

スマートものづくりIoT体験講座

～ 自社工場内で行える！IoTお試し講座～

セミナーの目的

1. ソリューションの内製化を見据え、PCを活用してプロトタイピングを行う足がかりを学ぶ
 - Windows10とpythonの利用
 - Webカメラの活用
2. QRコードを利用したソリューションのイメージを掴む
 - QRコードやバーコードをどのように活用する(できるのか)検討する足がかりにする
 - 生産管理に活用してみるケースを実習する
3. レガシーなデバイスをインターネットに接続するためのデバイスの活用イメージを掴む
 - 電気信号がデータになる仕組みを掴む
 - どうやってユーザーに通知するか、活用方法を考えてみる

セミナー内容

1. PCをQRコードスキャナとして利用する
2. 生産管理システムを利用したQRコードによる通材シミュレーションと製品のロス率を算定製品の通材を作ったQRコードスキャナにより実現し、システムに反映する
3. レガシーな設備をIoT化する
 - パトランプの接点情報(ON／OFF)をインターネット経由で参照する
 - メールでアラートを送る

1. PCをQRコードスキャナとして 利用する

PCをQRコードスキャナにする

PCに接続されたカメラをQRコードスキャナとして活用します

- Pythonを利用して実装します
- Windows10のPCをご用意ください
- (Web)カメラをご用意ください
- テキストエディタをご用意いただけると捗ります

セットアップ方法

実習にあたって、下記の通りセットアップを行います

1. Windows Store から「python3.9」をインストールする
 2. アプリケーションの実行に必要なライブラリのインストールを行います
 3. コードを実装します
 4. (読み取るQRコードを用意します)
 5. コードを実行してQRコードの読み取りを行います
- 【番外編: 時間が許せば試してみる】
 - アプリケーションを改修して、任意のQRコード画像を生成します

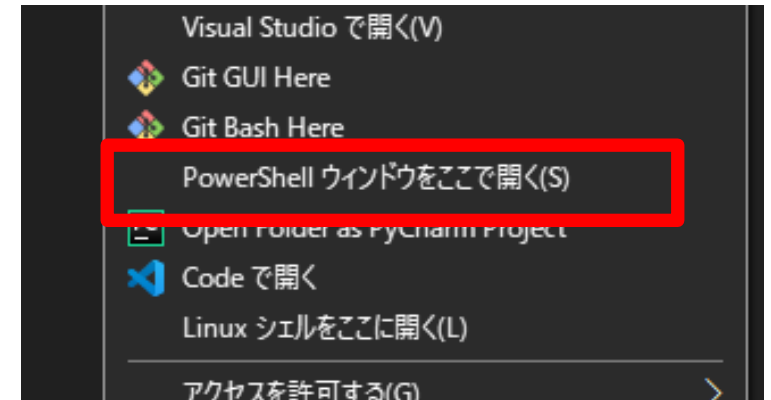
Microsoft Store から 「python3.9」をインストールする

- スタートメニューから「Microsoft Store」を探す
- 「Python」で検索してインストールする



アプリケーションの実行に必要なライブラリのインストールを行います

- アプリケーションを設置するディレクトリを作成します
 - デスクトップなどにディレクトリを新規作成する
- ディレクトリ内でShiftキーを押しながら右クリックし、下記を選択します
 - 「PowerShellウィンドウをここで開く」または「コマンドプロンプトをここで開く」
- コンソールが開いたら下記コマンドを入力します
 - `python3 -m pip install opencv-python`
 - `python3 -m pip install pyzbar`
 - `python3 -m pip install requests`
 - `python3 -m pip install qrcode`
 - `python3 -m pip install Pillow`



```
Windows PowerShell
PS E:\sample> python3 -m pip install opencv-python
Requirement already satisfied: opencv-python in c:\users\ctrl\appdata\local\packages\pythonsoftwaref
oundation.python.3.9_qbz5n2kfra8p0\localcache\local-packages\python39\site-packages (4.5.3.56)
Requirement already satisfied: numpy>=1.19.3 in c:\users\ctrl\appdata\local\packages\pythonsoftwaref
oundation.python.3.9_qbz5n2kfra8p0\localcache\local-packages\python39\site-packages (from opencv-pyt
hon) (1.21.1)
WARNING: You are using pip version 21.1.3; however, version 21.2.3 is available.
You should consider upgrading via the 'C:\Users\ctrl\AppData\Local\Microsoft\WindowsApps\PythonSoftw
areFoundation.Python.3.9_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
PS E:\sample> _
```


【参考】テキストエディタを用意する

下記のエディタを用意しておくとスムーズにソースコードを編集できます
(下に行くほどコード書くのに適している)

- メモ帳
- Visual Studio Code (おすすめ)
 - <https://azure.microsoft.com/ja-jp/products/visual-studio-code/>
- Terapad
 - <https://tera-net.com/library/tpad.html>
- サクラエディタ
 - <https://sakura-editor.github.io/>
- 秀丸エディタ
 - <https://hide.maruo.co.jp/software/hidemaru.html>

コードを実装します

- 実習を行うディレクトリにファイルを作成します
- (実習ファイル群にサンプルコードが添付されているので設置します)
- 実行時はコマンドプロンプト(PowerShell)で下記コマンドを実行

