```
import ujson
from machine import Pin, I2C, ADC
from math import sin, cos, radians
from time import sleep
import framebuf
from ssd1306 import SSD1306_I2C
from ir_rx.nec import NEC_16
import utime
import time

# Constants and initialization
FLASH_FILE = 'settings.json'
volume_level = 50  # Default volume level

# Functions to load and save volume level
def load_settings():
    global volume_level
    try:
        with open(FLASH_FILE, 'r') as f:
            settings = ujson.load(f)
            volume_level = settings.get('volume_level', 50)  # Default to 50 if not found
    except OSError:
        # File doesn't exist or can't be read, using default values
        volume_level = 50

def save_settings():
    settings = {'volume_level': volume_level}
    with open(FLASH_FILE, 'w') as f:
        ujson.dump(settings, f)

# Initialize OLED
OLEDinit = Pin(15, Pin.OUT)
OLEDinit.low()
sleep(0.2)
OLEDinit.high()

OLEDinit = Pin(2, Pin.OUT)
OLEDinit.low()
OLEDinit = Pin(3, Pin.OUT)
OLEDinit.high()

i2c = I2C(0, sda=Pin(0), scl=Pin(1))

# Scan the I2C bus for devices
devices = i2c.scan()

# Addresses for the two OLED displays
OLED_ADDR1 = 0x3C
OLED_ADDR2 = 0x3D

if devices:
    print('I2C devices found:', [hex(device) for device in devices])
else:
    print('No I2C devices found')

# Initialize the OLED displays
oled1 = SSD1306_I2C(128, 64, i2c, addr=OLED_ADDR1)
oled2 = SSD1306_I2C(128, 64, i2c, addr=OLED_ADDR2)

# Load saved settings
load_settings()

# Constants for OLED display
OLED_WIDTH = 128
OLED_HEIGHT = 64

# Define the pins for the left and right inputs
left_pin = 26  # GPIO26, can be adjusted based on your connection
right_pin = 28
```

```
70
71  # Set up the ADC for the left input
72  left_adc = ADC(Pin(left_pin))
73  right_adc = ADC(Pin(right_pin))
74
75  # Sample window in milliseconds
76  sample_window = 5
77
78  # Meter parameters
79  hMeter = 65
80  vMeter = 85
81  rMeter = 80
82
83  # Initial values for peaks and signal min/max
84  PeakLeft = 400
85  PeakRight = 400
86  fallTimeLeft = 0
87  falltimeright = 0
88
89
90  # Constants for menu
91  MENU_ITEMS = ["Volume Control", "VU Meter"]
92  menu_index = 0
93
94  # Global variable to control the state
95  current_state = 'menu'
96
97  def callback(data, addr, ctrl):
98      if data > 0:
99          decoded_value = decodeKeyValue(data)
100         handle_ir_command(decoded_value)
101         print(decoded_value)
102
103 ir = NEC_16(Pin(14, Pin.IN), callback)
104
105 def decodeKeyValue(data):
106     if data == 0x16:  # Button 1
107         return "1"
108     if data == 0x19:  # Button 2
109         return "2"
110     if data == 0x44:  # DOWN button (decrease volume)
111         return "DOWN"
112     if data == 0x43:  # UP button (increase volume)
113         return "UP"
114     if data == 0x4A:  # # key (exit to menu)
115         return "EXIT"
116     return "UNKNOWN"
117
118 def handle_ir_command(command):
119     global current_state
120     if command == "1":
121         if current_state == 'menu':
122             current_state = 'volume'
123     elif command == "2":
124         if current_state == 'menu':
125             current_state = 'vu_meter'
126     elif command == "DOWN":
127         if current_state == 'volume':
128             decrease_volume()
129     elif command == "UP":
130         if current_state == 'volume':
131             increase_volume()
132     elif command == "EXIT":
133         if current_state in ['volume', 'vu_meter']:
134             current_state = 'menu'
135
136 # VU meter background mask image
137 VUMeter = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
```

```
138     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00,
139     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00,
140     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00,
141     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00,
