarm_time[0] and self.minute == self.alarm_time[1]
29
30     def alarming(self):
31         if self.alarm_check() and self.armed:
32             print('Alarming!')
33             self.alarm.ring()
34             self.is_snoozed = self.alarm.snoozed
35             if not self.is_snoozed:
36                 self.armed = False
37             else:
38                 self.snooze()
39
40     def update(self):
41         self.time = datetime.now()
42         self.hour = self.time.hour
43         self.minute = self.time.minute
44         self.is_snoozed = self.alarm.snoozed
45         self.armed = self.alarm.armed
46
47     def get_alarm_message(self):
48         if not self.is_snoozed:
49             armed = 'Armed' if self.armed else 'Disarmed'
50             return armed + '!' + str(self.alarm_time[0]) + ':' + str(self.alarm_time[1]) + '!' + str(self.is_snoozed)
51         return 'Snoozed' + '!' + str(self.snooze_time[0]) + ':' + str(self.snooze_time[1]) + '!' + str(self.is_snoozed)
52
53     def get_time_message(self):
54         return 'Time: ' + str(self.time.strftime('%H:%M:%S'))
55
56     def next_snooze_time(self, snooze_time):
57         min = self.snooze_time[1] + snooze_time
58         snoozed = self.snooze_time[0] + int(min // 60)
59         snoozed = min % 60
60         return (snoozed, snooze_time)
61
62     def snooze(self, minutes):
63         if self.snooze_time == None:
64             self.is_snoozed = True
65             self.snooze_time = (self.hour, self.minute)
66         self.snooze_time = self.next_snooze_time(minutes)