`python3 main.py`

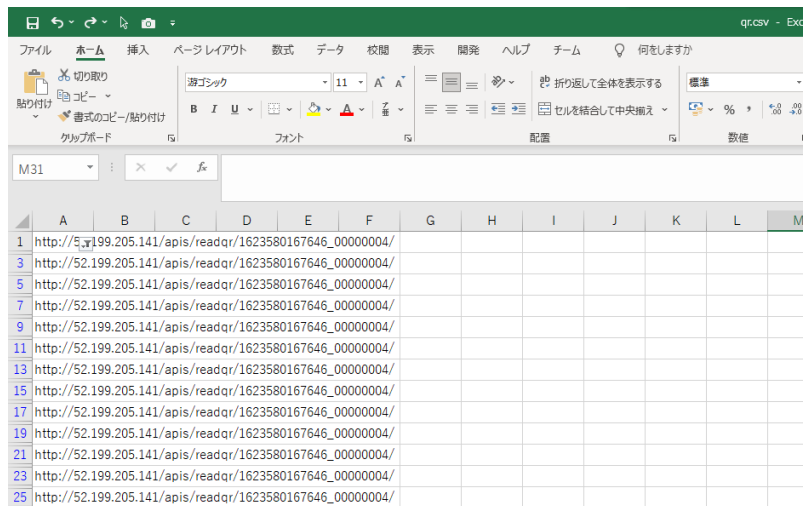
- 終了する場合は
 - コマンドプロンプト(PowerShell)上で「Ctrl+C」を連打
 - カメラウィンドウ上で「Q」ボタンを押す
- ソース編集後, 変更内容を適用するためには一度アプリケーションを終了して再立上げする必要があります

読み取ったQRコードをCSVファイルに書き出してExcelで開いてみる

- 下記のコードを追加する
 - L.85

```
if(len(ret) > 0):
    with open("qr.csv", "a") as f:
        f.write(ret[0] + "\n")
```

- 読み取り時にCSVファイルが出力されるのでExcelで開くとQRの読み取り結果を取得できる



QRコードを用意します

- 手元に実際に読み取るためのQRコードを用意します
 - (任意のコードを作りたい場合, 下記のようなWebサービスを利用すると便利です)
 - <https://qr.quei.jp/url.php>
 - https://chart.googleapis.com/chart?cht=qr&chs=300x300&chl=https://developers.google.com/chart/infographics/docs/qr_codes?authuser=0&choe=UTF-8
- 任意の文字列やURLなどをQRコードにエンコードして用意します
 - 作成したアプリケーションを起動して実際に読み込みを行い, 検証を行います
 - `python3 main.py` でカメラキャプチャウィンドウが立ち上がります
- 読み出しを行ったQRコードを処理する方法を考えてみる
 - URLにアクセスする
 - 別の機器やデバイス, サーバなどに送る
- このあとの実習で活用するために改修する
 - QRコードを用いた製品の受け入れ・実績入力
 - QRコードを読めたら一度だけQRコード内のURLにアクセスするように改修する

QRコードの生成も試す

- 実習を行うディレクトリにファイルを作成します
- (実習ファイル群にサンプルコードが添付されているので設置します)
- 実行時はコマンドプロンプト(PowerShell)で下記コマンドを実行

`python3 qrgen.py`

- 終了する場合は
 - コマンドプロンプト(PowerShell)上で「Ctrl+C」を連打
- ソース編集後、変更内容を適用するためには一度アプリケーションを終了して再立上げする必要があります

【参考】コマンドラインからの実行

- 実行時はコマンドプロンプト(PowerShell)で下記コマンドを実行

```
python3 qrgen_arg.py [target_string] [filename]
```

- (大量に作成する場合やリストとの照合を行いながら生成する場合) バッチファイルを用いて連続実行するのに向いている実装
- バッチファイルを作成して連続実行してみる

バッチファイルを用いた連続実行

- テキストファイルを作成する
 - 中身には実行したいコマンドを改行を用いて列挙する
- テキストファイルの拡張子を「〜〜.bat」に変更する
 - 拡張子(ファイル名末尾に付与された「.(ドット)」以降の文字)が表示されていない場合は事前に「フォルダオプション」から拡張子を表示しておくこと
- コマンドプロンプト(またはPowerShell)で実行する

2.生産管理システムを利用した QRコードによる通材シミュレーションと 製品のロス率を算定

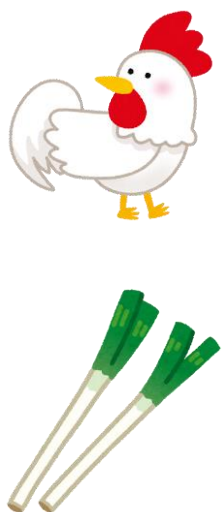
生産管理システムを利用して製品のロス率を計算する

- 先程構築したPCによるQRコードリーダーを用いて製品の通材シミュレーションを行い、製造実績からロス率の計算を行います
1. 前提とする工程とプロセスの課題感に関する整理
 2. IT/IoTによる課題の解決策に関する議論
 3. ロス率の見える化と算出

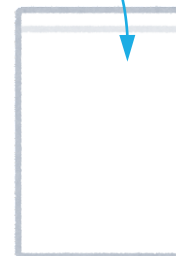
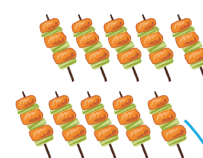
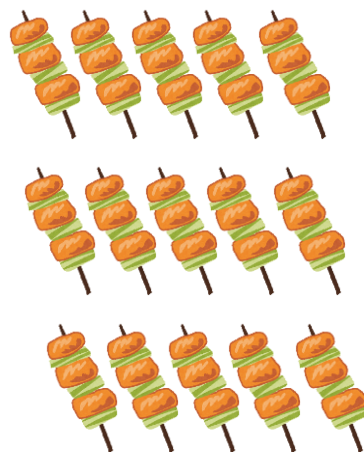
前提とする工程