142     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x03, 0x00, 0x60, 0xC0, 0x00, 0x00,
        0x00, 0x00,
143     0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x09, 0x04, 0x80, 0x21, 0x20, 0x00, 0x00,
        0x00, 0x00,
144     0x00, 0x00, 0x00, 0x01, 0x98, 0x08, 0x06, 0x03, 0x80, 0x21, 0x20, 0x00, 0x00,
        0x00, 0x00,
145     0x00, 0x00, 0x00, 0x00, 0xA4, 0x10, 0x09, 0x00, 0x80, 0x21, 0x20, 0x07, 0x00,
        0x00, 0x00,
146     0x00, 0x00, 0x00, 0x00, 0xA4, 0x10, 0x06, 0x03, 0x00, 0x20, 0xC0, 0x00, 0x80,
        0x00, 0x00,
147     0x00, 0x00, 0x71, 0x80, 0xA4, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00,
        0x00, 0x00,
148     0x00, 0x00, 0x0A, 0x40, 0x98, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
        0x3C, 0x00,
149     0x00, 0x00, 0x3A, 0x40, 0x00, 0x00, 0x02, 0x01, 0x00, 0x40, 0x80, 0x07, 0x00,
        0x20, 0x00,
150     0x00, 0x00, 0x42, 0x40, 0x00, 0x08, 0x02, 0x01, 0x08, 0x40, 0x80, 0x00, 0x00,
        0x38, 0x00,
151     0x00, 0x00, 0x79, 0x80, 0x04, 0x08, 0x02, 0x01, 0x08, 0x81, 0x10, 0x00, 0x00,
        0x04, 0x00,
152     0x00, 0x00, 0x00, 0x00, 0x04, 0x08, 0x02, 0x01, 0x08, 0x81, 0x11, 0x04, 0x00,
        0x38, 0x00,
153     0x00, 0x00, 0x00, 0x00, 0x02, 0x04, 0x02, 0x01, 0x08, 0x81, 0x21, 0x04, 0x00,
        0x00, 0x08,
154     0x00, 0x00, 0x00, 0x84, 0x02, 0x04, 0x0F, 0xFF, 0xFF, 0xC3, 0xE2, 0x04, 0x00,
        0x00, 0x08,
155     0x00, 0x00, 0x00, 0xC2, 0x01, 0x07, 0xF0, 0x00, 0x00, 0x3B, 0xFE, 0x08, 0x40,
        0x40, 0x08,
156     0x00, 0xFE, 0x00, 0x62, 0x01, 0xF8, 0x00, 0x00, 0x00, 0x03, 0xFF, 0xE8, 0x40,
        0x80, 0x7F,
157     0x00, 0x00, 0x00, 0x21, 0x1E, 0x00, 0x04, 0x00, 0x80, 0x00, 0x7F, 0xFE, 0x80,
        0x80, 0x08,
158     0x00, 0x00, 0x03, 0x31, 0xE0, 0x00, 0x04, 0x00, 0x80, 0x04, 0x01, 0xFF, 0xC1,
        0x00, 0x08,
159     0x00, 0x00, 0x07, 0x1E, 0x00, 0x40, 0x00, 0x00, 0x00, 0x04, 0x00, 0x1F, 0xFA,
        0x00, 0x08,
160     0x00, 0x00, 0x07, 0xF0, 0x00, 0x40, 0x3B, 0x07, 0x60, 0x00, 0x00, 0x01, 0xFF,
        0x00, 0x00,
161     0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x34, 0x81, 0x90, 0xCC, 0xC0, 0x00, 0x3F,
        0xC0, 0x00,
162     0x00, 0x00, 0x0C, 0x00, 0x03, 0x30, 0x0C, 0x82, 0x90, 0x53, 0x20, 0x00, 0x07,
        0xF8, 0x00,
163     0x00, 0x00, 0x70, 0x40, 0x00, 0xC8, 0x3B, 0x02, 0x60, 0x53, 0x20, 0x00, 0x00,
        0xFE, 0x00,
164     0x00, 0x01, 0x80, 0x20, 0x01, 0xC8, 0x00, 0x00, 0x00, 0x4C, 0xC0, 0x00, 0x00,
        0x3F, 0x80,
165     0x00, 0x06, 0x00, 0x00, 0x03, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x07, 0xE0,
166     0x00, 0x08, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x01, 0xFC,
167     0x00, 0x30, 0x00, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x78,
168     0x00, 0x00, 0x40, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x10,
169     0x00, 0x00, 0xA0, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00,
170     0x00, 0x00, 0x44, 0x00, 0x00, 0x00, 0x02, 0x02, 0x30, 0xC0, 0x00, 0x00, 0x00,
        0x00, 0x00,
171     0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x03, 0x06, 0x30, 0xC0, 0x00, 0x00, 0x00,
        0x00, 0x00,
172     0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x01, 0x8C, 0x30, 0xC0, 0x00, 0x00, 0x00,
```

```
    0x00, 0x00,
    0x00, 0x00, 0x22, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x30, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x70, 0x19, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x20, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
])



# FrameBuffer for VUMeter
fb = framebuf.FrameBuffer(VUMeter, OLED_WIDTH, OLED_HEIGHT, framebuf.MONO_HLSB)

def display_menu():
    oled1.fill(0)
    oled1.text(" PLEASE SELECT ", 0, 0)
    oled1.text("1.Volume Control", 0, 20)
    oled1.text("2.VU Meter", 0, 40)
```

```
213         oled1.show()
214         oled2.fill(0)  # Clear the second display
215         oled2.show()
216 
217     def volume_control(volume):