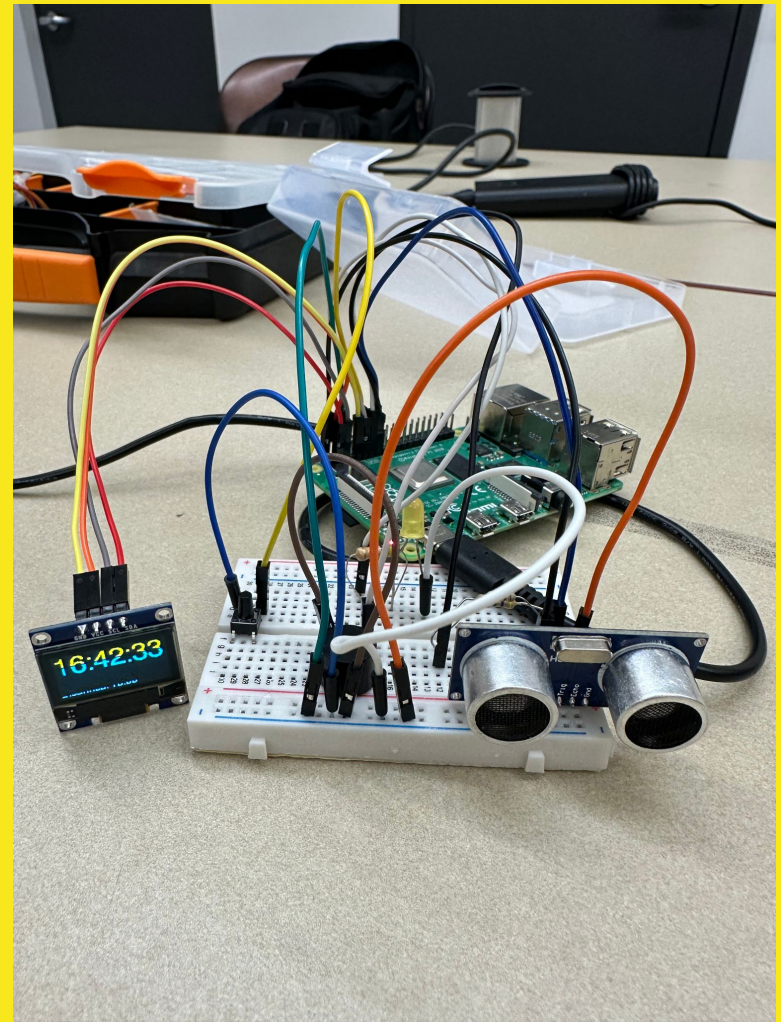
◆ alarm.py 1 X
1 # Alarm
2 from gpiozero import Button, LED, Buzzer, DistanceSensor
3 from time import sleep
4
5 class Alarm:
6     def __init__(self):
7         self.button = Button(18)
8         self.led = LED(27)
9         self.buzzer = Buzzer(17)
10         self.buzzer.on = True
11         self.sensor = DistanceSensor(echo22, trigger23)
12         self.is_ringing = False
13         self.snoozed = None
14         self.armed = True
15
16     def ring(self):
17         t1 = threading.Thread(target = self.ringing)
18         t2 = threading.Thread(target = self.counting_wave, args = (5,))
19         t3 = threading.Thread(target = self.counting_button, args = (10,))
20         t1.start()
21         t2.start()
22         t3.start()
23         t1.join()
24         t2.join()
25         print('Snoozed: ' + str(self.snoozed))
26
27     def ringing(self):
28         self.is_ringing = True
29         while self.is_ringing:
30             print('Ring!')
31             self.led.on()
32             if self.buzzer.on:
33                 self.buzzer(1500, 0.5)
34                 sleep(0.5)
35                 self.led.off()
36                 sleep(0.5)
37
38     def buzz(self, pitch, duration):
39         period = 1.0 / pitch
40         cycles = int(duration * pitch)
41         self.buzzer.beep(on_timeperiod, off_timeperiod, n=cycles/2)
42
43     def counting_wave(self, wave_time):
44         current_wave = 0
45         distance = self.sensor.distance
46         while current_wave < wave_time and self.is_ringing:
47             print(self.sensor.distance)
48             if abs(distance - self.sensor.distance) > .5:
49                 distance = self.sensor.distance
50                 current_wave += 1
51                 print('Wave: ' + str(current_wave))
52                 sleep(0.2)
53
54         self.is_ringing = False
55         if not self.snoozed == False:
56             self.snoozed = True
57
58     def counting_button(self, presses = 10):
59         count = 0
60         while count < presses:
61             if self.button.value == 1:
62                 print('Button pressed:')
63                 count += 1
64                 sleep(0.2)
65
66         self.is_ringing = False
67         self.snoozed = False
68         self.armed = False
69
70     def sleep(self):
71         if self.button.value == 1:
72             self.led = True
73
Ln 8, Col 1 (selected) Spaces 4 UTF-8 LF Python 3.12.2 64-bit ✓ Continue
```

How PiAlarm works behind the scenes



# First Came the Wires

15 different wires



# And lots of coding later...

◆ PiAlarm.py 3 X

```
◆ PiAlarm.py > ...
1 import board
2 from PIL import Image, ImageDraw, ImageFont
3 import adafruit_ssd1306
4 from time import sleep
5 from clock import Clock
6
7 # button 18, buzzer:17, LED:27, Ultrasonic(echo:22, trigger: 23)
8
9 class PiAlarm:
10     def __init__(self):
11         self.display = adafruit_ssd1306.SSD1306_I2C(128, 64, board.I2C(), addr=0x3C)
12         self.small_font = ImageFont.truetype('FreeSans.ttf', 12)
13         self.large_font = ImageFont.truetype('FreeSans.ttf', 33)
14         self.clock = Clock()
15
16     def setup_display(self):
17         self.display.fill(0)
18         self.display.show()
19
20     def set_alarm(self, h,m):
21         self.clock.set_alarm(h,m)
22
23     def display_message(self, top_line, line_2):
24         width = self.display.width
25         height = self.display.height
26         image = Image.new('1', (width, height))
27         draw = ImageDraw.Draw(image)
28         draw.rectangle((0,0,width,height), outline=0, fill=0)
29         draw.text((0, 0), top_line, font=self.large_font, fill=255)
30         draw.text((0, 50), line_2, font=self.small_font, fill=255)
31         self.display.image(image)
32         self.display.show()
33
34     def run(self):
35         self.setup_display()
36         self.set_alarm(16,6)
37         while True:
38             self.clock.update()
39             self.display_message(self.clock.get_time_message(), self.clock.get_alarm_message())
40             self.clock.alarming()
41             sleep(0.2)
```

◆ \_final-scrip.py X

◆ \_final-scrip.py > ...

