NFCタグを用いた出席管理・入退室記録システムの開発と運用

DEVELOPMENT AND OPERATION OF THE ATTENDANCE AND ENTRY/EXIT RECORDING SYSTEM BY USING THE NFC TAG TECHNOLOGY

相良　二朗　　芸術工学部プロダクト・インテリアデザイン学科　教授

Jiro SAGARA　　Department of Product & Interior Design, School of Arts and Design, Professor

要旨

コロナ禍の発生に伴い、接触機会の減少が感染症対策として求められた。プロダクト・インテリアデザイン学科では、出席を、出席カードを授業開始時に学生へ配布し、学生が手描きしたカードを授業の終わりに教員が回収し、エクセルなどの表計算ソフトへ入力するという方法で行っていたが、接触機会が多いことと、工数が多いことから、これを機に合理化することとした。

NFC（近距離電界通信）という微弱な電波を受信することで発電し、記憶された情報を電送する技術は、公共交通機関のICカードなどで日常使用されているが、シール形状のNFCタグが安価に入手可能になったことから、学生証へ貼付し、教室入り口に専用リーダーを設置した。リーダー部分に近づけるだけで、時刻と学籍番号、氏名を蓄積し、授業終了後に担当教員へメール送信する構成とした。

コロナの感染拡大にともない、他教室での授業やサテライト授業となったため、可搬型のリーダーも開発した。さらに、工房の出入りやスタジオ自習利用の入退室管理システムを開発した。

本稿では開発したシステムの概要と使用する中で生じた問題点などを記し、今後の活用の資源とすることを目的としている。

Summary

Since 2020, reduction of touch opportunity becomes one of the new standards. We, the faculty member of the Dept. of Product & Interior design, adopted small paper to get attendance of students for each class. It requires a lot of touch opportunity for both faculty member and students. Adding this, it was time wasting process to just record attendance.

So, I developed automated record system by using NFC; Near Filed Communication technology, which is very familiar for us such as IC card of public transportation system. Thin seal of NFC tag which holds the students ID number and name was delivered to each student in April. Students advised to stick it to ones ID card, then touch it to the reader in the classroom. All data are stored in the reader and e-mailed to assigned faculty staff automatically, as attached csv format.

I developed wall mount type at first for 6101 lecture room, according to the advance of the COVID-19, two portable devices for another class rooms were developed. Furthermore, two types of the entry/exit recording system were developed for self-study students at laboratory building and studios.

The outline of the development and some problems in the operation are described in this paper.

1. はじめに

NFC タグは Near Field Communication（近距離電界通信）と呼ばれる技術で、1970 年代に開発された RFID: Radio Frequency Identification の発展形である。RFID は電磁波をコイルで受け、発生する起電力にて保有しているデータを送信するもので（図 1）、初期の RFID は書き換えできない 64bit の固有の ID を送信する機能しかなく、直径 4mm 弱のガラス管に封入されていた。現在でもペットの皮下に埋め込み、個体識別に利用されている。筆者は 1993 年に入院患者の無断外出（徘徊）を検知するシステムへの応用を試みたことがあるが、床マット下に設置したアンテナと靴に仕込んだ RFID では通過時間内に確実な通信が行えず、断念した経験がある [1)]。

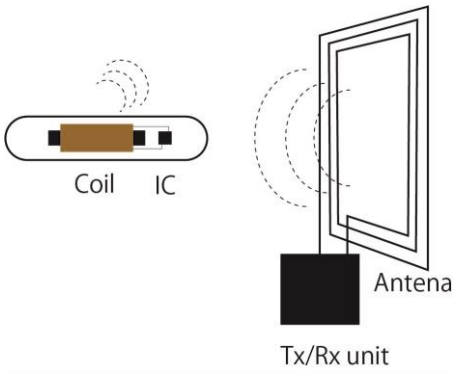

図 1. RFID の原理

NFC は公共交通機関の非接触 IC カードの通信規格からスマートフォンやウエアラブルデバイスなどの機器間データ交換などに利用されており、国際標準規格が定められている。開発当初は各社が独自の方式を提案していたが、NFC フォーラムが設立され、SONY の FeliCa を包含するユニバーサルな規格となった [2)]。

現在主流の NFC タグは、EEPROM を搭載しデータの書き換えが可能となり、形状も大幅に小型化し、プラスチックカードへの封入やシールとしても利用可能となった。

2. 開発に至る経緯

2019 年夏に参加した国際会議(i-CREATe-2019)において、NFC タグを応用したデバイスの発表に触れ、小規模な応用もコスト的にも技術的にも可能であることを知った。プロダクト・インテリアデザイン学科では、講義科目の出席確認において、出席カードの配布、記入、回収、データ入力と多くの工数を要していたため、合理化のためにマークシートの利用を試行していたが、用紙のコスト負担やエラー対応などの問題があった。このため、NFC タグを利用した出席確認システムの構築に 2019 年冬に着手した。2020 年度は新型コロナ感染症の蔓延とともに始まり、接触の回避が手指の消毒、換気、マスクの着用と共に重要な感染対策となった。従来の出席カードやマークシートでは接触を回避することができないため、NFC タグを用いた出席確認システムの必要性が更に高まった。

NFC タグを用いた入退室管理は、セキュリティ対策として企業では導入が進められているが、社員証の更新やリーダーの設置およびシステム構築に多額の初期投資が必要であり、現実的ではない。

3. 開発した装置の概要

実習・演習授業は小人数で実施されるため、講義科目を中心とした。講義は 6101 教室で行われるため、6101 教室の入り口付近の壁面に壁掛け型リーダーを設置することにした。また、新型コロナ感染症対策上、より大きな教室での講義や、6102 スタジオをサテライト会場とした講義となったため、可搬型リーダーも開発した。図 2 に壁掛け型リーダーの、図 3 に可搬型リーダーの外観を示す。