218         oled2.fill(0)  # Clear the display
219         oled2.text('HEADPHONE', 28, 0)
220         bar_width = int((volume / 100) * OLED_WIDTH)
221         oled2.fill_rect(0, 30, bar_width, 10, 1)
222         oled2.text('Volume: {}%'.format(volume), 0, 20)
223         oled2.text("Press # to exit", 0, 50)
224         oled2.show()
225 
226     def increase_volume():
227         global volume_level
228         if volume_level < 100:
229             volume_level += 5
230             save_settings()  # Save the new volume level
231         volume_control(volume_level)
232 
233     def decrease_volume():
234         global volume_level
235         if volume_level > 0:
236             volume_level -= 5
237             save_settings()  # Save the new volume level
238         volume_control(volume_level)
239 
240 
241     def draw_vu_meter_LEFT():
242         x0 = hMeter
243         y0 = vMeter
244         r0 = rMeter
245         a1_Left = int(x0 + (sin(radians(MeterValueLeft)) * r0))  # meter needle horizontal
        coordinate
246         a2_Left = int(y0 - (cos(radians(MeterValueLeft)) * r0)) # meter needle vertical
        coordinate
247         oled1.blit(fb, 0, 0) # Draw vu meter background
248         oled1.line(x0, y0, a1_Left, a2_Left, 1)  # draws needle
249         oled1.show()
250 
251     def draw_vu_meter_RIGHT():
252         x0 = hMeter
253         y0 = vMeter
254         r0 = rMeter
255         a1_Right = int(x0 + (sin(radians(MeterValueRight)) * r0))  # meter needle horizontal
        coordinate
256         a2_Right = int(y0 - (cos(radians(MeterValueRight)) * r0)) # meter needle vertical
        coordinate
257         oled2.blit(fb, 0, 0)# Draw vu meter background
258         oled2.line(x0, y0, a1_Right, a2_Right, 1)  # draws needle
259         oled2.show()
260 
261 
262     def vu_meter(left_adc,right_adc):
263         global MeterValueLeft, MeterValueRight
264         # Sample window in milliseconds
265         sample_window = 2  # Adjust as necessary
266 
267         # Initial values for peaks and signal min/max
268         PeakLeft = 400
269         PeakRight = 400
270         fallTimeLeft = 0
271         falltimeright = 0
272         startMillis = time.ticks_ms()
273         SignalMaxLeft = 0
274         SignalMinLeft = 1024
275         SignalMaxRight = 0
276         SignalMinRight = 1024
277 
```

```
    while time.ticks_diff(time.ticks_ms(), startMillis) < sample_window:
        sampleLeft = left_adc.read_u16() / 64
        sampleRight = right_adc.read_u16() / 64

        if sampleLeft < 1024:
            if sampleLeft > SignalMaxLeft:
                SignalMaxLeft = sampleLeft
            if sampleLeft < SignalMinLeft:
                SignalMinLeft = sampleLeft

        if sampleRight < 1024:
             if sampleRight > SignalMaxRight:
                 SignalMaxRight = sampleRight
             if sampleRight < SignalMinRight:
                SignalMinRight = sampleRight

    PeaktoPeakLeft = SignalMaxLeft - SignalMinLeft
    PeaktoPeakRight = SignalMaxRight - SignalMinRight


    if PeaktoPeakLeft > PeakLeft:
        PeakLeft = PeaktoPeakLeft
    elif PeaktoPeakLeft < PeakLeft:
        fallTimeLeft = (PeakLeft - PeaktoPeakLeft) / 2
        PeakLeft -= fallTimeLeft

    if PeaktoPeakRight > PeakRight:
        PeakRight = PeaktoPeakRight
    elif PeaktoPeakRight < PeakRight:
        fallTimeRight = (PeakRight - PeaktoPeakRight) / 2
        PeakRight = PeakRight - fallTimeRight


    MeterValueLeft = PeakLeft / 8
    MeterValueLeft -= 60
    MeterValueRight = PeakRight / 8
    MeterValueRight = MeterValueRight - 60

    # Draw VU meter
    draw_vu_meter_LEFT()
    draw_vu_meter_RIGHT()


# Main loop
while True:
    if current_state == 'menu':
        display_menu()
    elif current_state == 'volume':
        volume_control(volume_level)
    elif current_state == 'vu_meter':
        vu_meter(left_adc,right_adc)
    utime.sleep_ms(150)
```