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...)

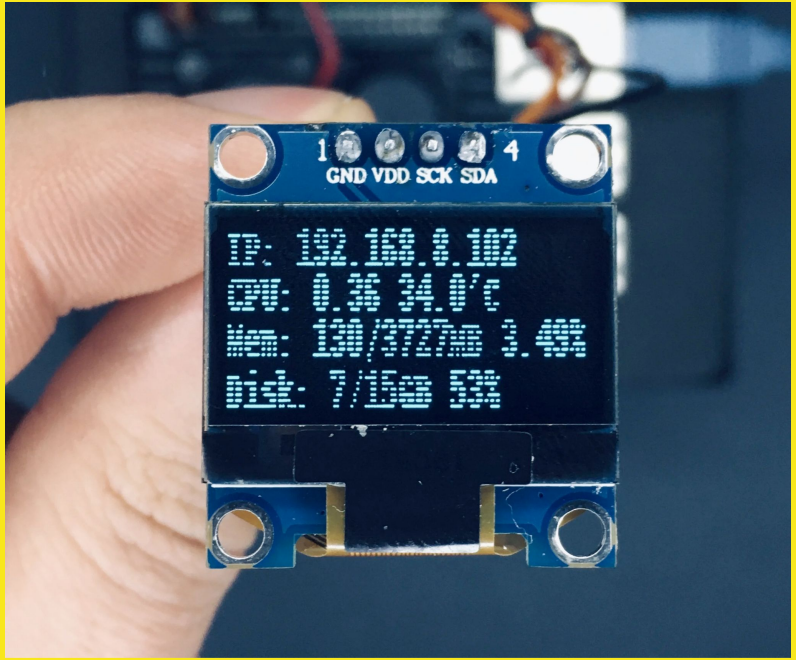
Meet PiLarm!

Just a distance sensor

(14.19 Measuring Distance Using Ultrasound)

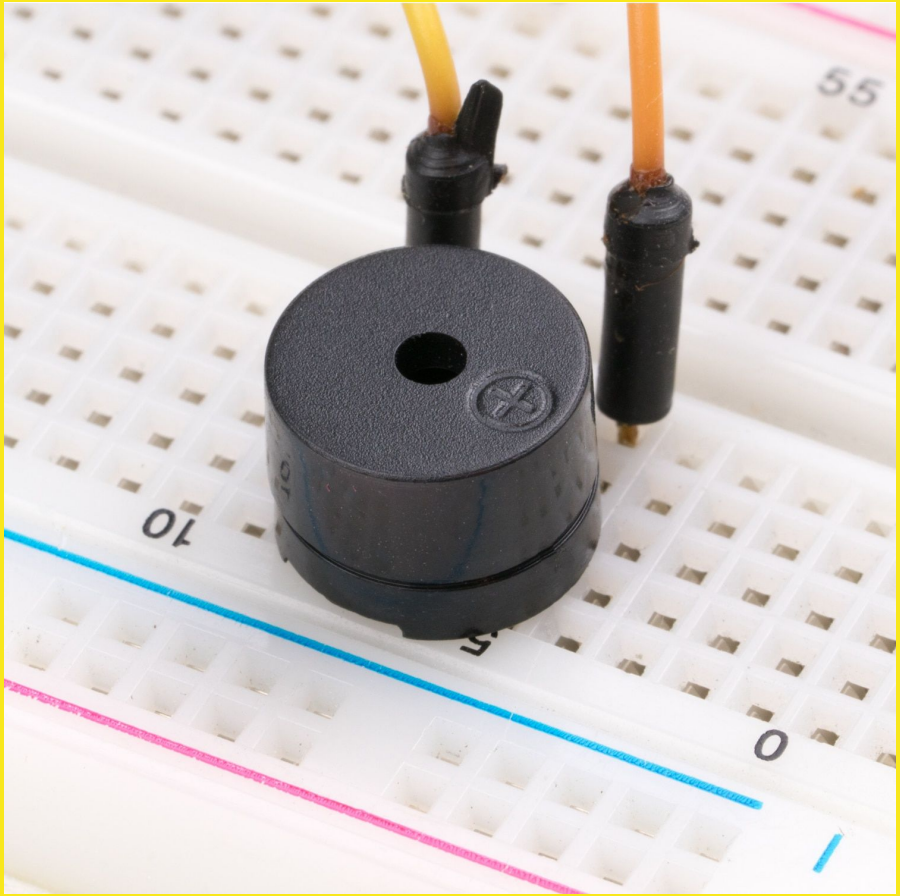
An OLED Display

(15.4 Using an OLED Graphical Display)