3.1. ハードウェア構成

図 2. 壁掛け型 NFC タグリーダー

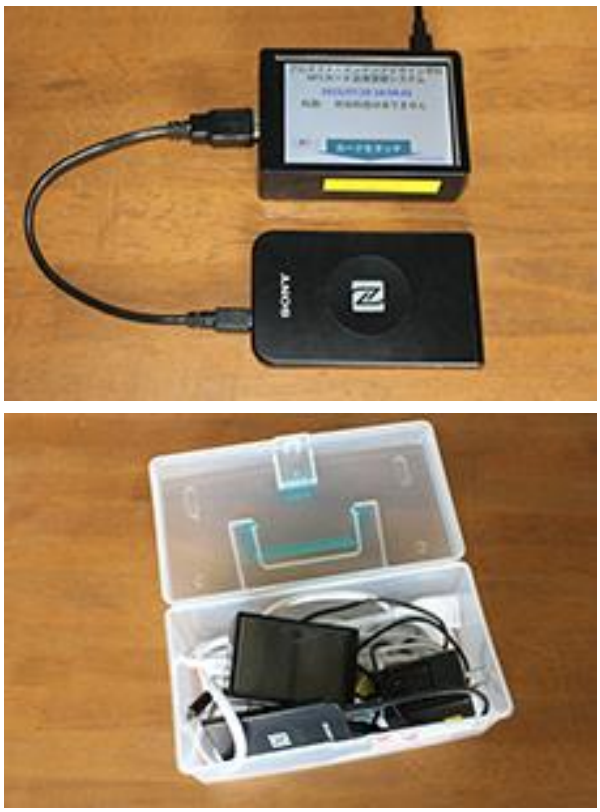

図 3.　可搬型 NFC タグリーダー

上：使用時

下：収納時

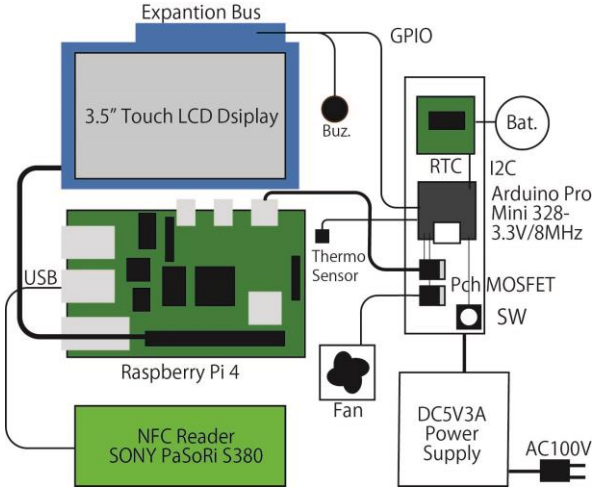

図 4.　壁掛け型 NFC タグリーダーの構成

図 5.　NFC タグ（右は他学科生用）

いずれも Raspberry Pi を制御用マイクロコンピュータとし、3.5 インチ LCD タッチパネルを表示装置とした。NFC タグリーダーは SONY 製 PaSoRi-S380 を採用した。壁掛け型では、常時通電状態となるため、午後 9 時に電源を落とし、午前 8 時に起動するよう、8bit の SOC マイクロコントローラ（adafruit 社製 Arduino Pro Mini 328-3.3V/8MHz）に RTC（リアルタイムクロック）モジュールを組み合せ、MOSFET で Raspberry Pi の電源を制御することとした。メンテナンス時に安全に Raspberry Pi をシャットダウンさせるために、裏面から操作できる場所に押し釦スイッチを設け、マイクロコントローラを介して GPIO ピンを変化させるようにした。図 4 に壁掛け型の構成図を示す。

可搬型は、図 3 上図に示すように Raspberry Pi4 に 3.5 インチ LCD タッチパネルを組み合わせたもので、市販されている専用のプラスチック製ケースに収めた。

NFC タグは、144Bytes の EEPROM を搭載した NTAG213 チップと薄膜アンテナをシート状にしたもので、粘着テープで貼り付けることができる。学科学生には学生証に貼付するように指導し、他学科生にはアクリル板に貼付したものを貸し出すようにした（図 5）。NFC タグには学籍番号と氏名（アルファベット表記）データを記憶させ、フリーウェアのリレーショナルデータベースシステムである MariaDB にてデータを蓄積し、設定した授業終了時刻にその授業コマ内のデータを、設定している授業担当者へ CSV 形式のデータを添付したメールを送信することにした。

3.2.　ソフトウェア構成

制御ソフトウェアは、生産性の高さから Python を採用し、GUI は Python のライブラリーである Tkinter にて

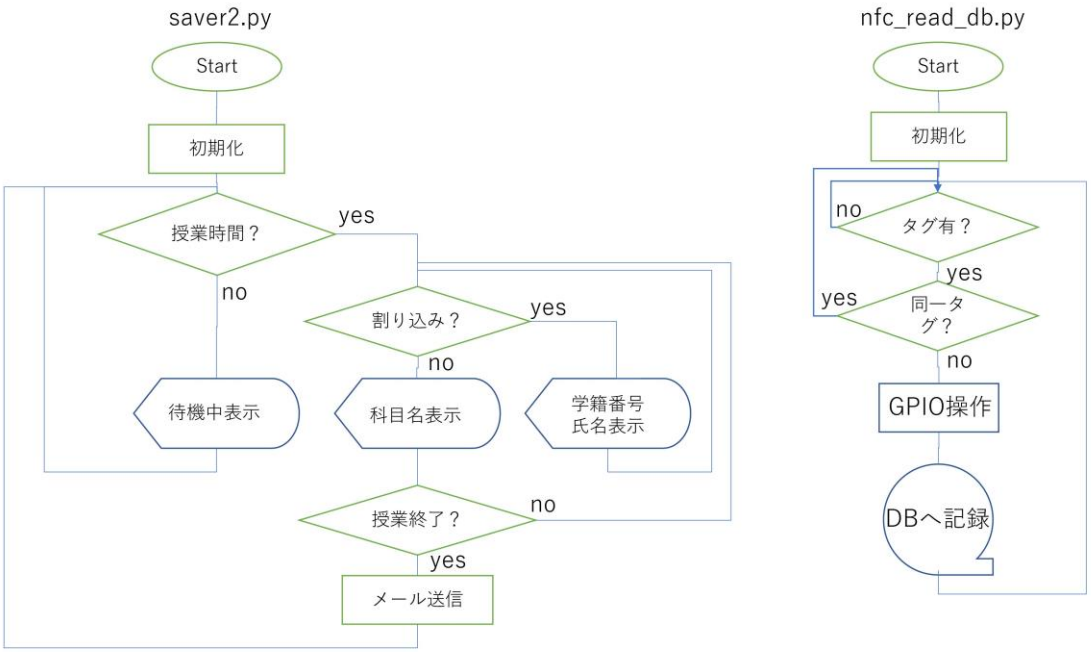

図 6.　NFC リーダーの主たるルーチンのフローチャート

構築した。タグを読み取り表示を行うソフトウェアの構造は図 6 に示すように、NFC タグの読み取りとデータベースへの記録を行うルーチンと、読み取った NFC タグの情報を表示し、授業終了時にデータをメール送信するルーチンに分けて構成した。

画面は二つのモードとした。一つは授業の間の NFC タグを受け付けないモードで、暗い画面上に現在時刻と Out of Service Now.の文字を表示する。もう一つは入力画面で、授業開始 10 分前（後に 3 限は 12:10 からに修正した）に切り替わり、図 7 上に示す画面が表示される。NFC タグがリーダー部分に近接すると、図 7 下のように画面中央に学籍番号と氏名が約 1 秒間表示し、読み取りと同時に短い電子音を鳴らす。

NFC タグ情報の読み出しおよび書込みには GitHub 上に公開されている Python library の nfcpy[3),4)]を利用した。開発当初は Python2.7 にのみ対応していたが、利用開始後に Python 3.8 対応版が提供されたので切り替えた。読み取りアプリケーションは、nfcpy が Python2.7 対応のみだったため[A)]、システム常駐で動作し、NFC タグを読み取ると MariaDB の Students Table にデータを書き込み、GPIO を介して割り込みをかけるようにした。

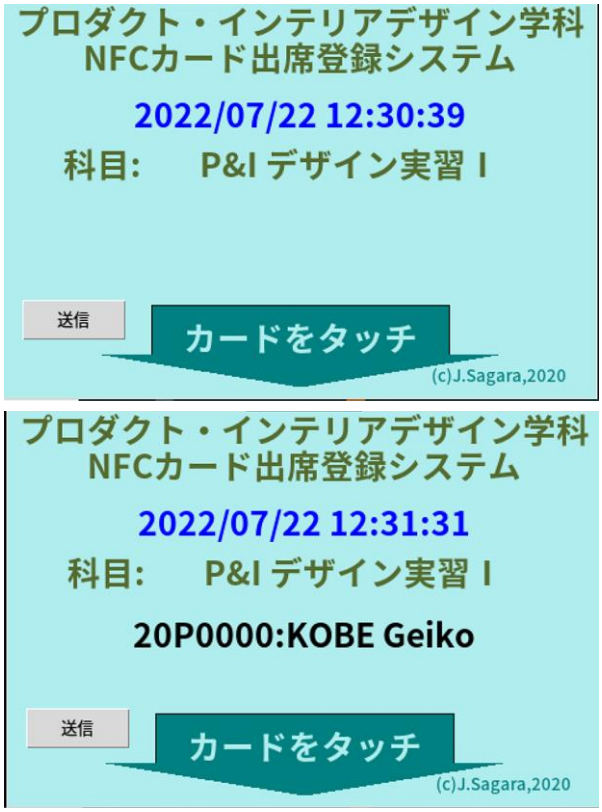

図 7.　授業時間中の画面（上）と NFC タグをタッチしたときの画面（下）

NFC カードへのデータ書き込みは、独立したアプリケーションとして Python で開発した。1 枚ずつ書込むもの

と、学籍番号と氏名のCSVファイルから連続して書き込むものの2種類を用意した。このCSVファイルはKDUポータル上の履修者名簿からダウンロードできるが、アルファベット氏名の姓と名の間が全角空白文字となっており、そのままでは書き込めなかったため、全角空白文字を半角空白文字に変換するプログラムを別途作成した。