```
1 from PiAlarm import PiAlarm
2
3 pialarm = PiAlarm()
4 pialarm.run()
```

◆ alarm.py 1 X

```
◆ alarm.py > ...
1 from gpiozero import Button, LED, Buzzer, DistanceSensor
2 from time import sleep
3 import threading
4
5 class Alarm:
6     def __init__(self):
7         self.button = Button(18)
8         self.led = LED(27)
9         self.buzzer = Buzzer(17)
10        self.buzzer_on = True
11        self.sensor = DistanceSensor(echo=22, trigger=23)
12        self.is_ringing = False
13        self.snoozed = None
14        self.armed = True
15
16    def ring(self):
17        t1 = threading.Thread(target = self.ringing)
18        t2 = threading.Thread(target = self.counting_wave, args = (5,))
19        t3 = threading.Thread(target = self.counting_button, args= (10,))
20        t1.start()
21        t2.start()
22        t3.start()
23        t1.join()
24        t2.join()
25        print("snoozed: " + str(self.snoozed))
26
27    def ringing(self):
28        self.is_ringing = True
29        while self.is_ringing:
30            print('Ring!!')
31            self.led.on()
32            if self.buzzer_on:
33                self.buzz(1500, 0.5)
34            sleep(0.5)
35            self.led.off()
36            sleep(0.5)
37
38    def buzz(self, pitch, duration):
39        period = 1.0 / pitch
40        cycles = int(duration * pitch)
41        self.buzzer.beep(on_time=period, off_time=period, n=cycles/2)
42
43    def counting_wave(self, wave_time):
44        current_wave = 0
45        distance = self.sensor.distance
46        while current_wave < wave_time and self.is_ringing:
47            print(self.sensor.distance)
48            if abs(distance - self.sensor.distance) > .5:
49                distance = self.sensor.distance
50            current_wave += 1
51            print('Wave: {}'.format(current_wave))
52            sleep(0.2)
53
54        self.is_ringing = False
55        if not self.snoozed == False:
56            self.snoozed = True
57
58    def counting_button(self, presses = 10):
59        count = 0
60        while count < presses:
61            if self.button.value == 1:
62                print('Button pressed!')
63                count += 1
64                sleep(0.2)
65
66        self.is_ringing = False
67        self.snoozed = False
68        self.armed = False
69
70        sleep(60)
71        if self.button.value == 1:
72            self.armed = True
73
```

◆ clock.py 1 X

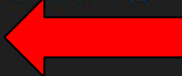
```
◆ clock.py > ...
1 from datetime import datetime
2 from alarm import Alarm
3 from gpiozero import Button
4
5 class Clock:
6     def __init__(self):
7         self.time = datetime.now()
8         self.hour = self.time.hour
9         self.minute = self.time.minute
10        self.second = self.time.second
11        self.alarm_time = (0, 0)
12        self.alarm = Alarm()
13        self.armed = True
14        self.isSnoozed = False
15        self.snooze_time = None
16
17    def set_alarm(self, hour, minute):
18        self.alarm_time = (hour, minute)
19        print('Alarm set to: ' + str(hour)+':'+str(minute))
20
21    def set_waving_time(self, waving_time):
22        self.waving_time = waving_time
23
24    def alarm_check(self):
25        if self.isSnoozed:
26            return self.hour == self.snooze_time[0] and self.minute == self.snooze_time[1]
27        else:
28            return self.hour == self.alarm_time[0] and self.minute == self.alarm_time[1]
29
30    def alarming(self):
31        if self.alarm_check() and self.armed:
32            print('Alarming!')
33            self.alarm.ring()
34            self.isSnoozed = self.alarm.snoozed
35            if not self.isSnoozed:
36                self.armed = False
37            else:
38                self.snooze()
39
40    def update(self):
41        self.time = datetime.now()
42        self.hour = self.time.hour
43        self.minute = self.time.minute
44        self.isSnoozed = self.alarm.snoozed
45        self.armed = self.alarm.armed
46
47    def get_alarm_message(self):
48        if not self.isSnoozed:
49            armed = 'Armed' if self.armed else 'Disarmed'
50            return armed + ': ' + str(self.alarm_time[0]).zfill(2) + ':' + str(self.alarm_time[1]).zfill(2)
51            return "Snoozed: " + str(self.snooze_time[0]).zfill(2) + ':' + str(self.snooze_time[1]).zfill(2)
52
53    def get_time_message(self):
54        return '(:MM:MM:SS)'.format(self.time)
55
56    def nextSnoozeTime(self, snooze_time):
57        min = self.snooze_time[1] + snooze_time
58        snoozeH = self.snooze_time[0] + int(min / 60)
59        snoozeM = min % 60
60        return [snoozeH, snoozeM]
61
62    def snooze(self, minutes=0):
63        if self.snooze_time == None:
64            self.isSnoozed = True
65            self.snooze_time = (self.hour, self.minute)
66
67        self.snooze_time = self.nextSnoozeTime(minutes)
68
69
```

## With Many Bugs

Turns out CPU usage is important and can cause weird problems, thank god for the sleep() function

