



GoDirect Python One Pager



テーマ

適正な温度でお茶を淹れよう

使用センサ：ワイヤレス温度センサ GDX-TMP

内容

お茶の種類によって適正な温度は次のように変わります。

- 100度 : 紅茶
- 90度 : ほうじ茶 玄米茶
- 70～80度 : 煎茶
- 50～60度 : 玉露



センサを使って各お茶の淹れ頃の温度になったら音(声)で知らせてくれるプログラムを作ってみましょう。



日常で使われている温度センサはどんなものがあるか

温度センサは非常に幅広い分野で活用されています。

エアコンやお風呂など、身近に使われている使われているものから、最近はIoTを活用した農業や施設の環境管理にも温度センサが活用されています。

例：お風呂の自動追い炊き機能等



例：冷蔵庫の温度調整



プログラムに工夫を入れてより便利に

- ・条件を満たしたら音を鳴らす(sounddevice等)または音声で知らせる(pytttsx3等)
↳ 作りたい機能によって使い分ける(今回は音声を使用)
- ・条件を満たしたら1回だけ鳴らし、設定したしきい値の間は重複して鳴らさない
- ・設定温度ごとに音を変える
- ・センサの測定間隔の調整

プログラム インストールが必要なライブラリ：numpy、pyttsx3

```
from gdx import gdx
import numpy as np
import pyttsx3

gdx = gdx.gdx()

# GDX-TMP xxxxxxxx をお使いのシリアル番号(本体に記載)を入れてください。
gdx.open(connection='ble', device_to_open="GDX-TMP 0F10E4A7")
gdx.select_sensors([1])
gdx.start(3000) # 3秒に1回測定

# 初期化
engine = pyttsx3.init()
engine.setProperty('rate', 130) # 読み上げスピードの設定

# 各温度範囲ごとの通知済みフラグ
notified_100 = False
notified_90 = False
notified_70 = False
notified_50 = False

while True:
    measurements = gdx.read()
    temp = measurements[0]
    if temp is None:
        break # データがなければループを抜ける
    print(f'{round(temp, 1)} °C')

    # 条件ごとに1回だけ通知するためのフラグをチェック
    if temp >= 99 and not notified_100:
        print(f'紅茶の温度になりました{round(temp, 1)} °C')
        engine.say('紅茶の温度になりました')
        notified_100 = True
    elif temp >= 90 and not notified_90:
        print(f'ほうじ茶・玄米茶の温度になりました{round(temp, 1)} °C')
        engine.say('ほうじ茶・玄米茶の温度になりました')
        notified_90 = True
    elif temp >= 70 and not notified_70:
        print(f'煎茶の温度になりました{round(temp, 1)} °C')
        engine.say('煎茶の温度になりました')
        notified_70 = True
    elif temp >= 50 and not notified_50:
        print(f'玉露の温度になりました{round(temp, 1)} °C')
        engine.say('玉露の温度になりました')
        notified_50 = True
    engine.runAndWait()

gdx.close()
```



https://www.rika.com/wp-content/uploads/2025/05/temp_opg.zip

テーマ

距離で音が変わる楽器を作ろう

使用センサ：ワイヤレスモーション(距離)センサ GDX-MD

内容

距離センサを使って「テルミン」のような楽器を作ってみましょう。

テルミンとは？

1920年に発明された、“ふれずに演奏できる”ことが特徴の世界最古の電子楽器で、アンテナからの手の位置で音高と音量を調節する楽器です。



プログラム インストールが必要なライブラリ：pygame

```
import os
import pygame
import time
from gdx import gdx

# GDX-MD 距離センサーを初期化
gdx = gdx.gdx()
# gdx.open(connection='ble', device_to_open='GDX-MD XXXXXX') # 特定のデバイスに接続する場合
gdx.open(connection='ble') # Bluetooth 接続
gdx.select_sensors([5]) # 距離センサーを選択
gdx.start(100) # データ取得開始 (100ms ごと)

# 音階ファイルのパス(ダウンロードした音階ファイル(mp3)の格納フォルダを指定)
SOUND_FOLDER = r"C:\Users\user\PycharmProjects\pythonProject\venv\Theremin-master\flute"

# 距離の範囲設定
MIN_DISTANCE = 0.1 # 最小距離 (メートル)
MAX_DISTANCE = 2.0 # 最大距離 (メートル)

def load_sounds(folder):
    """音階ファイルをロードして辞書に格納"""
    sounds = {}
    notes = ["C4", "D4", "E4", "F4", "G4", "A4", "B4", "C5"] # ド〜ド (1オクターブ)
    for note in notes:
        file_path = os.path.join(folder, note + ".mp3")
        if os.path.exists(file_path):
            sounds[note] = pygame.mixer.Sound(file_path)
    return sounds
```