授業終了10分前（設定可能）になると、その授業内での出席者データをCSV形式の添付ファイルとして、データベースで設定している担当教員のメールアドレスに宛ててメール送信する。また授業時間中でも画面上の「送信」ボタンに触れるとその時刻までのデータを同様にメール送信する。

メールは当初Gmailを利用していたが、本学のネットワーク環境の変化に伴い、学科専用の出席確認用アカウントを発行してもらい、office365mailを利用することにした。Python初心者であるため、稚拙なコードではあるが主たるソースコードである、nfc_read_db.py[a]、saver2.py[b]、askDB.py[c]、office365mail.py[d]を文末に示す。

3.3. 電源管理サブシステム

壁掛け型リーダーに搭載したArduino Pro Mini 328のプログラムはArduino SDKを用いてC++で記述した。Raspberry PiとはGPIOで接続し、毎12:30:00にRaspberry PiからのGPIO信号でRTCの時計を合わせるようにした。また、温度センサーを接続し、Raspberry Piの温度上昇に応じて冷却ファンを制御するようにした[B]。

4. スタジオ等自習利用の入退室記録システム

新型コロナ感染症の脅威は継続しており、スタジオ等の自習利用者の記録も必要となった。用紙に学籍番号、氏名、入室時刻、退室時刻を記入し、事務職員がコンピュータに入力する方法で行われていたが、接触回避の観点からはNFCタグの利用が望ましい。授業の出席の場合は、教室に入った時点での記録のみで済むが、入室と退室を分けて記録する必要があった。92号棟（プロダクト工房）の利用は滞在時間が比較的長くなるため、二つの時刻の

図8　92号棟入り口

図9　6号棟1階スタジオ入り口

図10　6号棟2階スタジオ入り口

間を利用時間と判断できる。このため、壁掛け型を修正し、常時受け付けるようにした装置を作成し正面入り口に設

置した（図 8）。スタジオ等では頻繁な出入りが予想されたため、5 インチの LCD タッチパネルを採用し、画面上で入室と退室を選択するようにした。また、設置台数を少なくするため、1 台で複数のスタジオに対応できるようにした。利用者はスタジオを選び、次いで入室か退室を選ぶ。

この装置は 3D プリンタで筐体を制作し、1 階スタジオ入り口のスチールサッシ（図 9）と、2 階スタジオ入り口の壁面（図 10）にそれぞれ設置した。

4.1.　ハードウェア構成

世界的な半導体不足の影響から Raspberry Pi の購入も不可能となったため、ストックしていた Raspberry Pi 3b+を利用した。電源管理サブシステムは省略し、深夜に再起動するようにし、冷却ファンは OS の機能を利用するようにした[B)]。AC アダプタを電源としたため、昇圧回路を加えて 5.2V を供給するようにした。NFC タグリーダーは小型且つ安価である NXP Semiconductors 製 PN532[5)]を利用した。

4.2.　ソフトウェア構成

壁掛け型を基本とし、GUI 部分のみを改変した。待ち受け画面を図 11 に、教室選択後の入退室選択画面を図 12 に示す。入室または退室が選択され、NFC タグが近接すると学籍番号と氏名とともに、入室時には「おはようございます」を退室時には「おつかれさまでした」を 1 秒間表示し、NFC タグの接触を促す（図 13）。

データベースのテーブルは年月日、学籍番号、氏名、時刻、スタジオ名、入室／退室をコラムとし、午後 9 時になると、その日のすべてのデータを学科主任、学科事務員および筆者宛に送信する。同様に主たるソースコードである entryExit2.py[e)]を文末に示す。

5.　結果と考察

2020 年度は講義科目履修者が多い 1 年生と 2 年生全員に NFC タグを配布し、3 年生と 4 年生は希望者にのみ配布した。2021 年度、2022 年度と新入生に配布したため、2022 年度は全員が NFC タグを所持することになった。毎年、数名が NFC タグの不具合を訴え、新規の発行が必要となる。多くは洗濯をした、硬いものに擦れたなどで、アンテナ部分の断線が原因のようであるが、粘着テープのシートを貼付しただけなので、想定よりも少ない事故発生数である。NFC タグの価格は 100 枚入りを購入すると単価が 40 円弱であり、マークシートを 1 科目 1 期で一人当たり 15 枚使用するよりも安価となり、且つエクセルシートに読み込むだけであるため、大幅な業務改善となった。

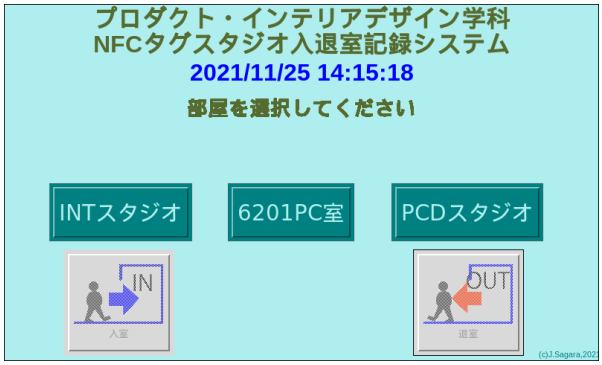

図 11.　待ち受け画面

図 12.　部屋選択後の画面

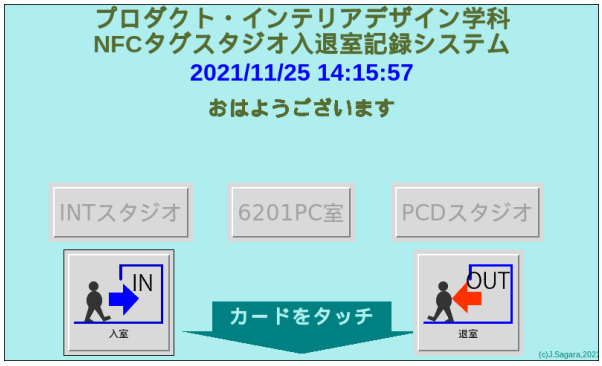

図 13.　出入り選択後の NFC タグ接触待ち

可搬型では、Raspberry Pi 起動後 Wi-Fi に接続し、時

計が合った後にリーダーアプリケーションを起動しないと正しい授業科目が表示されない。RTC を搭載すれば立ち上がりと同時に使用可能となるが、タッチパネル装着で GPIO の殆どが専有されているうえ、冷却ファンと圧電ブザーを搭載しているため、スペースが確保できない。Raspberry Pi 本体への RTC 搭載が待たれる。3 限目の授業の際に、学生が早めに教室に入り、12：50 の出席受付開始時刻を待たずに利用することがあったため、3 限目については 12：10 から登録が可能となるように変更した。また、2022 年度はコロナ禍も落ち着きを見せ、サテライト教室の運用が不要となったことから、携帯型で教室アイコンによる起動としていた仕様を変更し 6101 教室以外での授業を自動起動するようにした。これは、可搬型の 1 台のタッチパネルが不調となったことへの対策ともなった。