- 焼きたりの串打ち & パッケージング工程

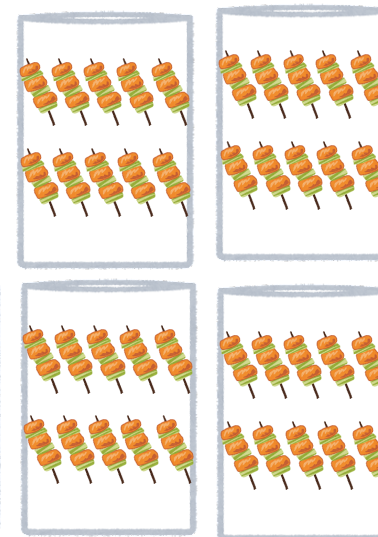
原料受入



材料切り出し
半製品の製造



梱包



出荷

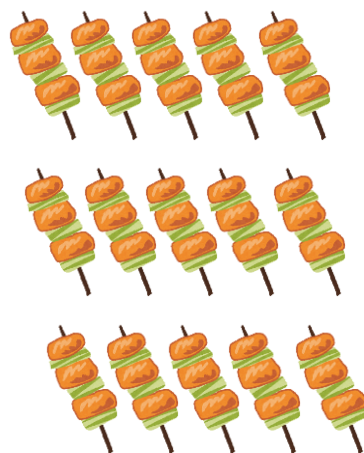
前提とする工程

- 焼きたりの串打ち & パッケージング工程

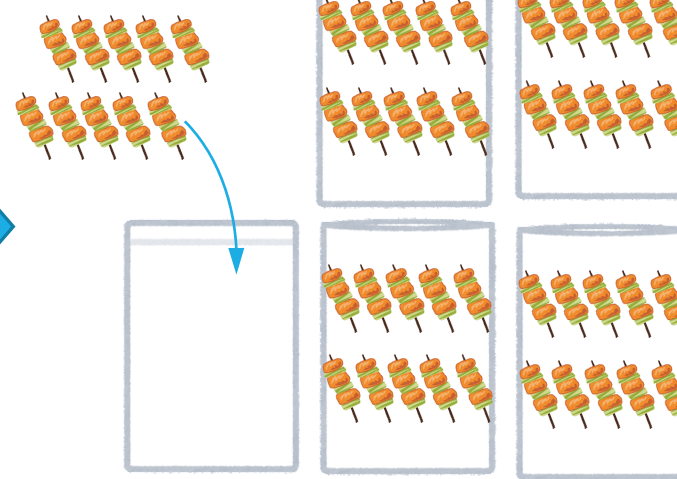
原料受入



材料切り出し
半製品の製造



梱包



出荷



製品の残重量と
抜き取り検査した消費重量が計測可能

パッケージに印字した
QRコードの読み取りが可能

生産管理システムを利用してみる

- ブラウザからアクセスして利用してみます
- アクセスすると各種メニューが表示されます



生産計画を確認する

- 「生産計画入力」ボタンを押下すると、当日予定されている製品の製造計画が表示されます
 - 通材が確認されていないものは「通材日時」が「0000-00-00 00:00:00」となっています
 - 通材確認後に通材日時が入力されます。

生産計画・実績一覧

Show entries

| パッケージID | 製品ID | コード | 単位重量 | 計測重量 | 入力日時 | 納期日時 | 通材日時 |
|---------|------|------------------------|-------|------|---------------------|---------------------|---------------------|
| 31 | 1 | 1626180678236_00000003 | 0.45 | 0 | 2021-07-13 21:51:18 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 30 | 1 | 1626180678236_00000002 | 0.45 | 0 | 2021-07-13 21:51:18 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 29 | 1 | 1626180678236_00000001 | 0.45 | 0 | 2021-07-13 21:51:18 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 28 | 1 | 1626180678236_00000000 | 0.45 | 0 | 2021-07-13 21:51:18 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 27 | 2 | 1626180644576_00000001 | 0.45 | 0 | 2021-07-13 21:50:44 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 26 | 2 | 1626180644576_00000000 | 0.45 | 0 | 2021-07-13 21:50:44 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 25 | 3 | 1626175287900_00000001 | 0.675 | 0 | 2021-07-13 20:21:27 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 24 | 3 | 1626175287900_00000000 | 0.675 | 0 | 2021-07-13 20:21:27 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 23 | 1 | 1626175284800_00000004 | 0.45 | 0 | 2021-07-13 20:21:24 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 22 | 1 | 1626175284800_00000003 | 0.45 | 0 | 2021-07-13 20:21:24 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 21 | 1 | 1626175284800_00000002 | 0.45 | 0 | 2021-07-13 20:21:24 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |

通材シミュレーションを行ってみる

- 本デモでは擬似的に通材する方法としてシミュレーション機能を用意しています
 - トップ画面から「製品実績入力用QR一覧」をクリックします
 - 製造計画で確認したパッケージに貼り付けされたQRコードが表示されるのでこれを読み取ります
 - 読み取り結果としてURLが指定されているのでアクセスすると通材が行われます
 - (右端の「リンク」絡むに表示された■をクリックすると直接ブラウザアクセスを行うことで通材ができます)

製造実績入力用QR一覧
QRコードにより実績入力を行うサ
ンプルページ

| パッケージID | 製品ID | QRコード | コード | 単位重量(kg) | リンク |
|---------|------|--|------------------------|----------|---|
| 14 | |  | 1623580167646_00000002 | 0.45 |  |
| 15 | 1 |  | 1623580167646_00000003 | 0.45 |  |