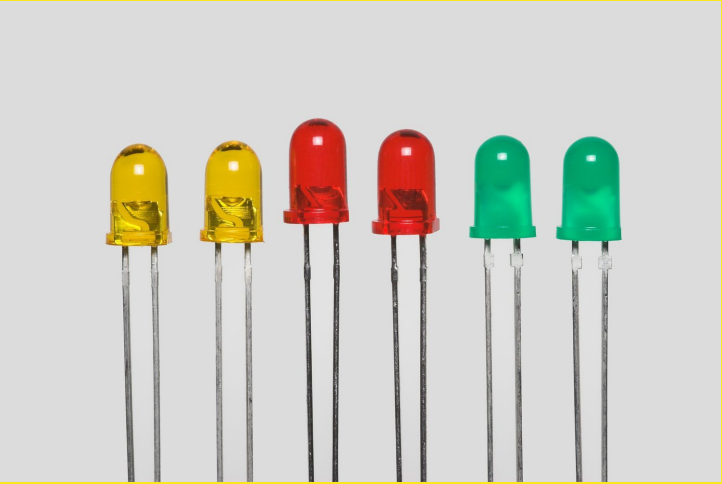
A Buzzer

(16.7 Making a Buzzing Sound)

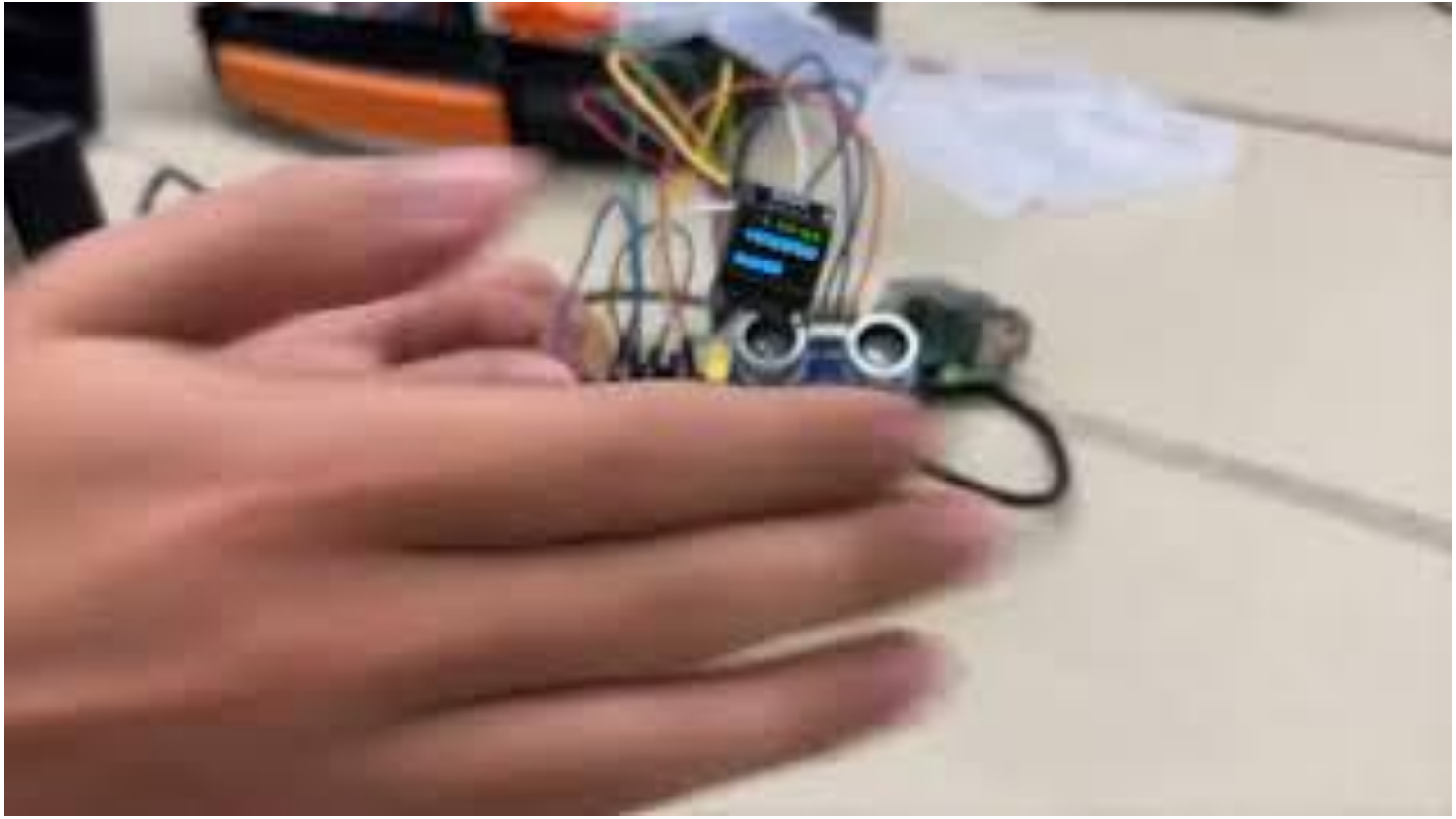