壁掛け型は、学生に対して内部構造を見せることによる教育効果も考えてスケルトンとし、NFC タグリーダーが見えるようにしていた。しかし、導入当初には一部の学生が LCD 画面に NFC タグを触れていた。

データ送信の電子メールは、前述したように当初は Google 社の Gmail を用いていたが、2020 年秋に学内のメールシステムが Office365 へ変更となったため、学科専用出席確認用メールアドレスを発行してもらい、office365 メールシステムを利用することとした。office365 メールシステムは python3.8 以上にしか対応していなかったため、gmail.py のみを python3.8 に対応させた office365mail.py に書き換えた。Gmail は 2022 年 6 月に大幅な仕様変更を行い、従来のユーザーアカウントとパスワードによる認証から、事前に web 上で認証を受け、特殊な token を必要とするようになった [6]。この対応は別のプロジェクトで余儀なくされ、原因究明を含めて 20 日間以上の作業が必要であった。この時点で office365 メールシステムに切り替えできたことは幸運と言える。もっとも、全てのデータは MariDB 上に保管されているため、簡単な命令で抽出することができ、トラブルが生じても出席データ自体が失われることはほとんどない。

時間割は、前期後期の別、時限、科目名、教員名、教員メールアドレスをデータベース上に記録しているが、データベーステーブルの書き替えは MariDB のコマンドに習熟していないとできない点は改善の余地がある。

本システムは教員側の合理化を主目的としたが、NFC タグによる出席に対して、学生に簡単なアンケート調査を行った。267 名中 63 名(23.6%)が回答した。41 名(65.1%)は NFC タグが良いと回答し、4 名は出席カードが良いと、17 名がどちらも同じと回答した。NFC タグを忘れたことがある学生は 18 名(28.6%)で内 1 名は 5 回以上と答えた。NFC タグの読み取りエラーは 43 名(68.3%)が無いと回答し、20 名が有ると回答した。自由記述の中には、出席が取れたのかどうかが不安であるとした学生が居た。この点は確認メールを送信するなど今後改善していきたい。

6. 今後の展開

個人認証技術は ICT の進展とともに急速に進んでおり、マスクを装着したままでの顔認証も可能となっている。しかし、このような先端技術の導入には高額な費用が必要であり、学生証の IC カード化も相当の初期投資が必要となる。学生証には磁気カードが埋め込まれているが、磁気カードリーダーは物理的な接触を必要とするため定期的なメンテナンスが必要である上、現在では磁気カードリーダーの購入も難しくなっている。クレジットカードや交通系 IC カード、スマートフォンへも NFC が利用されており、これらに記録されている個人情報から個人を特定することも可能ではあるが、個人情報を読み取られるかも知れないという不安をぬぐうことができない。やはり、専用の NFC タグの利用が望ましい。

開発したシステムは 2 年を経過する中で安定状態となり、普段の利用には手がかからなくなっている。しかし、年度当初には新入生への NFC タグ作成と配布、時間割の変更に伴うデータベースの更新、破損 NFC タグの再発行といった作業が必要となる。筆者は 2023 年度末での退職となるため、これらの作業を誰でもできるようにする対話型アプリケーションの開発が、このシステムの持続使用には必要と感じている。

もちろん、全学的に学生証の IC カード化と各教室への

リーダー設置が行われることが最善である。

参考文献

1) 相良二朗・赤澤康史・坊岡正之 他、「微弱電波を用いた徘徊者看視システムの開発-第2報」、『福祉のまちづくり工学研究所報告集平成7年度』、1996、pp122-125。
2) SONY株式会社、「FeliCa　非接触ICカード技術 NFCの定義」、https://www.sony.co.jp/Products/FeliCa/NFC、2022/07/23
3) GitHub enterprise, nfcpy, https://github.com/nfcpy/nfcpy/blob/master/docs/overview.rst, 2022/07/23
4) nfcpy, Python module for near field communication, https://nfcpy.readthedocs.io/en/latest/index.html, 2022/07/23
5) ALLDATASHEET.COM , PN532 Datasheet(PDF) – NXP Semicondactors, https://pdf1.alldatasheet.com/datasheet-pdf/pdf/595327/NXP/PN532.html, 2022/07/23
6) Google, Gmail for Developers > Gmail API Python Quickstart, https://developers.google.com/gmail/api/qickstart/python#step_3_run_the_sample, 2022/07/24

脚注

A) 現在はpython3.8以降へも対応している。
B) 現在のOSではCPU温度情報でGPIOに信号を出せる仕様となっているため、トランジスタのみで可能となった。

ソースコード

フォーマットに合わせるために一部書き換えている。各パスワード等は伏字としている。

a) nfc_read_db.py NFCタグを読み取りデータベースへ

```
#!/usr/bin python 2.7
# char set *-utf-8-*
# read nfc tag then insert into mysql
pid6101/students

import nfc
import datetime
import mysql.connector as mysql
import time
import pigpio

old = ''
AtWhen = int(time.time())

connect = mysql.connect(
        host = 'localhost', port = '3306',
        user='pid', password='********',
                database='pid6101'
        )
cursor = connect.cursor()

GPIO_READ = 15
pi = pigpio.pi()
pi.set_mode(GPIO_READ, pigpio.OUTPUT)
# to communicate with tkinter

def on_startup(targets):
   global AtWhen, old
   print ("Put your card.")
   if int(time.time()) - AtWhen > 3:
   old =''
   return targets

def on_connect(tag):
   global old, connect, cursor, AtWhen
   if tag != '':
   if tag.ndef:
      if tag.ndef.length > 0:
      print (tag.ndef.length)
      Tag = format(tag).split()
      if len(Tag) == 4:
         if Tag[2].strip("") == 'NTAG213':
         records = tag.ndef.records
         d = []
         for record in records:
            d = str(record).split()
         if len(d) > 5:
            if (d[6] != old):
            Date = datetime.datetime.now().
strftime("%Y-%m-%d")
            Time = datetime.datetime.now().
strftime("%H:%M:%S")
            who = d[8] + ' ' + d[9]
            SQL = 'INSERT INTO students VALUES ('
            SQL = SQL + d[6] + ',' + who + ','
            SQL = SQL + """+ str(Date) + "","
            SQL = SQL + """+ str(Time) + "");"
            print (SQL)
            try:
               cursor.execute(SQL)
               connect.commit()
            except:
               connect.rollback()
               raise
            f = open('/home/pid/ramdisk/attend.txt',
'w')
            f.write(d[6] + ':' + who)
            f.close()
            old = d[6]
            pi.write(GPIO_READ,1)
            time.sleep(0.005)
            pi.write(GPIO_READ,0)
            AtWhen = int(time.time())

def on_release(tag):
   print ("on_release()")
   if int(time.time()) - AtWhen > 3:
   old =''
   if tag.ndef:
   print tag.ndef.records[0]

clf = nfc.ContactlessFrontend('usb')   #for Pasori
#clf = nfc.ContactlessFrontend('tty:USB0:pn532')
```

```
if clf:
   try:
   while True:
      if int(time.time()) - AtWhen > 3:
      old=''
      print ("Clf: {}".format(clf))
      clf.connect(rdwr={
      'on-startup': on_startup,
      'on-connect': on_connect,
      'on-release': on_release
      })
   except KeyboardInterrupt:
   clf.close()
   cursoe.close()
   connect.close()
   thred.__stop()
```