読み取ってブラウザアクセス

(または)
リンクからブラウザアクセス

通材シミュレーションを行う

- 通材シミュレーションを行うことによって製品が製造され、実績データに通材日時が入力されるとともに原材料が消費されます
- 通材シミュレーションを複数回実行後、消費した原材料(原材料残)について「原料残一覧」から確認できます。
 - 原料IDと原料の名称は「原料受入」から確認できます

原料残一覧

Show

100

 entries

Search:

| 原料残量ID | 原料残量計算日時 | 製造した製品ID | 原料ID | 推定残量 (kg) | 推定重量の差分 (前の推定から) | 実績残量 (kg) | 実績重量計測日時 |
|--------|---------------------|----------|------|-----------------|------------------|-----------|---------------------|
| 1 | 2021-06-13 16:01:51 | 0 | 1 | 30 | 0 | -1 | 2021-06-13 16:01:51 |
| 2 | 2021-06-13 16:02:11 | 0 | 3 | 15 | 0 | -1 | 2021-06-13 16:02:11 |
| 3 | 2021-06-13 16:02:40 | 0 | 2 | 45 | 0 | -1 | 2021-06-13 16:02:40 |
| 4 | 2021-06-13 16:02:52 | 0 | 4 | 10 | 0 | -1 | 2021-06-13 16:02:52 |
| 5 | 2021-06-13 16:03:03 | 0 | 5 | 19 | 0 | -1 | 2021-06-13 16:03:03 |
| 6 | 2021-06-13 16:03:16 | 0 | 6 | 100 | 0 | -1 | 2021-06-13 16:03:16 |
| 17 | 2021-08-01 18:18:33 | 1 | 1 | 29.594907407407 | 0.40509259259259 | -1 | 0000-00-00 00:00:00 |
| 18 | 2021-08-01 18:18:33 | 1 | 3 | 14.897959183673 | 0.10204081632653 | -1 | 0000-00-00 00:00:00 |

原料残一覧
原料残テーブルの閲覧

生産実績を確認する

- 「生産実績閲覧」ボタンを押下すると、当日予定されている製品の製造計画が表示されます
 - 通材が確認されていないものは「通材日時」が「0000-00-00 00:00:00」となっています
 - 通材確認後に通材日時が入力されます。

生産計画・実績一覧

Show entries

Search

| パッケージID | 製品ID | コード | 単位重量 | 計測重量 | 入力日時 | 納期日時 | 通材日時 |
|---------|------|------------------------|------|------------------|---------------------|---------------------|---------------------|
| 13 | 1 | 1623580167646_00000001 | 0.45 | 0.50713340891912 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 | 2021-08-01 18:18:33 |
| 14 | 1 | 1623580167646_00000002 | 0.45 | 0 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 15 | 1 | 1623580167646_00000003 | 0.45 | 0 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 16 | 1 | 1623580167646_00000004 | 0.45 | 0 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 17 | 2 | 1623580334441_00000000 | 0.45 | 0 | 2021-06-13 19:32:14 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 18 | 2 | 1623580334441_00000001 | 0.45 | 0 | 2021-06-13 19:32:14 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |
| 19 | 1 | 1626175284800_00000000 | 0.45 | 0 | 2021-07-13 20:21:24 | 0000-00-00 00:00:00 | 0000-00-00 00:00:00 |

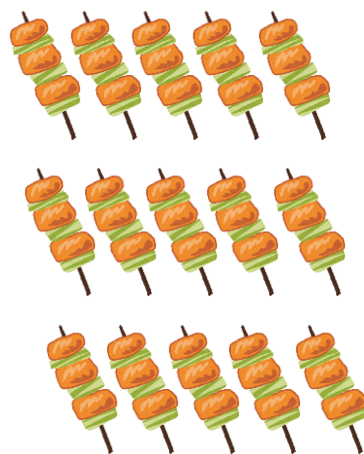
(改めて) 前提とする工程

- 焼きたりの串打ち & パッケージング工程

原料受入

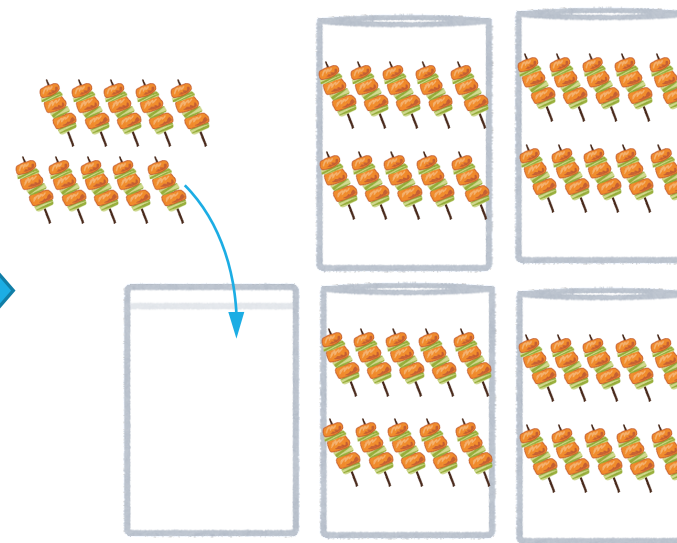


材料切り出し
半製品の製造



製品の残重量と
抜き取り検査した消費重量が計測可能

梱包



パッケージに印字した
QRコードの読み取りが可能

出荷



半製品の考え方

- 原料は直接製品にならず、半製品を経由する仕組みになっている
 - 製造したい製品は「ねぎま(皮付き)10本パック」の場合：
 - とり肉とネギを切り出して「ねぎま(皮付き)1本」をつくる
 - 10本集めてパッケージにする
- 半製品の構成内容を確認する方法：
 - 「半製品構成情報一覧」から構成情報を確認できます
 - 例)ねぎま(皮付き)1本に対し、原材料ID=1の鶏もも肉(皮付き)0.035[kg]とID=3の青ネギ0.01[kg]を消費する