A Button and an LED

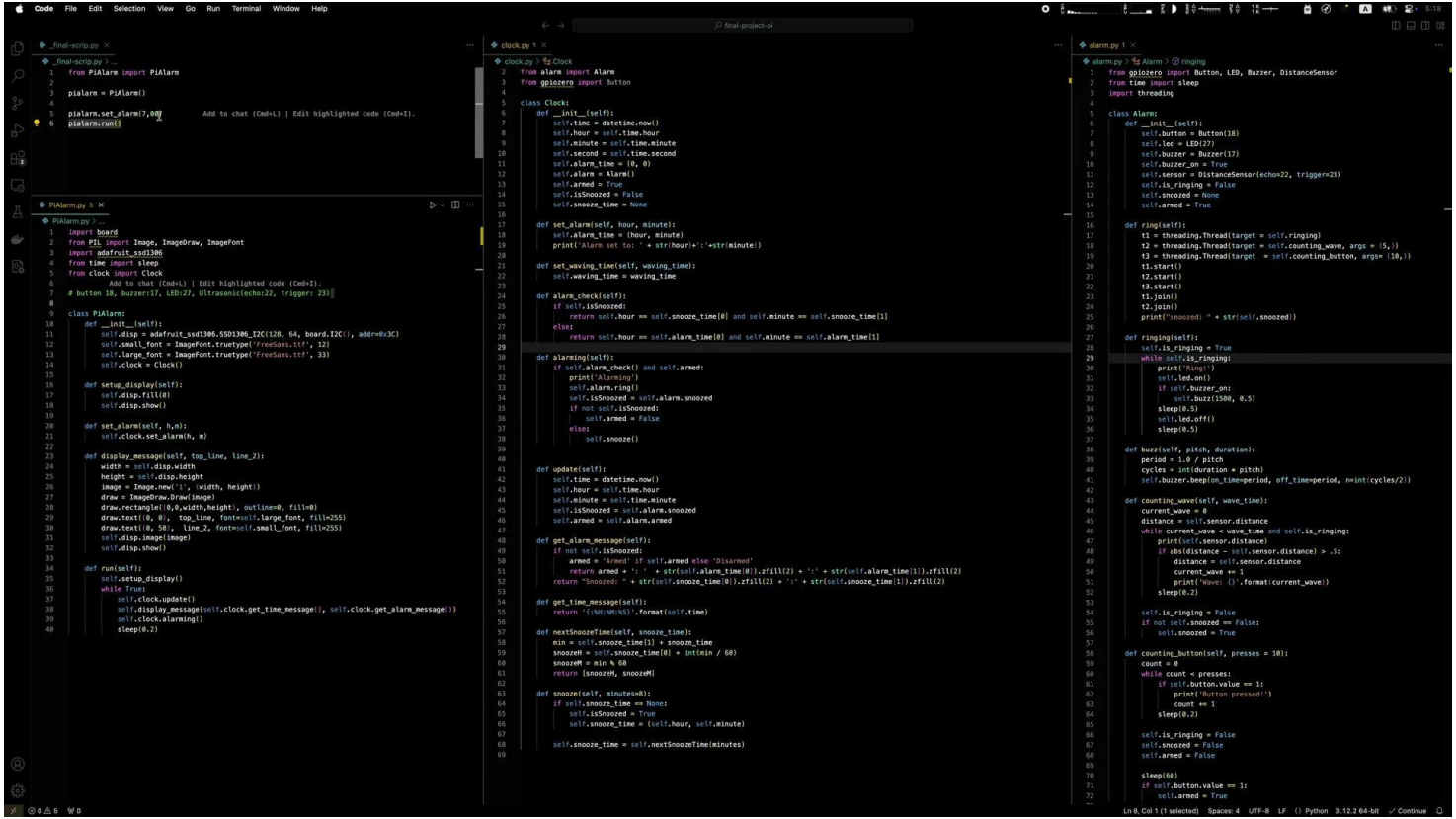
(11.1 Connecting an LED && 13.1 Connecting a Push Switch)