b)　saver2.py　画面表示他

```
#!/usr/bin/python3
# coding:utf-8

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sys
import datetime
import time
import pigpio
import askDB
import csv
import random
import six_oclock

GPIO_READ = 15
GPIO_CUE = 5
# to calibrate RTC of trinket Low when 12:30:00
GPIO_BELL = 21
Dark_PID = 'darkolivegreen'
Mid_PID = 'teal'
Light_PID = 'lightseagreen'
Pale_PID = 'paleturquoise'
BG_color = Dark_PID
Width = 480
Height = 320
AtWhen = int(time.time())

class Application(tk.Tk):
   global Width, Height, flag
   def __init__(self):
      tk.Tk.__init__(self)
      self.geometry(str(Width)+'x'+str(Height))
      self.attributes("-fullscreen", True)
      self.frame = Saver_page(self)
      self.frame.pack(expand=True, fill="both")

   def switch_frame(self, new_frame):
      self.frame.pack_forget()
      self.frame = new_frame(self)
      self.frame.pack(expand=True, fill="both")

   def backToSaver(self):
      self.frame.pack_forget()
      self.frame = Saver_page(self)
      self.frame.pack(expand=True, fill="both")

class Saver_page(tk.Frame):
   global Width, Height, flag
   def __init__(self, master=None, **kwargs):
      do_nfc=pi.callback(GPIO_READ,
pigpio.RISING_EDGE, self.NFC_read)
      tk.Frame.__init__(self, master, **kwargs)
      master.geometry(str(Width)+'x'+str(Height))
      master.title("NFC_PID_OoS")
      self.grid(column=0, row=0, sticky=tk.NSEW)
      self.master.config(bg=Dark_PID)
      self.canvas= tk.Canvas(self, width=Width,
height=Height, bg=Dark_PID)
      self.canvas.pack()
      self.count = 0
      self.master.after(60,self.update)

   def _close(self):
      self.master.switch_frame(Display_page)

   def update(self):
      # return when times up
      NowHMS = datetime.datetime.now()
.strftime('%H:%M:%S')
      if NowHMS == '12:30:00':
         pi.write(GPIO_CUE, pigpio.LOW)
         time.sleep(0.05)
         pi.write(GPIO_CUE, pigpio.HIGH)
      if NowHMS == '18:00:00':
         # send mail
         six_oclock.send_gmail()
      for i in koma:
         if i[0] == NowHMS:
         flag = False
         self.canvas.create_text(100,100, text=
'Times Up', font=('', 20, 'bold'), fill ='red')
         self._close()
      self.count = self.count+1
      if self.count > 60:
         self.canvas.delete('mes')
         self.canvas.delete('time')
         x = random.randint(160,320)
         y = random.randint(80,290)
         self.canvas.create_text(x, y,
text = 'Out of Service Now.',
            font=('', 20, 'bold'), fill=Mid_PID,
tag='mes')
         self.canvas.create_text(x, y +26, text =

datetime.datetime.now().strftime('%Y/Ym/%d
            %H:%M:%S'),
font=('',20,'bold'),fill=Mid_PID,
            Tag='time')
         self.connt=0
         self.count=0
      self.master.after(60,self.update)

def NFC_read(self, gpio, level, tick):
            pass

class Display_page(tk.Frame):
   global Width, Height, flag, AtWhen, win, pi,
askdb
   def __init__(self,master=None, **kwargs):
      do_nfc=pi.callback(GPIO_READ,
pigpio.RISING_EDGE, self.NFC_read)
```

```
      tk.Frame.__init__(self, master, **kwargs)
      self.my_canvas = tk.Canvas(self, width=Width,
height=Height, bg=Pale_PID)
      self.my_canvas.grid(row=0, column=0,
rowspan=3)
      self.my_canvas.pack()
      self.my_canvas.create_text(240,15,text=
'プロダクト・インテリアデザイン学科',
            font=('Helvetica',20,'bold'),
fill=Dark_PID)
      self.my_canvas.create_text(240,44,text=
'NFC カード出席登録システム',
           font=('Helvetica',20,'bold'), fill=Dark_PID)
      self.my_canvas.create_text(80,130,text='科目:',
font=("", 20, 'bold'), fill=Dark_PID)
      Sub_data = askdb.askSubject('subjects')
      Subject = Sub_data[2]

self.my_canvas.create_text(280,130,text=Subject,
font=("", 20, 'bold'),tag='kamoku', fill=Dark_PID)
self.my_canvas.createrectangle(120,244,360,284,
fill=Mid_PID)
      self.my_canvas.create_polygon(80,284,
400,284, 240,310, 80,284, fill=Mid_PID)
      self.my_canvas.create_text(240,270,
text='カードをタッチ',
          font=("",20,'bold'), fill=Pale_PID)
      self.Send = ttk.Button(self, text='送信',
command =self.send_data)
      self.Send.place(x=16, y=240)
      self.my_canvas.create_text(400,300,
text='(c)J.Sagara,2020',
          font=('Helvetica', 10, 'normal'),
fill=Mid_PID)
      self.master.after(30, self.update)

   def _close(self):
      self.master.backToSaver()

   def update(self): # loop
      global AtWhen, koma
      self.my_canvas.delete('time')

self.my_canvas.create_text(240,90,text=datetime.
datetime.now().strftime
('%Y/%m/%d %H:%M:%S"),font=('',20,'bold'),
Fill=Mid_PID, tag='time')
      diff = int(time.time()) - AtWhen
      if diff > 5:
         self.my_canvas.delete('who')
      # return when times up
      NowHMS =datetime.datetime.now().strftime
('%H:%M:%S')
      for i in koma:
         if i[1] == NowHMS:
         flag = False
         self._close()
         return
      self.master.after(30,self.update)

   def send_data(self):
      self.Send.state(['disabled'])
      self.Send.state(['pressed'])
      To = askdb.askSubject('subjects')
      teacher = To[5]
      m_addr = To[6]
      Koma = To[4]
      subject = To[2]
      Mes = '出席状況を' + m_addr + 'へ送信しますか?'
      ret = messagebox.askyesno('確認', Mes)
      self.Send.state(['!disabled'])
      self.Send.state(['!pressed'])
      if ret == True:
         askdb.setdata(subject, Koma, teacher,
m_addr)

   def NFC_read(self, gpio, level, tick):
      # read attend.txt and show it
      global AtWhen
      f = open ('/home/pid/ramdisk/attend.txt', 'r')
      who = f.read()
      who.strip()
      f.close()
      print(who)
      pi.write(GPIO_BELL, pigpio.HIGH)
      time.sleep(0.01)
      pi.write(GPIO_BELL, pigpio.LOW)
      self.my_canvas.delete('who')
      self.my_canvas.create_text(240, 180, text =
 who.replace("", ""), font=('', 20, 'bold'), tag='who')
      AtWhen = int(time.time())

def Saver(): #初期化
   global koma, pi, app, flag, askdb
   flag = True
   askdb = askDB.askDB()

   pi = pigpio.pi()
   pi.set_mode(GPIO_READ, pigpio.INPUT)
   pi.set_pull_up_down(GPIO_READ,
pigpio.PUD_DOWN)
   pi.set_mode(GPIO_CUE, pigpio.OUTPUT)
   pi.write(GPIO_CUE, pigpio.HIGH)
   pi.set_mode(GPIO_BELL, pigpio.OUTPUT)
   pi.write(GPIO_BELL, pigpio.LOW)

   csv_obj = csv.reader(open('PID_koma.dat', 'r'))
   koma = [ v for v in csv_obj]
   csv_obj = csv.reader(open('PID_semester.dat', 'r'))
   sem = [ v for v in csv_obj]

   # run Application
   app = Application()
   app.mainloop()

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

c) askDB.py データベースからデータを取得

```
#!/usr/bin/python
# _*_ coding: utf-8 _*_

# ask semester is in ?
# ask subject now ?

from urllib.parse import urlparse
import mysql.connector as mysql
import datetime
import csv
```

