

ミクロなエリアにおける人流状況分析Webアプリの開発

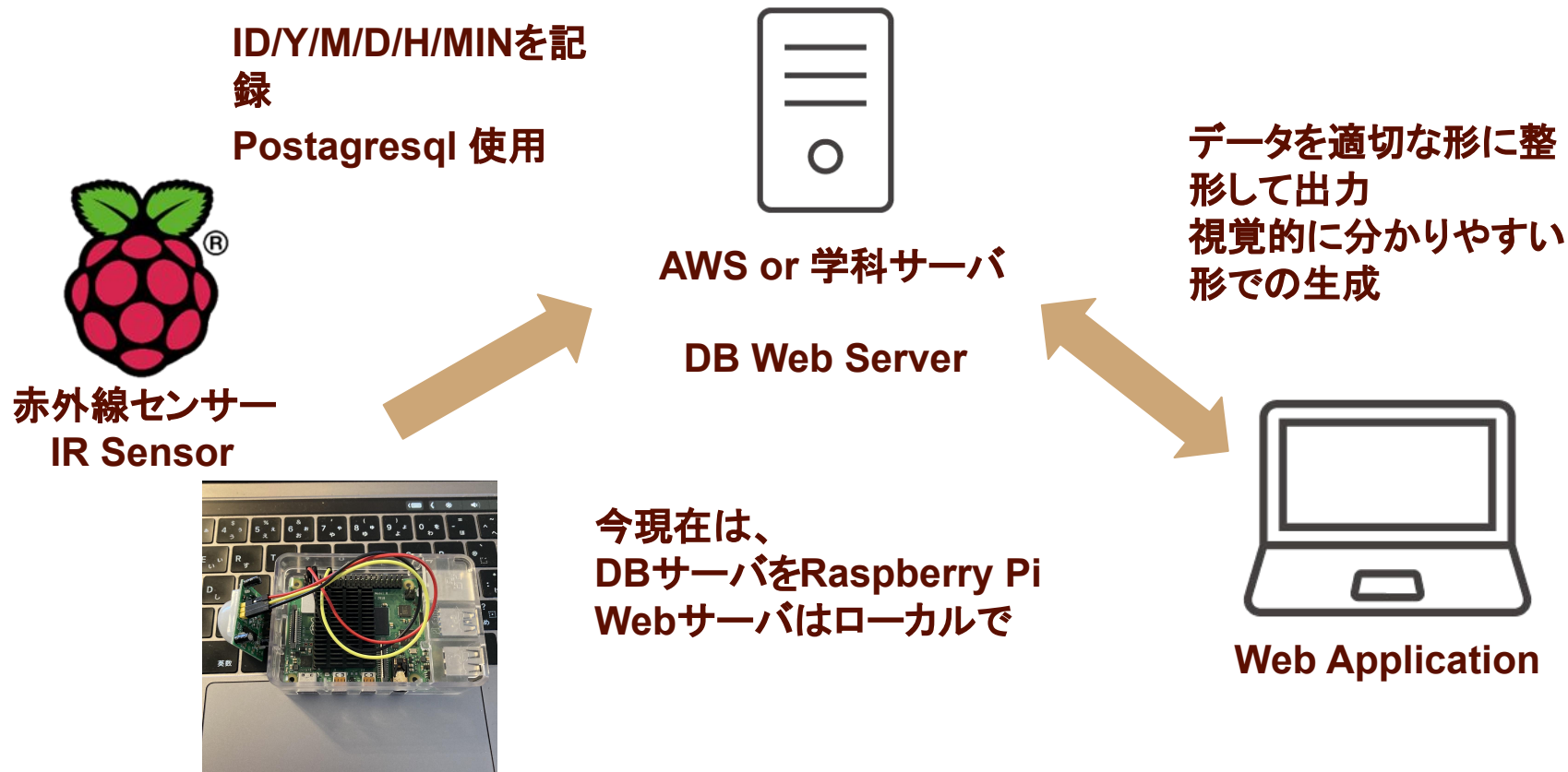
Development of a congestion analysis system for a small area

