'''

ctplayer plays mp3 files

Audio files on DFPlayer

GP21 [p05] TX UART0 DfPlayer (RX [p2])

GP20 [p04] RX UART0 DFPlayer (TX [p3])

GP05 [p11] SDA I2C0 OLED [internally connected]

GP06 [p12] SCL I2C0 OLED [internally connected]

GP10 [p16] Button1 play/stop

GP09 [p15] Button2 rewind (REW)

GP08 [p14] Button3 fast forward (FF)

GP04 [p10] Button5 Vol +

GP03 [p09] Button6 Vol -

GP02 [P06] Battery Monitor

GP00 [P08] Tape head Switch

'''

# Hardware: ESP_C3_SuperMini, DFPlayerMini

# Micropython: https://micropython.org/download/LOLIN_C3_MINI/

# MicroPython v1.28.0 on 2026-04-06; LOLIN_C3_MINI with ESP32-C3FH4

# DFPlayerMini: https://github.com/redoxcode/micropython-dfplayer/blob/main/src/dfplayer/__init__.py

# _int_.py renamed to dfplayer.py

# SSD1306: https://github.com/Grrtzm/esp32-c3-0.42-oled-micropython/blob/main/oled/ssd1306oled042b.py

# Required libaries: dfplayer.py, ss1306oled042b.py

# Libraries saved on ESP32_C3 in /lib

import micropython

micropython.alloc_emergency_exception_buf(100)

```
import sys
import _thread
import time
from machine import I2C, Pin, ADC
from dfplayer import DFPlayer
from ssd1306oled042b import SSD1306

# initialize DFPlayer
df=DFPlayer(uart_id=0,tx_pin_id=21,rx_pin_id=20)

# setup Display I2C communication
i2c = I2C(0, sda=Pin(5), scl=Pin(6))
oled = SSD1306(i2c)
oled.flip()

# Enable and configure ADC with a range
adc=ADC(Pin(2))
adc.atten(ADC.ATTN_6DB) # 2V range
adc.width(ADC.WIDTH_12BIT)

# configure buttons
but01 = Pin(10, Pin.IN, Pin.PULL_UP) # play/stop
but02 = Pin(9, Pin.IN, Pin.PULL_UP)  # rew
but03 = Pin(8, Pin.IN, Pin.PULL_UP)  # ff
but05 = Pin(4, Pin.IN, Pin.PULL_UP)  # Vol +
but06 = Pin(3, Pin.IN, Pin.PULL_UP)  # Vol -

allfiles = 0 # sum of files
```

```
count = 0 # track count
folders = 0 # total folders
neg = 0
sdir = 1 # selected directory
sfiles = 0 # selected files
music = False
pin_interrupt = 0
audio = 15 # default audio level
PStoggle = False # Play[True]/Stop[False]
timeout = False # screen timeout

def Volume(level):
    global audio

    # decrement volume
    if level == -1:
        audio -= 1
        if audio < 0:
            audio = 0

    # increment volume
    if level == 1:
        audio += 1
        if audio > 30:
            audio = 30

    # ssd1306
    fm = str("%02d" % (audio))
```

```
    oled.fill_rect(0,30,63,30,0) # delete previous text
    oled.text("Vol "+fm[0:2],0,30,1)
    oled.show()

    df.volume(audio)
    time.sleep(0.2)

    return

def Track_Value(subd,file,mode):
    global count
    print('D: '+str(subd) + ' T: '+str(file) + ' #: '+str(count))

    # ssd1306
    fm = str("%02d%02d%02d" % (subd, file, count))
    oled.clear()
    oled.text('ctplayer',0,0,1)
    oled.text('D'+fm[0:2]+' T'+fm[2:-2],0,10,1)
    oled.fill_rect(0,20,63,30,0) # delete previous text
    if mode == 'REW':
        oled.text("REW <<",0,20,1)
    if mode == 'FF':
        oled.text("FF >>",0,20,1)
    oled.show()

    return

def Track_Select(subd,file):
```

```
    global count
    print('D: '+str(subd) + ' T: '+str(file) + ' #: '+str(count))

    # ssd1306
    fm = str("%02d%02d%02d" % (subd, file, count))
    oled.clear()
    oled.text('ctplayer',0,0,1)
    oled.text('D'+fm[0:2]+' T'+fm[2:-2],0,10,1)
    oled.text('Play> '+fm[4:],0,20,1)
    oled.show()

    read_bat()

    df.play(subd,file)
    time.sleep(0.5)

    Stop()

    return

def Play_range(start):
    global allfiles, sfiles, tlist, sdir, folders, count