Come Together…



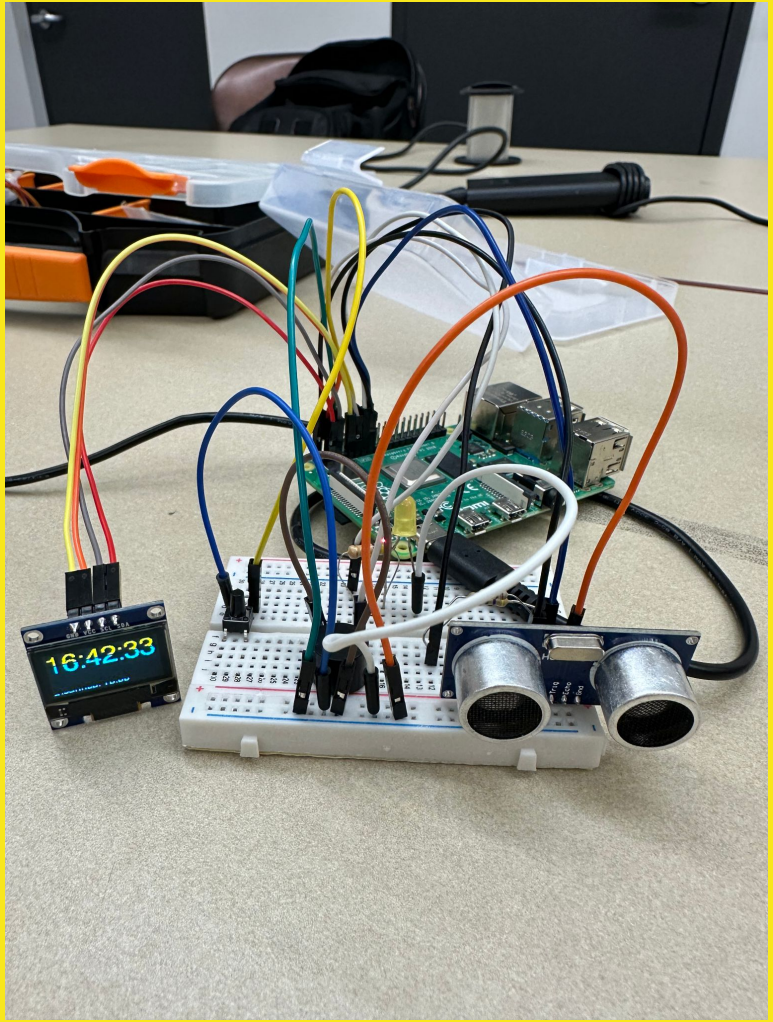
For PiLarm!



How Pilarm works behind the scenes

First Came the Wires

15 different wires



And lots of coding later...