概要

部屋や通路など、建造物内の各エリアに人感センサーを設置し、その反応時間を記録、保存するシステムを作成する。

またそのデータを用いて、各エリア、時間ごとの反応回数に関する分析、その視覚化を出力するWebアプリの開発を行う。

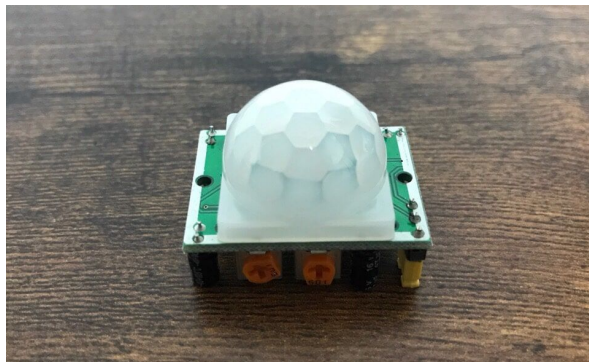
想定されるシステムの流れ



使用している機材



Raspberry Pi3B+
(以前使ったので)



赤外線センサ
IR Sensor
([アマゾンリンク](#))



ジャンパーワイヤ
([アマゾンリンク](#))

キーボードやディスプレイは除外(安定した接続のためにもしかしたらLANケーブル)

現況

年：2021

月：5

日：16

```
{0: [0, 0, 0, 0], 1: [0, 0, 0, 0], 2: [0, 0, 0, 0], 3: [0, 0, 0, 0], 4: [0, 0, 0, 0], 5:
: [0, 0, 0, 0], 6: [0, 0, 0, 0], 7: [0, 0, 0, 0], 8: [0, 0, 0, 0], 9: [0, 0, 0, 0], 10:
[0, 0, 0, 0], 11: [0, 0, 0, 0], 12: [0, 0, 0, 0], 13: [0, 0, 0, 0], 14: [0, 0, 0, 0],
15: [0, 0, 0, 0], 16: [0, 0, 0, 0], 17: [0, 0, 0, 0], 18: [0, 2, 19, 81], 19: [9, 0, 0,
0], 20: [0, 0, 0, 0], 21: [0, 0, 0, 0], 22: [0, 0, 0, 0], 23: [0, 0, 0, 0]}
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 19, 81, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

{時間:4クォータ}の辞書と、リスト化

- ・赤外線センサの動作スクリプト作成
- ・赤外線センサからのデータをDBサーバへ書き込み
- ・データを適切な形へ整形し出力 (15分区切りでカウント etc...)

現況

```
pi@raspi1: ~/Desktop/test $ python3 test_send.py
3 / 2021 / 05 / 16 / 18 / 42
None
4 / 2021 / 05 / 16 / 18 / 43
5 / 2021 / 05 / 16 / 18 / 43
None
6 / 2021 / 05 / 16 / 18 / 43
7 / 2021 / 05 / 16 / 18 / 43
8 / 2021 / 05 / 16 / 18 / 43
None
None
None
None
None
9 / 2021 / 05 / 16 / 18 / 43
None
10 / 2021 / 05 / 16 / 18 / 43
None
None
None
11 / 2021 / 05 / 16 / 18 / 43
None
None
None
None
12 / 2021 / 05 / 16 / 18 / 43
N
```

i d	y	m	d	h	mi n
0	9999	99	99	99	99
1	2021	5	16	18	23
2	2021	5	16	18	23
3	2021	5	16	18	42
4	2021	5	16	18	43
5	2021	5	16	18	43
6	2021	5	16	18	43
7	2021	5	16	18	43
8	2021	5	16	18	43
9	2021	5	16	18	43
10	2021	5	16	18	43
11	2021	5	16	18	43
12	2021	5	16	18	43
13	2021	5	16	18	44
14	2021	5	16	18	44
15	2021	5	16	18	44
16	2021	5	16	18	44
17	2021	5	16	18	44
18	2021	5	16	18	44
19	2021	5	16	18	44
20	2021	5	16	18	44
21	2021	5	16	18	44
22	2021	5	16	18	45
23	2021	5	16	18	45

現況

DBへリモートで接続する為に
libpq5かlibpq-devを事前に入れる必要がある

```
from datetime import datetime
import time
import RPi.GPIO as GPIO
import psycpg2

conn = psycpg2.connect(\
    "host=ホスト名 \
    port=ポート番号(pgsqlなら5432とか) \
    dbname=db名 \
    user=ユーザ名(権限周り確認) \
    password=パス" \
)