```
import subprocess

class askDB():
  csv_obj = sv.reader
(open('/home/pid/PID_koma.dat', 'r'))
  Koma = [ v for v in csv_obj]
  csv_obj =csv.reader
(open('/home/pid/PID_semester.dat', 'r'))
  sem = [ v for v in csv_obj]

  def __init__(self):
    pass

  def askTF(self):   # ret 1-5 or -1 @out of time
    Now = datetime.datetime.now()
    NowHMS = Now.strftime('%H:%M:%S')
    val = 1
    for i in self.Koma:
      if (NowHMS >= i[0]) and (NowHMS < i[1]):
        break
      val = val +1
    if (val > 5):
      return (-1)
    else:
      return val

  def askSemester(self):
    semester = -1
    Today =
datetime.date.today().strftime('%m-%d')
    if (Today >= self.sem[0][0].strip()) and
 (Today <= self.sem[0][1].strip()):
      semester= 0
    elif (Today >= self.sem[1][0].strip()) and
(Today <= '12-27'):
      semester= 1
    elif (Today >= '01-05') and
(Today <= self.sem[1][1].strip()):
      semester= 1
    return semester

  def askSubject(self, table):
    Semester = self.askSemester()
    val = self.askTF()
    if (Semester != -1) and (val != -1):
      connect = mysql.connect(
        host='localhost', port = '3306',
        user='pid', password='********',
        database='pid6101')
      csr = connect.cursor(buffered=True)
      wkd = datetime.datetime.today().weekday()
      sql = 'SELECT * from ' + table + ' where
semester= ' + str(Semester) + ' AND time_frame= '
      sql += str(val) + ' AND weekday = ' +str(wkd)
+';'
      csr.execute(sql)
      Result = csr.fetchall()
      rowcount= csr.rowcount
      csr.close()
      connect.close()
      if (rowcount > 0):
        Ret_val = Result[0]
      else:
        Ret_val = (-1, Semester,
'対応科目はありません',-1, val, 'Nobody', 'sagara-
j@kobe-du.ac.jp')
    else:
      Ret_val = (-1, -1, '期間外です',-1, 0, 'Nobody',
'sagara-j@kobe-du.ac.jp')
    return Ret_val

  def setdata(self, name, koma, teacher, m_addr):
# select students by koma
    Today = datetime.datetime.now()
    today = Today.date()
    path = '/home/pid/ramdisk/'
    filename = 'attendees.csv'
    if koma == 0:
    koma = self.askTF()
    connect = mysql.connect(
      host='localhost', port = '3306',
      user='pid', password='**********',
      database='pid6101')
    cursor = connect.cursor(buffered=True)
    SQL = 'SELECT code, name, date, time '
SQL+= 'from students WHERE date = '
    SQL += "'" + str(today) + "' and time
BETWEEN "
    SQL += "'" + str(self.Koma[koma-1][0])
SQL += "' AND '" + str(self.Koma[koma-1][1])
    SQL += "';"
    cursor.execute(SQL)
  #open file /home/pid/data.csv
    with open(path + filename, 'w') as file:
      for row in cursor.fetchall():
      file.write(row[0]+','+ row[1]
+','+str(row[2])+','+str(row[3])+'¥n')
    cursor.close()
    connect.close()
    print (m_addr)
    exec_list = ['/usr/local/bin/python3.8',
'/home/pid/office365mail.py', m_addr,teacher,
name,filename,path ]
    print (exec_list)
    subprocess.call (exec_list)
    print (name + ' is send. to:' + m_addr)

  def askSubjectsData(self): #
    pass

if __name__ == "__main__":
  exit(0)
```

d) office365mail.py #メール送信処理

```
#!/usr/bin/python3
# _*_ coding: utf-8 _*_
# to suit office365, python3.8.0 only work
# read file and send mail

import smtplib
import datetime
import sys
import json
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from email.utils import formatdate
```

```
def
create_message(from_adr,to_adr,who,subject,mine,
,attach_file):
   msg = MIMEMultipart('alternative')
   msg['Subject'] = subject
   msg['From'] = from_adr
   msg['to'] = to_adr
   cset = 'utf-8'
   with open (attach_file['path'], 'rt',encoding='utf-
8')
 as file:
      list = file.read()
   if len(list) != 0:
      Body = who + '¥r¥n¥r¥n'
      Body += body +
 'の利用者データを CSV で添付しています。
¥r¥n¥r¥n'
      Body += '¥r¥nfrom your loyal pid-robot.¥r'
Body += '----------¥r'+ list
      Body = MIMEText(Body, 'plain', 'utf-8')
      msg.attach(Body)
      today=datetime.datetime.now().date()
      Fbody=attach_file['name'].split('.')
      Fname=body+str(today)+'.'+Fbody[1]
      attachment = MIMEText(list, 'plain', 'utf-8')
      attachment.add_header
('Content-Disposition','attachment', filename=('utf-
8', '', Fname))
      msg.attach(attachment)
      return msg
   else:
      msg = ""
      return msg

def send_nullmessage(from_adr, to_adr,
who,rooms):
   Body = who + '¥n¥n'
Body += datetime.datetime.now().
Strftime('%Y-%m-%d %H:%M:%S')
   Body += '¥r¥n 今日は利用者データが'
Body += 'ありません。'¥r¥n¥r¥n'
Body += ' from your loyal pid-robot.'
   msg = MIMEText(Body, "html")
   msg['Subject'] = f'今日の{rooms}スタジオ利用者'
   msg['From'] = from_adr
   msg['To'] = to_adr
   sendGmail(from_adr,to_adr,msg)

def sendGmail(from_adr, to_adr, msg):
   host, port = 'smtp.office365.com', 587
   account = '*******@kobe-du.ac.jp'
   password = '*******'
   smtp = smtplib.SMTP(host, port)
   smtp.starttls()
   smtp.login(account, password)
   smtp.sendmail(from_adr, to_adr, msg.as_string())
   smtp.quit()

def sendcsv(to, who, what, filename, path):
   # read parameter
   rooms = ''.join(what)
   to_adr = to
   from_adr = 'pid-robot@kobe-du.ac.jp'
   subject = f'今日の{rooms}スタジオ利用者'
   mine = {'type':'text',
'subtype':'comma-separated-values'}
   attach_file={'name':filename,
'path':path+filename}

   if ' ' in to or ';' in to or ',' in to:
      toA = []
      toA = to.replace(' ','').replace(';',',').split(',')
      for t in toA:
         msg =,create_message(from_adr,str(t),who,
subject,what,mine,attach_file)
         if len(msg) != 0:
            sendGmail(from_adr,str(t),msg)
            print ('Send mail to ', + str(t)
         else:

send_nullmessage(from_adr,str(t),who,rooms)
            print ('No data to send.')
   else:
      msg = create_message(from_adr, to_adr,who,
            subject,what,mine,attach_file)
      if len(msg) != 0:
         sendGmail(from_adr, to_adr, msg)
         print ('Sended mail to ' + to)
      else:
         send_nullmessage(from_adr,to,who,rooms)
         print ('No data to send.')

if __name__ == "__main__":
   args = sys.argv
   if len(args) < 6:
      sys.exit()
   else:
      sendcsv(args[1], args[2], args[3], args[4],
args[5])
```

e) entryExit2.py 入退室記録
Saver2.py をベースに作成

```
#!/usr/bin/python3
# coding:utf-8

import tkinter as tk
from tkinter import ttk
import sys
import datetime
import time
import gc
import mysql.connector as mysql
import csv
import random
import json
import office365mail
import subprocess
import pyautogui
import os

DEBUG = True

GPIO_BELL = 21
GPIO_KILLME = 10
GPIO_HR = 20   # signal of alive or dead
closeTime='21:00:00'
sleep = False
Dark_PID = 'darkolivegreen'
Mid_PID = 'teal'
Light_PID = 'lightseagreen'
```