半製品構成

Show entries

Search:

| 半製品構成ID | 半製品ID | 半製品名 | 原料ID | 原料名 | 単位重量 | 数量 |
|---------|-------|-----------|------|-----------|-------|----|
| 1 | 1 | ねぎま (皮付き) | 1 | 鶏もも肉(皮付き) | 0.035 | 1 |
| 2 | 1 | ねぎま (皮付き) | 3 | 青ネギ | 0.01 | 1 |
| 3 | 2 | ねぎま (皮なし) | 2 | 皮なし鶏もも肉 | 0.035 | 1 |
| 4 | 2 | ねぎま (皮なし) | 4 | 白ネギ | 0.01 | 1 |

Showing 1 to 4 of 4 entries

Previous

1

Next

製品のロス率を計算する

- 製品の通材シミュレーション後、通材日時が記録された上で原料残が更新されます
 - 原料残の時系列データから原料のロス率の計算を行います
1. トップページ「Excel出力」ボタンを押下してExcelファイルをダウンロードします
 2. 原料推定残量テーブルタブを開きます
 3. 対象の原料IDの「推定残量 (estimate_weight)」を製品IDでフィルタを用いて抽出します
 4. 対象の原料IDの「設計上の使用量」と「推定残量」を比較します
 - 設計上の仕様量: $(\text{半製品の単位重量}) \times (\text{製品パッケージ内の半製品数量})$

| カラム | 内容 |
|----------------------|---------|
| id | ID |
| inputdate | 入力日 |
| product_id | 製品ID |
| material_id | 原料ID |
| estimate_weight | 推定原料残 |
| diff_estimate_weight | 推定原料使用量 |
| actual_weight | 重量実績 |
| actualwinputdate | 重量実績入力日 |

QRコード読み取りアプリケーションを改修する (特定のURLにアクセスを行ってアプリケーションを終了する)

- 先程の実習でQRコード読み取りを行ったアプリケーションを改修する

- 改修項目

1. 読み取ったURLにアクセスする
2. 読み取りが終わったらアプリケーションを終了する

- カメラにQRコードが表示されている間は継続して読み取りするので、重複読み取りしないようにアプリケーションを終了する
- (連続して読み取りするにはどう実装すれば良いか考えてみる)

QRコード読み取りアプリケーションを改修する (特定のURLにアクセスを行ってアプリケーションを終了する)

- 特定のURLに読みだしたURLでアクセスする
 - リクエストそのものを行う関数を実装する
 - L.11 以降に下記の関数を実装する

```
import requests

def get_request(url):
    response = requests.get(url)
    print(response.status_code)    # HTTPのステータスコード取得
    print(response.text)         # レスポンスのHTMLを文字列で取得
```

- 一度だけ読みだしたらアプリケーションを終了する
 - L.81 行頭の「#」を消す

```
# if(len(ret) > 0):
#     get_request(ret[0])
#     break
```



```
if(len(ret) > 0):
    get_request(ret[0])
    break
```

別のシステムと連携する

QRコードを読み取った結果を書き出しして、他のシステムで出力したデータと突き合わせを行ってみる方法

- QRコードを書き出したCSVファイルを用意する
- システムから出力したExcelファイルを用意する
- QRコードのリストと実績データの突き合わせを行う
 - QRコードのリストが受入実績データなので、これに規格データなどを紐付ける

3.レガシーな機器をIoT化する

レガシーなIoT機器

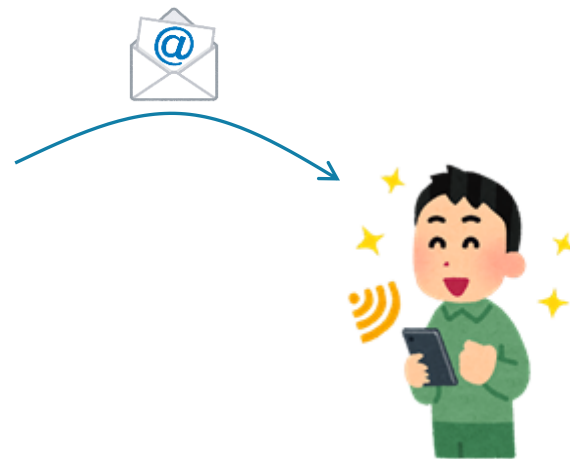
- ネットワーク接続方法が確保されていないなどの理由で、データとして信号を取り出すのが難しい機器
- たとえば・・・
 - リレーシーケンスの制御盤
 - ネットワーク接続のないスタンドアロンなPLC
 - パトライト
 - 押しボタン
 - トグルスイッチ

前提とするケース

- アラートや故障がでた際に光るパトライトがあったとする
- パトライトが光った際にメールを受け取るなどして迅速に対応して修繕・復帰を行いたい
- ただし、パトライト(や制御盤)には外部出力がなく、直接データを取得することはできない



点灯しても
近くにいないと見えない



点灯したら通知がくる

利用するデバイスについて

- SORACOM LTE-M Button Plus
 - 接点付きのIoTデバイス
 - ボタンを押した情報をLTE-M通信によりクラウド側に送信可能
 - シングルクリック・ダブルクリック・長押しの3通り
 - 外部接点を用いてボタンを押した場合と同様に接点情報を送信可能
 - 単4電池2本で動作
 - 8,118 円／個
 - SORACOMの各種サービスを利用してソリューションを構築可能