裏面へつづく

```
def distance_to_note_index(distance):
    """距離を音階インデックスに変換"""
    if distance < MIN_DISTANCE or distance > MAX_DISTANCE:
        return None
    scaled_distance = (distance - MIN_DISTANCE) / (MAX_DISTANCE - MIN_DISTANCE)
    index = int(scaled_distance * 7) # 0~7のインデックス
    return index
```

```
def main():
    pygame.init() # pygame の初期化
    sounds = load_sounds(SOUND_FOLDER)
    notes = list(sounds.keys()) # 音階リスト
    current_note = None

    try:
        while True:
            measurements = gdx.read()
            if measurements is None:
                continue

            distance = measurements[0] # 距離 (メートル)
            print(f"Distance: {distance:.2f} m") # デバッグ用
            note_index = distance_to_note_index(distance)
```

```
            if note_index is None:
                # 範囲外の場合は音を停止
                if current_note is not None:
                    sounds[current_note].stop()
                    current_note = None
                continue
```

```
            new_note = notes[note_index]
            if new_note != current_note:
                # 新しい音階を再生
                if current_note is not None:
                    sounds[current_note].stop()
                sounds[new_note].play(-1) # ループ再生
                current_note = new_note
```

```
            time.sleep(0.1) # 適切な間隔で更新
```

```
except KeyboardInterrupt:
    print("終了します。")
```

```
finally:
    gdx.stop()
    gdx.close()
    pygame.quit()
```

```
if __name__ == "__main__":
    main()
```

プログラム



https://www.rika.com/wp-content/uploads/2025/05/distance_music.zip

音階(mp3)



<https://www.rika.com/wp-content/uploads/2025/05/mp3.zip>

テーマ

センサで測ったpH値がどんなものに相当するかを見てみよう

使用センサ：ワイヤレスpHセンサ GDX-PH

内容



センサで測ったpH値がどんなものと同じぐらいのpH値が表示されます。
また表にあるものを実際に測り、その通りの絵が表示されるかを確認してみましょう。

プログラム インストールが必要なライブラリ：tkinter、PIL

```
import tkinter as tk
from PIL import Image, ImageTk
from gdx import gdx
import time
```

```
# 画像ディレクトリ(※自身の画像ディレクトリに変更してください)
image_directory = r"C:\Users\user\PycharmProjects\pythonProject\venv\iloveimg-converted"
```

```
# pHに対応する画像ファイルとラベル(※画像は別途ご用意ください)
```

```
ph_images = {
    14: ("水酸化ナトリウムに相当", "suisankanatoriumu.jpg"),
    13: ("漂白剤に相当", "hyouhakuzai.jpg"),
    12: ("アンモニアに相当", "anmonia.jpg"),
    11: ("洗濯洗剤に相当", "sentakusenzai.jpg"),
    10: ("石鹸に相当", "soap.jpg"),
    9: ("重曹に相当", "juusou.jpg"),
    8: ("海水に相当", "umi.jpg"),
    7: ("水に相当", "water.jpg"),
    6: ("食器洗剤に相当", "syokkisenzai.jpg"),
    5: ("コーヒーに相当", "coffee.jpg"),
    4: ("トマトに相当", "tomato.jpg"),
    3: ("オレンジに相当", "orange.jpg"),
    2: ("酢に相当", "vinegar.jpg"),
    1: ("胃酸に相当", "isan.jpg"),
    0: ("塩酸に相当", "ensan.jpg")
}
```

```
# 初期化
```

```
gdx = gdx.gdx()
gdx.open(connection='usb', device_to_open="GDX-EA 065026H0")
gdx.select_sensors([2])
gdx.start(1000)
```

```

# tkinterウィンドウ設定
root = tk.Tk()
root.title("pHセンサー")
root.geometry("450x500")
# 各種ラベル
image_label = tk.Label(root) #イメージ
image_label.pack()
countdown_label = tk.Label(root, text="", font=("Helvetica", 16)) #カウントダウン
countdown_label.pack(pady=10)
text_label = tk.Label(root, text="", font=("Helvetica", 16)) #結果表示
text_label.pack(pady=10)

def update_image(ph_value): #pH値に応じて画像とテキストを更新
    rounded_ph = round(ph_value, 2) # 小数点第2位で四捨五入
    for ph_level, (description, filename) in ph_images.items():
        if rounded_ph >= ph_level:
            text_label.config(text=f"測定値: {rounded_ph} | pH{ph_level}: {description}")
            image_path = f"{image_directory}/{filename}"
            img = Image.open(image_path)
            img = img.resize((300, 300)) # 画像サイズ調整
            img_tk = ImageTk.PhotoImage(img)
            image_label.config(image=img_tk)
            image_label.image = img_tk # 参照を保持する必要がある
            break
    countdown_label.config(text="") # 測定後にカウントダウンの表示を消す

def perform_measurement(): #測定を実行
    countdown(10) # 10秒間のカウントダウン

def countdown(seconds): #カウントダウン表示
    if seconds >= 0:
        countdown_label.config(text=f"測定中...残り {seconds}秒")
        root.after(1000, countdown, seconds - 1) # 1秒後に再度呼び出し
    else:
        ph = read_sensor()
        if ph is not None:
            update_image(ph)

def read_sensor(): #センサーからデータを取得して表示
    measurements = gdx.read()
    if measurements:
        return measurements[0] # pH値を返す
    return None

# 測定開始ボタン
start_button = tk.Button(root, text="測定開始", command=perform_measurement, font=("Helvetica", 16))
start_button.pack(pady=20)

# tkinterメインループ
root.mainloop()

gdx.stop()
gdx.close()

```



https://www.rika.com/wp-content/uploads/2025/05/gdx_ph.zip