```
Pale_PID = 'paleturquoise'
BG_color = Dark_PID
Width = 800
Height = 480
Center = Width / 2

toWhom = ' '
toMaddr = ''
subject = 'プロダクト・インテリアデザイン学科教室自習入退室記録'
name = 'PiD@KOBE-DU'
ramdisk = '/home/pid/ramdisk/'
datafile = 'data.csv'
dbtable = 'pidstudios'
direction = '入室'
room2F = ['INT スタジオ','6201PC 室','PCD スタジオ']
room1F = ['ＵＤスタジオ','ロッカー室','6102 スタジオ']
rooms = room1F
roomBtn = [60,240],[300,240],[520,240]
paramFile = 'parameter.json'
entryBtn = 80,330
exitBtn = 550,330
sendBtn = 16,440

class Application(tk.Tk):
  global Width, Height, sleep
  def __init__(self):
    tk.Tk.__init__(self)
    self.geometry(str(Width)+'x'+str(Height))
    self.attributes("-fullscreen", True)
    NowHMS = datetime.datetime.now()
.strftime('%H:%M:%S')
    sleep = False
    if NowHMS < '08:20:00':
      self.frame = Saver_page(self)
    elif NowHMS > '21:00:00':
      self.frame = Saver_page(self)
    else:
      self.frame = Display_page(self)
    self.frame.pack(expand=True, fill="both")

  def switch_frame(self, new_frame):
    self.frame.pack_forget()
    self.frame = new_frame(self)
    self.frame.pack(expand=True, fill="both")

  def backToSaver(self):
    self.frame.pack_forget()
    self.frame = Saver_page(self)
    self.frame.pack(expand=True, fill="both")

class Saver_page(tk.Frame):
  global Width, Height, sleep, pi, DEBUG,
closeTime
  def __init__(self, master=None, **kwargs):
    tk.Frame.__init__(self, master, **kwargs)
    gc.collect()
    master.geometry(str(Width)+'x'+str(Height))
    master.title("NFC_PID_OoS")
    self.grid(column=0, row=0, sticky=tk.NSEW)
    self.master.config(bg=Dark_PID)
    self.canvas= tk.Canvas(self, width=Width,
height=Height, bg=Dark_PID)
    self.canvas.create_text(Width/2,Height/2,
      bg=DarkPID',fill='white',justify='center',
font=('',32,'bold'))
    self.canvas.pack()
    self.count = 0
    self.heart = True
    if DEBUG:
      with open(ramdisk+'report.log','a') as
self.logfile:
        self.logfile.write('Saver page:' +:%S")+'¥n')
    self.jobIDS = self.master.after(20,self.update)

  def _close(self):
    self.master.switch_frame(Display_page)

  def str2timedelta(self,s):
    hours,minutes,seconds=map(int, s.split(':'))
    return datetime.timedelta(hous=hours,
minutes=minutes. seconds=seconds)

  def add60sec(self,s):
    now = self.str2timedelta(s)
    dif = self.str2timedelta('00:00:05')
    return now+dif

  def update(self):
    if self.heart is True:
      pi.write(GPIO_HR,pigpio.HIGH)
    else:
      pi.write(GPIO_HR,pigpio.LOW)
    self.heart = not self.heart
    NowHMS = datetime.datetime.now()
.strftime('%H:%M:%S')
    # return when times up
    if NowHMS == '08:20:00':
      if self.jobIDS is not None:
        self.master.after_cancel(self.jobIDS)
        self.jobIDS = None
        time.sleep(1)
      self._close()
    now = self.str2timedelta(NowHMS)
    if now > self.add60sec(closeTime):
      if DEBUG:
        with open(ramdisk+'report.log','a')
 as self.logfile:
          self.logfile.write('Close App:' +
datetime.datetime.now().
Strftime('%Y/%m/%d %H:%M:%S")+'¥n')

      if self.jobIDS is not None:
        self.master.after_cancel(self.jobIDS)
        self.jobIDS = None
      global app
      app.sleep = True
      app.destroy()
    self.jobIDS = self.master.after(20,self.update)

class Display_page(tk.Frame):
  # mode 0：show select direction and choose
  # mode 1：wait 5 seconds for NFC touch,
#           if no return to mode 0
  # mode 2：add data to database, return to mode
0
  global Width, Height, flag,win, pi,
global DEBUG, sleep, closeTime
```

```
  def __init__(self,master=None, **kwargs):
    ttk.Frame.__init__(self, master, **kwargs)
    self.canvas = tk.Canvas(self, width=Width,
height=Height, bg=Pale_PID)
    self.canvas.grid(row=0, column=0, rowspan=3)
    self.canvas.pack()
    self.canvas.create_text(Center,18,
text='プロダクト・インテリアデザイン学科',
      font=('Helvetica',24,'bold'), fill=Dark_PID)
    self.canvas.create_text(Center,52,
text='NFC タグスタジオ入退室記録システム',
      font=('Helvetica',24,'bold'), fill=Dark_PID)
    self.style = ttk.Style()
    self.style.theme_use('classic')
    self.style.configre("W.TButton",font=("",22),
        foreground=Pale_PID,
        background='teal')
    self.Room1 = ttk.Button(self.canvas,
        text=rooms[0],style="W.TButton",
        padding=[6,12], command=self.isRm1)
    self.Room1.place(x=roomBtn[0][0],
y=roomBtn[0][1])
    self.Room2 = ttk.Button(self.canvas,
text=rooms[1],style="W.TButton",
padding=[6,12], command=self.isRm2)
    self.Room2.place(x=roomBtn[1][0],
y=roomBtn[1][1])
    self.Room3 = ttk.Button(self.canvas,
text=rooms[2],style="W.TButton",
padding=[6,12], command=self.isRm3)
    self.Room3.place(x=roomBtn[2][0],
y=roomBtn[2][1])

    self.imgIn = tk.PhotoImage(file='in.gif')
    self.imgOut = tk.PhotoImage(file='out.gif')
    self.Entry = ttk.Button(self.canvas,text='入室'
,image=self.imgIn, compound='top',
padding=[6,12],

command=self.isEntry,state=tk.DISABLED)
    self.Entry.place(x=entryBtn[0], y=entryBtn[1])
    self.Exit = ttk.Button(self.canvas,text='退室',
image=self.imgOut,compound='top',
padding=[6,12],

command=self.isExit,state=tk.DISABLED)
    self.Exit.place(x=exitBtn[0], y=exitBtn[1])
    self.canvas.create_text(760,470,
 text='(c)J.Sagara,2021',
      font=('Helvetica', 8, 'normal'),
fill=Mid_PID)
    if DEBUG:
      with open(ramdisk+'report.log','a') as
self.logfile:
        self.logfile.write('Log start at::'
+ datetime.datetime.now().strftime
('%Y/%m/%d %H:%M:%S")+'¥n')
  pyautogui.FAILSAFE=False
    self.jobID = self.master.after(50, self.update)
    self.mode = 0
    self.mailWasSend = False
    self.AtWhen = int(time.time())
    self.heart = True

  def _close(self):
    if DEBUG:
      with open(ramdisk+'report.log','a') as
self.logfile:
        self.logfile.write('_close called at:' +
datetime.datetime.now().strftime
("%Y/%m/%d %H:%M:%S")+'¥n')
    self.master.backToSaver()

  def showArrow(self,Col):
    self.canvas.create_rectangle
(Center-120,400,Center+120,
440,fill=Col,outline=Col)
    self.canvas.create_polygon(Center-160,440,
 Center,470, Center+160,440,
Center-160,440, fill=Col)

  def str2timedelta(self,s):
    hours,minutes,seconds=map(int, s.split(':'))
    return dtetime.timedelta(hours=hours,
        minutes=minutes,seconds=seconds)

  def add5sec(self,s): # add 10 sec.
    now = self.str2timedelta(s)
    dif = self.str2timedelta('00:00:10')
    return now+dif