※本体のボタンと、ケーブルの接点が共通

LTE-Mボタンを利用してみる


- SORACOM コンソールにログインする
 - 左側のメニューから「ガジェット管理」→「SORACOM LTE-M Button for Enterprise」を選択する
 - 「デバイスを追加」ボタンを押して、対象のデバイスを選択し、「次へ: グループを選択」を押す.
 - 新規グループを作成して任意の名前をつけて「次へ: 設定を編集」を押す
 - 「メール送信を有効にする」にチェックし、送信先メールアドレスを入力する
 - 画面下部の「保存」ボタンを押して設定を保存する
 - ボタンを押すと設定済みのメールアドレスにメールが届く
-
- SORACOMユーザーコンソール
 - https://console.soracom.io/#/sam_login?o=OP0099631751&coverage_type=jp
-
- 【参考】新機能「SORACOM LTE-M Button」デバイスのスマート設定機能
 - <https://blog.soracom.com/ja-jp/2020/11/17/smart-configure-for-soracom-lte-m/>

グループを新規作成する

手元デバイスに貼り付けてある番号のデバイスを選択のうえ、グループを作成する



設定を保存するグループを選択

 LTE-M Button for Enterprise の設定は埋め込まれている SIM が所属する SIM グループに保存されます。SIM グループは後から変更ができます。同じ SIM グループ内にある SIM は同じ設定を共有します。

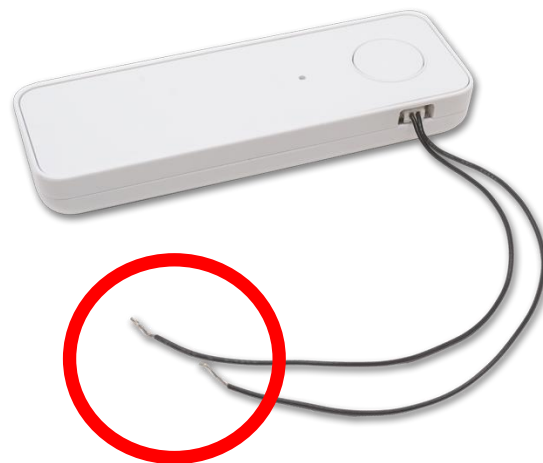
☒ 新規グループを作成

グループ名

実習用 1 番

ボタンではなく接点の情報を利用する

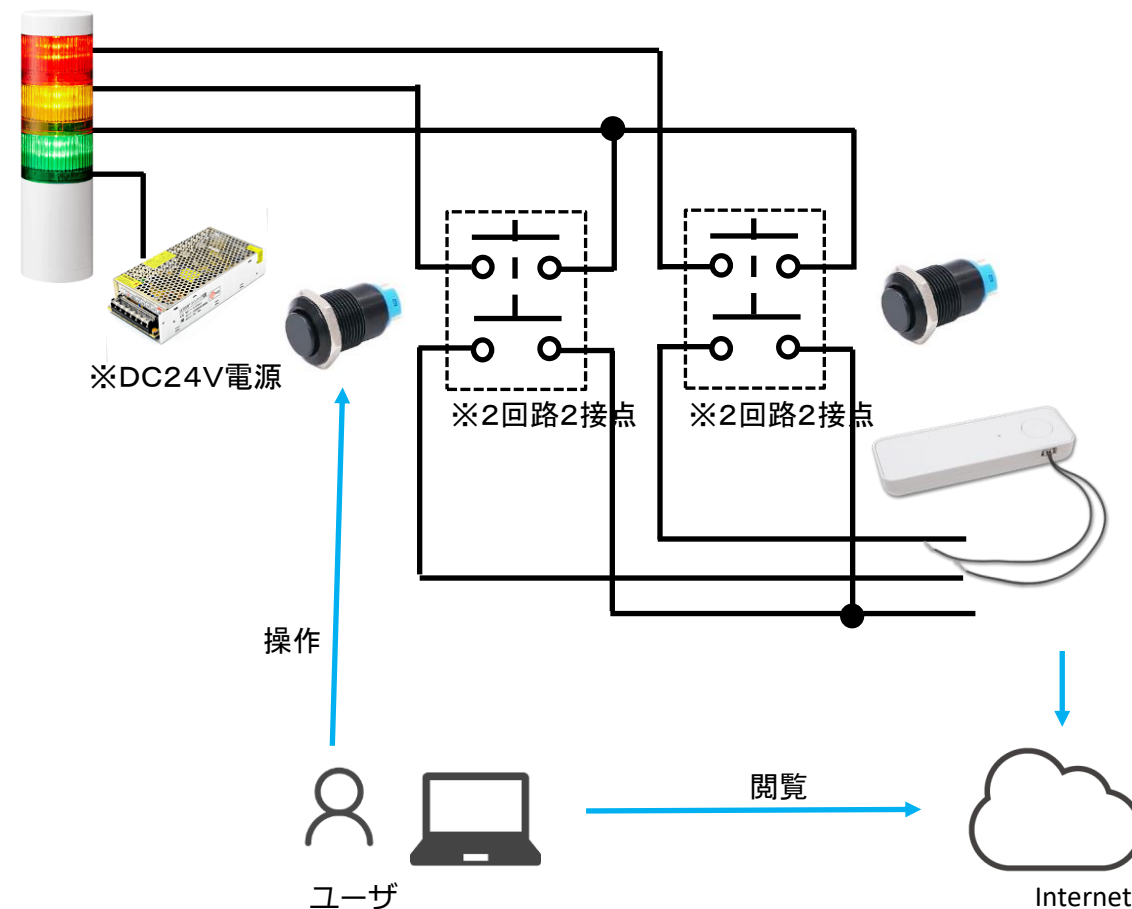
- ボタンを押してメールの送信機能の確認ができれば、接点の情報を利用してみる
 - まずは接点(本体から伸びているケーブル)を短絡してみる: 本体のボタンを押したときと同じ動作になる
 - 本体のLEDが橙点滅してデータ送信が行われる