    fl = start
    while ((fl < allfiles) and (music == True)):
        if allfiles > 0:
            fl = fl + 1
            if fl > allfiles:
                fl = 0
```

```
            if (sfiles == tlist[sdir]) and (sdir == folders):
                fl = allfiles
            else:
                if sfiles == tlist[sdir]:
                    sfiles = 1
                    sdir = sdir + 1
                    if sdir > folders:
                        sdir = folders
                else:
                    sfiles = sfiles + 1
            #print("count "+str(fl)+" "+str(start) +" " +str(allfiles))
            #print("sdir:"+str(sdir) +" sfiles:" + str(sfiles))
            if fl > count:
                Track_Select(sdir,sfiles)
                count = count + 1

    #count = allfiles
    return

def Play():
    global allfiles, count, neg, sdir, sfiles
    if allfiles > 0:
        if count >= allfiles:
            count = 0
            sdir = 1
            sfiles = 0
        if count >= 0:
            neg = 1
```

```
            print("Play > "+str(count))
            Play_range(count)
            time.sleep(0.25)
    else:
        print('No_Tracks')
        oled.text("No_Trks",0,20,1)
        oled.show()
        time.sleep(0.25)

    return

def Rewind():
    global allfiles, count, folders, neg, sdir, sfiles, tlist
    if neg == 1:
        if allfiles > 0:
            if count < 1:
                count = 0
            else:
                count = count - 1
                if (sfiles == 1) and (sdir == 1):
                    count = 0
                    neg = 0
                else:
                    sfiles = sfiles - 1
                    if sfiles < 1:
                        sdir = sdir - 1
                        sfiles = tlist[sdir]
                    if sdir < 1:
```

```
                sdir = folders
        if neg == 1:
            Track_Value(sdir,sfiles,'REW')

    return

def FastForward():
    global allfiles, count, folders, neg, sdir, sfiles, tlist
    if allfiles > 0:
        neg = 1
        count = count + 1
        if count > allfiles:
            count = allfiles
        else:
            if (sfiles == tlist[sdir]) and (sdir == folders):
                count = allfiles
            else:
                if sfiles == tlist[sdir]:
                    sfiles = 1
                    sdir = sdir + 1
                    if sdir > folders:
                        sdir = folders
                else:
                    sfiles = sfiles + 1

                if count < allfiles + 1:
                    Track_Value(sdir,sfiles,'FF')
                else:
```

```
                count = allfiles

    return

def Stop():
    df.stop()
    oled.fill_rect(0,20,63,30,0) # delete previous text
    oled.text("STOP <>",0,20,1)
    oled.show()

    return

def Hd_mode(pin):
    global music, pin_interrupt
    pin_interrupt = Play_mode.value()
    if Play_mode.value() == 0:
        music = True
    if Play_mode.value() == 1:
        music = False

    return

def Sw_mode(pin):
    global music
    if Stop_mode.value() == 0:
        music = False

    return
```

```
@micropython.native
def read_bat():
    Verr = 0.0035 # error correction voltage
    bat_max = 4.2 # charging cutoff voltage
    bat_nom = 3.7
    bat_min = 3   # discharge cutoff voltage
    R1 = 100000   # upper half of potential divider (pd)
    R2 = 43000    # lower half of pd