PiAlarm.py 3

PiAlarm.py > ...

```python
import board
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306
from time import sleep
from clock import Clock

# button 18, buzzer:17, LED:27, Ultrasonic(echo:22, trigger: 23)

class PiAlarm:
    def __init__(self):
        self.disp = adafruit_ssd1306.SSD1306_I2C(128, 64, board.I2C(), addr=0x3C)
        self.small_font = ImageFont.truetype('FreeSans.ttf', 12)
        self.large_font = ImageFont.truetype('FreeSans.ttf', 33)
        self.clock = Clock()

    def setup_display(self):
        self.disp.fill(0)
        self.disp.show()

    def set_alarm(self, h,m):
        self.clock.set_alarm(h, m)

    def display_message(self, top_line, line_2):
        width = self.disp.width
        height = self.disp.height
        image = Image.new('1', (width, height))
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.text((0, 0),  top_line, font=self.large_font, fill=255)
        draw.text((0, 50),  line_2, font=self.small_font, fill=255)
        self.disp.image(image)
        self.disp.show()

    def run(self):
        self.setup_display()
        self.set_alarm(16,6)
        while True:
            self.clock.update()
            self.display_message(self.clock.get_time_message(), self.clock.get_alarm_message())
            self.clock.alarming()
            sleep(0.2)
```

_final-scrip.py

_final-scrip.py > ...

```python
from PiAlarm import PiAlarm

pialarm = PiAlarm()
pialarm.run()
```

alarm.py 1

alarm.py > ...

```python
from gpiozero import Button, LED, Buzzer, DistanceSensor
from time import sleep
import threading

class Alarm:
    def __init__(self):
        self.button = Button(18)
        self.led = LED(27)
        self.buzzer = Buzzer(17)
        self.buzzer_on = True
        self.sensor = DistanceSensor(echo=22, trigger=23)
        self.is_ringing = False
        self.snoozed = None
        self.armed = True

    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("snoosed: " + str(self.snoozed))

    def ringing(self):
        self.is_ringing = True
        while self.is_ringing:
            print('Ring!')
            self.led.on()
            if self.buzzer_on:
                self.buzz(1500, 0.5)
            sleep(0.5)
            self.led.off()
            sleep(0.5)

    def buzz(self, pitch, duration):
        period = 1.0 / pitch
        cycles = int(duration * pitch)
        self.buzzer.beep(on_time=period, off_time=period, n=int(cycles/2))

    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
```

clock.py 1

clock.py > ...

```python
from datetime import datetime
from alarm import Alarm
from gpiozero import Button

class Clock:
    def __init__(self):
        self.time = datetime.now()
        self.hour = self.time.hour
        self.minute = self.time.minute
        self.second = self.time.second
        self.alarm_time = (0, 0)
        self.alarm = Alarm()
        self.armed = True
        self.isSnoozed = False
        self.snooze_time = None

    def set_alarm(self, hour, minute):
        self.alarm_time = (hour, minute)
        print('Alarm set to: ' + str(hour)+':'+str(minute))

    def set_waving_time(self, waving_time):
        self.waving_time = waving_time

    def alarm_check(self):
        if self.isSnoozed:
            return self.hour == self.snooze_time[0] and self.minute == self.snooze_time[1]
        else:
            return self.hour == self.alarm_time[0] and self.minute == self.alarm_time[1]

    def alarming(self):
        if self.alarm_check() and self.armed:
            print('Alarming')
            self.alarm.ring()
            self.isSnoozed = self.alarm.snoozed
            if not self.isSnoozed:
                self.armed = False
            else:
                self.snooze()


    def update(self):
        self.time = datetime.now()
        self.hour = self.time.hour
        self.minute = self.time.minute
        self.isSnoozed = self.alarm.snoozed
        self.armed = self.alarm.armed

    def get_alarm_message(self):
        if not self.isSnoozed:
            armed = 'Armed' if self.armed else 'Disarmed'
            return armed + ': '  + str(self.alarm_time[0]).zfill(2) + ':' + str(self.alarm_time[1]).zfill(2)
        return "Snoozed: " + str(self.snooze_time[0]).zfill(2) + ':' + str(self.snooze_time[1]).zfill(2)

    def get_time_message(self):
        return '{:%H:%M:%S}'.format(self.time)

    def nextSnoozeTime(self, snooze_time):
        min = self.snooze_time[1] + snooze_time
        snoozeH = self.snooze_time[0] + int(min / 60)
        snoozeM = min % 60
        return [snoozeH, snoozeM]

    def snooze(self, minutes=8):
        if self.snooze_time == None:
            self.isSnoozed = True
            self.snooze_time = (self.hour, self.minute)

        self.snooze_time = self.nextSnoozeTime(minutes)
```

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!