短絡する

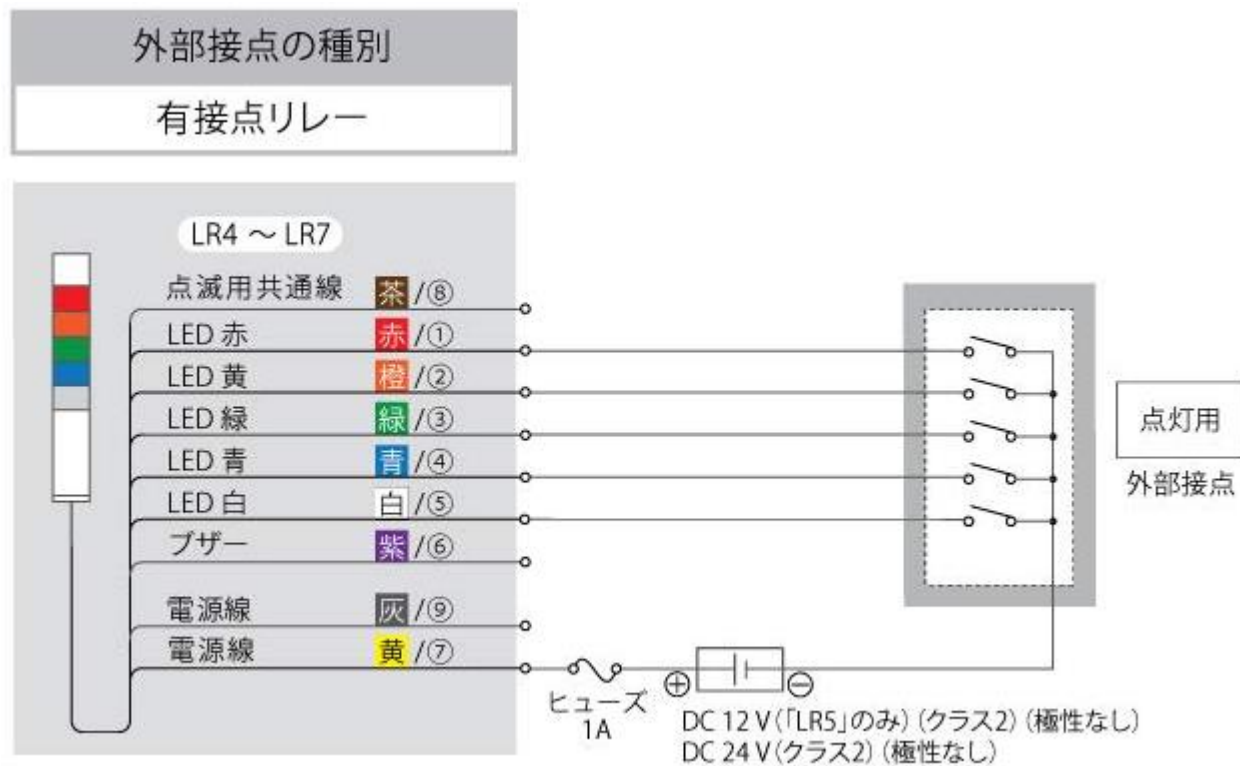
パトランプに 응용してみる

- 実際の機器で応用する方法を考える
 - PCLやリレー盤の機能をスイッチを利用して実現する
 - まずはボタンを押したらパトランプが点灯するよう配線する
 - 下記のステップで配線を行ってみる