#DBレコード最後尾のIDを取得
cur = conn.cursor()
cur.execute("SELECT id from test_table order by id desc limit 1")
before_id = cur.fetchall()[0][0]

#インターバルだったりスリープ時間を記録
#使用するGPIOピンを選択
INTERVAL = 1
SLEEPTIME = 2
GPIO_PIN = 18

#GPIOのセットアップ
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN, GPIO.IN)
```

```
if __name__ == '__main__':
    try:
        while True:
            # センサー感知
            if(GPIO.input(GPIO_PIN) == GPIO.HIGH):
                before_id += 1
                y = datetime.now().strftime("%Y")
                m = datetime.now().strftime("%m")
                d = datetime.now().strftime("%d")
                h = datetime.now().strftime("%H")
                min = datetime.now().strftime("%M")
                print(before_id, "/", y, "/", m, "/", d, "/", h, "/", min)
                #id/y/m/d/h/minをinsert
                cur.execute("INSERT INTO test_table \
                    VALUES (" + \
                        str(before_id) + "," + \
                        str(y) + "," + \
                        str(m) + "," + \
                        str(d) + "," + \
                        str(h) + "," + \
                        str(min) + ")")
                #反応後即動作だと反応感知が重複するので適切なスリープを挟む
                time.sleep(SLEEPTIME)
            else:
                print("None")
                time.sleep(INTERVAL)
        except KeyboardInterrupt:
            #Ctrl + C => 終了
            print("finish")
        finally:
            #GPIO周りをリセット
            GPIO.cleanup()

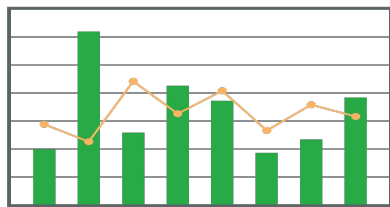
    cur.close()
    conn.commit()
    conn.close()
```

Webアプリ環境

フレームワーク	Flask Python3
サーバ	AWS Linux AMI t2 micro or 学科サーバ
サーバソフトウェア	Apache
フロントエンド	HTML5/CSS/Javascript
DB管理	Postgresql

グラフ化

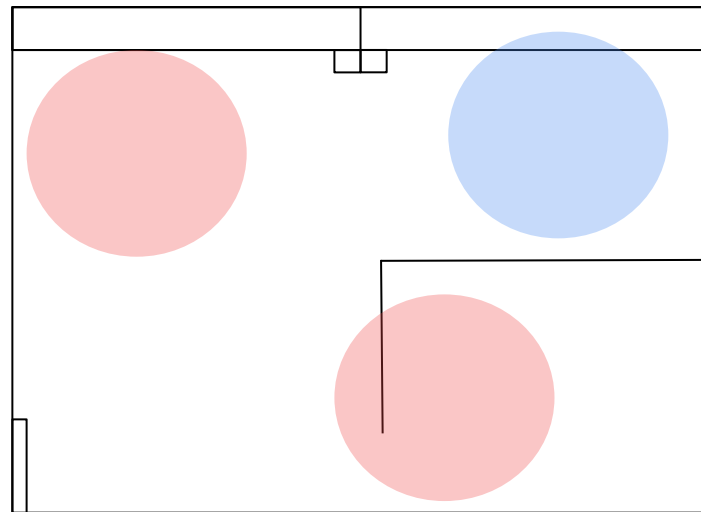
- ・15分毎の反応回数をグラフ化
- ・~~各エリアでグラフを分割~~
- ・頻度が高い/低い等の時間を出力
- ・分析内容の出力
(反応回数の比較等)



~~~~~  
~~~~~  
~~~~~

## マップ化

- ・各時間の頻度をヒートマップ化
- ・余裕があれば時間をスライダーで操作



# 開発スケジュール

1. Raspberry Pi/センサーの工作、セットアップ
2. センサーの動作確認
3. 出力内容の調整とRaspberry Piのサーバ接続確認
4. DBへの書き込みテスト
5. 実際の環境に設置して動作テスト
6. 複数の設置によるDB書き込みテスト、Webアプリ開発 ← 現況
7. Webアプリ開発、実際の使用によるフィードバック収集