    # Errors can be reduced by using measured values for R1 & R2
    Rratio = (R2/(R1+R2)) # pd ratio
    Vpd_max = round(bat_max*Rratio,2) # 1.34V
    Vpd_min = round(bat_min*Rratio,2) # 0.96V
    # print (Vpd_min, Vpd_max)
    adcsum = 0
    # moving average filter
    for i in range(1,100):
        adcsum += adc.read_u16()
        time.sleep_ms(1)

    adcval = int(adcsum/100) # average bit value
    vf = round(((adcval / 65535.0 * 1.6) - Verr),2) # pd voltage
    Vbat = round(vf/Rratio,2) # batery voltage
    # print("ADC Val:",adcval,"Vcalc:",vf,"Vbat:",Vbat)
    Vpct = int(((vf - Vpd_min) / (Vpd_max - Vpd_min)) * 100) # %age charge
    # bat = str(Vbat)+"V"
    if (Vpct < 0):
        pcnt = "0%"
```

```
    elif (Vpct <= 100):
        pcnt = str(Vpct)+"%"
    else:
        pcnt = "over"

    # ssd1306
    oled.fill_rect(0,30,63,30,0) # delete previous text
    oled.text("Bat: "+pcnt,0,30,1)
    oled.show()

    return

def screen_timeout():
    global timeout
    try:
        while True:
            time.sleep(60)
            timeout = True
    except Exception as e:
        print("Thread Err",e)

    return

# assign pins to initialise IRQ and levels
Play_mode = Pin(0, Pin.IN, Pin.PULL_UP)
Play_mode.irq(trigger=Pin.IRQ_FALLING|Pin.IRQ_RISING, handler=Hd_mode)
Stop_mode = Pin(10, Pin.IN, Pin.PULL_UP)
Stop_mode.irq(trigger=Pin.IRQ_FALLING, handler=Sw_mode)
```

```
'''
start of the main process
'''

# count folders and files on SDcard
files = 0
tlist = [0] # initialize tlist and fill 1st element with 0
for i in range(1,99): # count folders (max 99) and files
 files = df.get_files_in_folder(i)
 if files > 0:
    tlist.insert(i,files)
    allfiles = allfiles + files
    # print('Folder:' +str(i) + ' File(s):'+str(files))
 else:
    # terminate on first empty folder
    break

# tlist[0] = total folders, tlist[n>0] = files in subfolder n
tlist[0] = len(tlist) - 1
folders = tlist[0]
print('Folder(s):'+str(folders) + ' File(s):'+str(allfiles))

df.volume(audio)
time.sleep(0.2)

# ssd1306
oled.clear()
```

```
oled.contrast(128)
oled.text('ctplayer',0,0,1)
if allfiles > 0:
    fm = str("%02d%02d" % (folders, allfiles))
    oled.text('D'+fm[0:2]+' T'+fm[2:],0,10,1)
    oled.show()
else:
    oled.text('No_Trks',0,10,1)
    oled.show()
    sys.exit()

timer = _thread.start_new_thread(screen_timeout, ())

try:
    while True:

        # play
        if (but01.value() == 0):
            PStoggle = not(PStoggle)
            if PStoggle == True:
                print("b1 Play > " + str(count))
                music = True
                Play()
                music = False
            else:
                Stop()
        else:
            if (pin_interrupt == 0):
```

```
        if (music == True):
            Play()
        #else:
            #Stop()
    else:
        Stop()

# decrement tracks
if (but02.value() == 0):
    print("b2 REW << " + str(count))
    Rewind()

# increment tracks
if (but03.value() == 0):
    print("b3 FF >> " +  str(count))
    FastForward()

# volume up
if (but05.value() == 0):
    print("b5 Vol+ <> " + str(audio))
    Volume(1)

# volume down
if (but06.value() == 0):
    print("b6 Vol- <> " + str(audio))
    Volume(-1)

time.sleep(0.25)
```

```
        # thread action
        if timeout == True:
            print("clear screen")
            oled.clear()
            oled.show()
            timeout = False

except KeyboardInterrupt:
    df.stop()
```