  def update(self): # loop
    self.canvas.delete('time')
    self.canvas.create_text(Center,90 ,
text=datetime.datetime.now().strftime(
"%Y/%m/%d %H:%M:%S"),
      font=('Helvetica', 24, 'bold'),
fill='blue', tag='time')
    diff = int(time.time()) - self.AtWhen
    if diff > 1:
      self.canvas.delete('who')
      self.canvas.delete('id')
    # move mouse cursor to top
    pyautogui.moveTo(10,10)
    if self.heart is True:
      pi.write(GPIO_HR,pigpio.HIGH)
    else:
      pi.write(GPIO_HR,pigpio.LOW)
    self.heart = not self.heart
    NowHMS = datetime.datetime.now().
strftime('%H:%M:%S')
    now = self.str2timedelta(NowHMS)
    if self.str2timedelta(closeTime) <= now
< self.add5sec(closeTime):
      if not self.mailWasSend:
        self.setData()
      if self.jobID is not None:
        self.master.after_cancel(self.jobID)
        self.jobID = None
        time.sleep(0.1)
      sleep = True
      self._close()

    if self.mode == 0:
      self.canvas.delete('mes0')
      self.canvas.delete('mes1')
      self.showArrow(Pale_PID)
      self.canvas.create_text(Center,140,
text='部屋を選択してください',
        font=("",20,'bold'),
```

```
fill=Dark_PID,tag='whichR')
        self.Entry.state(['disabled'])
        self.Exit.state(['disabled'])
        self.Room1.state(['!disabled'])
        self.Room2.state(['!disabled'])
        self.Room3.state(['!disabled'])
        # erase NFC data
        with open(ramdisk+'nfc_data.csv','w') as file:
          file.write('')

      if self.mode == 1: # Room was selected
        self.canvas.delete('whichR')
        self.canvas.create_text(Center,140,
text='出入りを選択してください',
            font=("",20,'bold'),
fill=Dark_PID,tag='whichD')
        self.direction = '不明'
        self.Entry.state(['!disabled'])
        self.Exit.state(['!disabled'])

        self.Room1.state(['disabled'])
        self.Room2.state(['disabled'])
        self.Room3.state(['disabled'])
        if int(time.time()) - self.AtWhen > 5:
          self.AtWhen = int(time.time())     # timer
reset
          self.canvas.delete('whichD')
          self.mode = 0

      if self.mode == 2:
# show touch anounce then check nfc read
        self.canvas.delete('whichR')
        self.canvas.delete('whichD')
        self.canvas.create_text(Center,140,
  text=self.mes,font=("",20,'bold'),
  fill=Dark_PID, tag = 'mes0')
        self.showArrow(Mid_PID)
        self.canvas.create_text(Center,420,
text='カードをタッチ',
            font=("",20,'bold'), fill=Pale_PID, tag =
'mes1')
        with open(ramdisk+'nfc_data.csv','r') as file:
          self.res = file.read()
        if self.res != '':
          self.canvas.delete('mes0')
          self.canvas.delete('mes1')
          self.showArrow(Pale_PID)
          self.res = self.res.split(',')
          self.who = self.res[3].strip()
          self.id = self.res[2].strip()
          pi.write(GPIO_BELL, pigpio.HIGH)
          time.sleep(0.01)
          pi.write(GPIO_BELL, pigpio.LOW)
          self.canvas.create_text(Center, 180,
text = self.id, font=('', 20, 'bold'), tag='id')
          self.canvas.create_text(Center, 210, text =
self.who, font=('', 20, 'bold'), tag='who')
          self.mode = 3
          if DEBUG == True:
            self.logfile =
open(ramdisk+'report.log','a')
            self.logfile.write(str(self.mode) + ' at ' +
str(NowHMS)+'¥n')
            self.logfile.close()
          self.AtWhen = int(time.time())
# set new timer
        if int(time.time()) - self.AtWhen > 5:
          self.AtWhen = int(time.time())    # timer
reset
          self.mode = 0

      if self.mode == 3:
        Today = datetime.datetime.now().strftime(
'%Y-%m-%d')
        connect = mysql.connect(
          host = 'localhost', port = '3306', user =
'pid',
          password = '********', database =
'studios')
        csr = connect.cursor()
        sql = 'INSERT INTO ' + dbtable +
' (Room, Date, Time, StudentID, '
        sql += 'StudentName, Direction) VALUES ("'
        sql += self.room + '","' +str(Today) + '","'
sql += str(NowHMS)+'","' +self.id +'","'+self.who
        sql += '","' + self.direction +'");'
        csr.execute(sql)
        connect.commit()
        if DEBUG == True:
          self.logfile = open(ramdisk+'report.log','a')
          self.logfile.write(sql+'¥n')
          self.logfile.close()
        csr.close()
        connect.close()
        self.mode = 0
        self.AtWhen = int(time.time())

      self.jobID = self.master.after(20,self.update)

    def isRm1(self):
      self.room = rooms[0]
      self.mode = 1
      self.AtWhen = int(time.time())

    def isRm2(self):
      self.room = rooms[1]
      self.mode = 1
      self.AtWhen = int(time.time())

    def isRm3(self):
      self.room = rooms[2]
      self.mode = 1
      self.AtWhen = int(time.time())

    def isEntry(self):
      self.direction = '入室'
      self.mes = 'おはようございます'
      self.mode = 2
      self.canvas.delete('mes0')
      self.AtWhen = int(time.time())

    def isExit(self):
      self.direction = '退室'
      self.mes = 'おつかれさまでした'
      self.mode = 2
      self.canvas.delete('mes0')
      self.AtWhen = int(time.time())

    def setData(self):
      global dbtable, ramdisk, datafile,
```

```
global toMaddr,,toWhom

      Today = datetime.date.today()
      connect = mysql.connect(
          host = 'localhost', port = '3306', user = 'pid',
          password = '*******', database = 'studios')
      csr = connect.cursor()
      sql = f'SELECT * from {dbtable} WHERE Date
= "{str(Today)}";'
      csr.execute(sql)
      self.file = open(ramdisk+datafile, 'w')
      for row in csr.fetchall():
        self.file.write(str(row[1]+',' + str(row[2]) +
str{row[3]) + ‘,’}
        self.fille.write(str(row[4]+’,’+str(row[5])+
            str(row[6])+’¥n’)
      self.file.close()
      csr.close()
      connect.close()
      if DEBUG:
        with open(ramdisk+'report.log','a') as
self.logfile:
          self.logfile.write(f'{str(Today)}:
office365mail send to '
 + toMaddr +'/'+toWhom+'¥n')
      if Today.weekday() != 6:
        office365mail.sendcsv(toMaddr, toWhom,
str(rooms), datafile, ramdisk)
      self.mailWasSend = True

def Saver():
  global pi, app, dbtable, toWhom, toMaddr,
global rooms, sleep

  # read parameter
  json_obj = open(paramFile, 'r')
  param = json.load(json_obj)
  dbtable = param['dbtable']
  toWhom = param['mailTo']
  toMaddr = param['mailToAdr']
  rooms = param['room']
  json_obj.close()
  # write down my Proces ID
  with open(ramdisk+'myPID.txt','w') as pid:
    pid.write(str(os.getpid()))
  # wait internet access
  inet = False
  cmd = ["ping","-c","2","google.com"]
  while inet:
    porc = subprocess.run(cmd,stdout=subprocess.
PIPE,stderr=subprocess.PIPE)
    if porc.stdout.decode('cp932') != "":
    inet = True
  time.sleep(20)   #wait to get time

  # run Application
  app = Application()
  sleep = False
  while not sleep:
    app.mainloop()
    break
  pi.write(GPIO_KILLME,pigpio.LOW)
  time.sleep(1)
  pi.stop()
  app.quit()
  return

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