```
def counting_wave(self, wave_time):
    current_wave = 0
    distance = self.sensor.distance
    while current_wave < wave_time and self.is_ringing:
        print(self.sensor.distance)
        if abs(distance - self.sensor.distance) > .5:
            distance = self.sensor.distance
            current_wave += 1
            print('Wave: {}'.format(current_wave))
        sleep(0.2)

    self.is_ringing = False
    if not self.snoozed == False:
        self.snoozed = True
```



Very Important


## And Some Multithreading

For when you need to do two things at once. Like for example ringing an alarm and checking for when the alarm should stop ringing.

```
def ring(self):  
    t1 = threading.Thread(target = self.ringing)  
    t2 = threading.Thread(target = self.counting_wave, args = (5,))  
    t3 = threading.Thread(target = self.counting_button, args= (10,))  
    t1.start()  
    t2.start()  
    t3.start()  
    t1.join()  
    t2.join()  
    print("snoozed: " + str(self.snoozed))
```

## And How we Count

If the distance between two readings varies by more than .5 meters it adds a wave to the count. The max distance it can read is one meter so it must be mounted by your headboard.



```
def counting_wave(self, wave_time):
    current_wave = 0
    distance = self.sensor.distance
    while current_wave < wave_time and self.is_ringing:
        print(self.sensor.distance)
        if abs(distance - self.sensor.distance) > .5:
            distance = self.sensor.distance
            current_wave += 1
            print('Wave: {}'.format(current_wave))
            sleep(0.2)

    self.is_ringing = False
    if not self.snoozed == False:
        self.snoozed = True

def counting_button(self, presses = 10):
    count = 0
    while count < presses:
        if self.button.value == 1:
            print('Button pressed!')
            count += 1
            sleep(0.2)

    self.is_ringing = False
    self.snoozed = False
    self.armed = False

    sleep(60)
    if self.button.value == 1:
        self.armed = True
```

***Any Questions?***

---

**Thank You!**

---