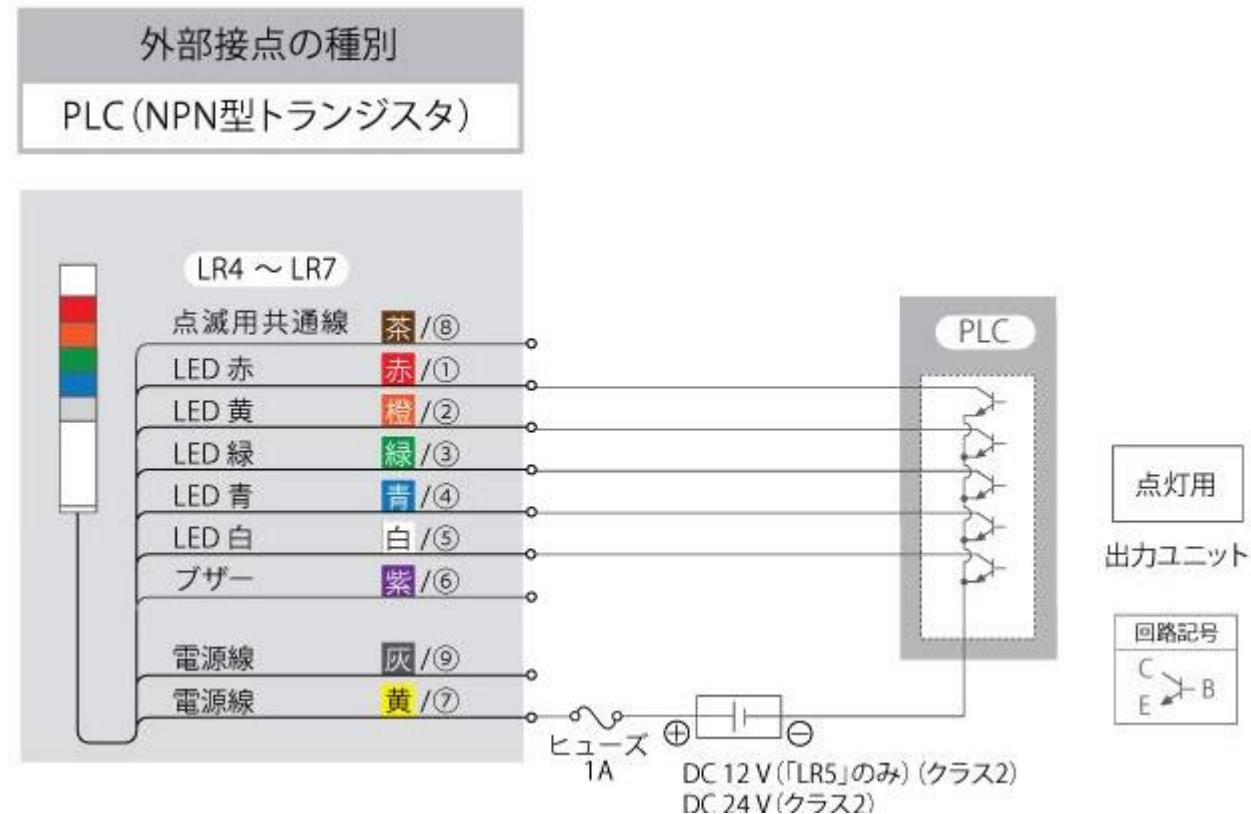
パトランプに応用してみる

- 実際の機器で応用する方法を考える(パトランプを外部からリレーで制御する場合)



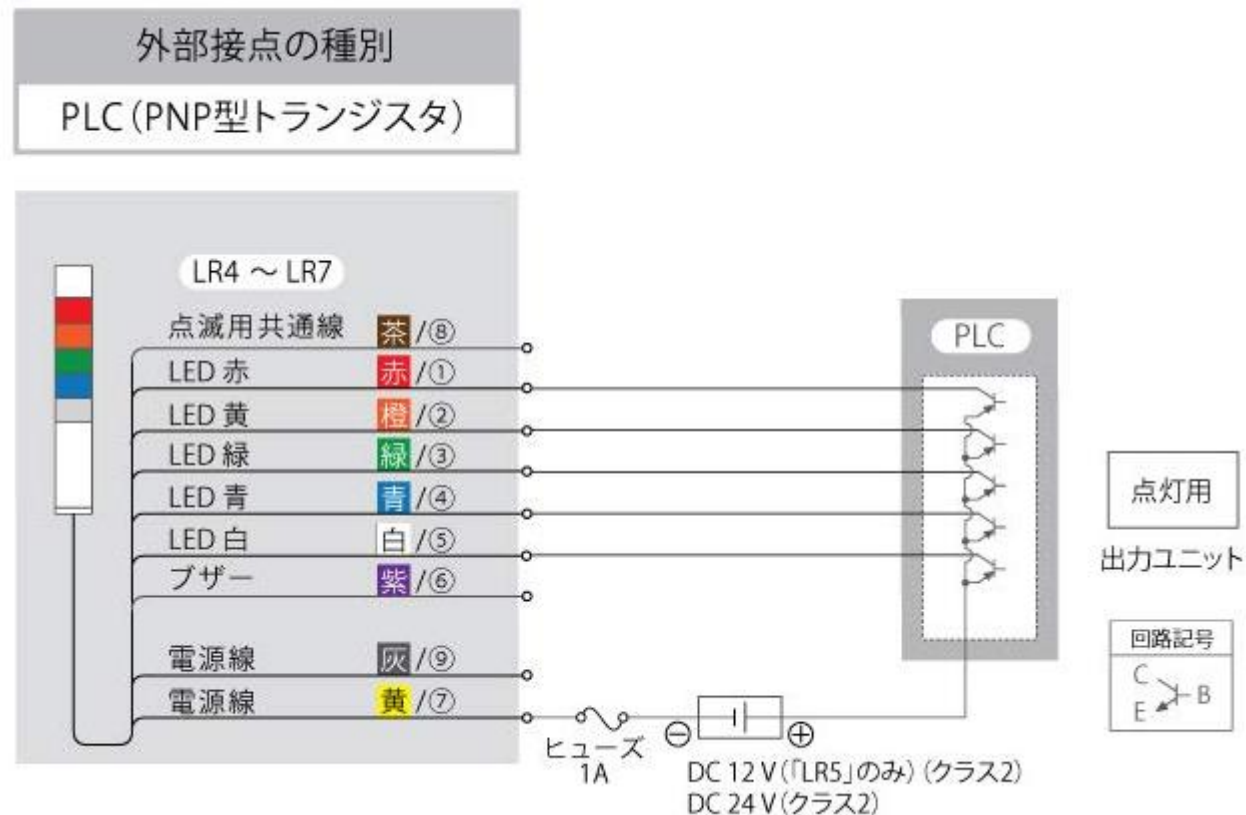
パトランプに応用してみる

- 実際の機器で応用する方法を考える(パトランプを外部からPLCで制御する場合)



パトランプに 응용してみる

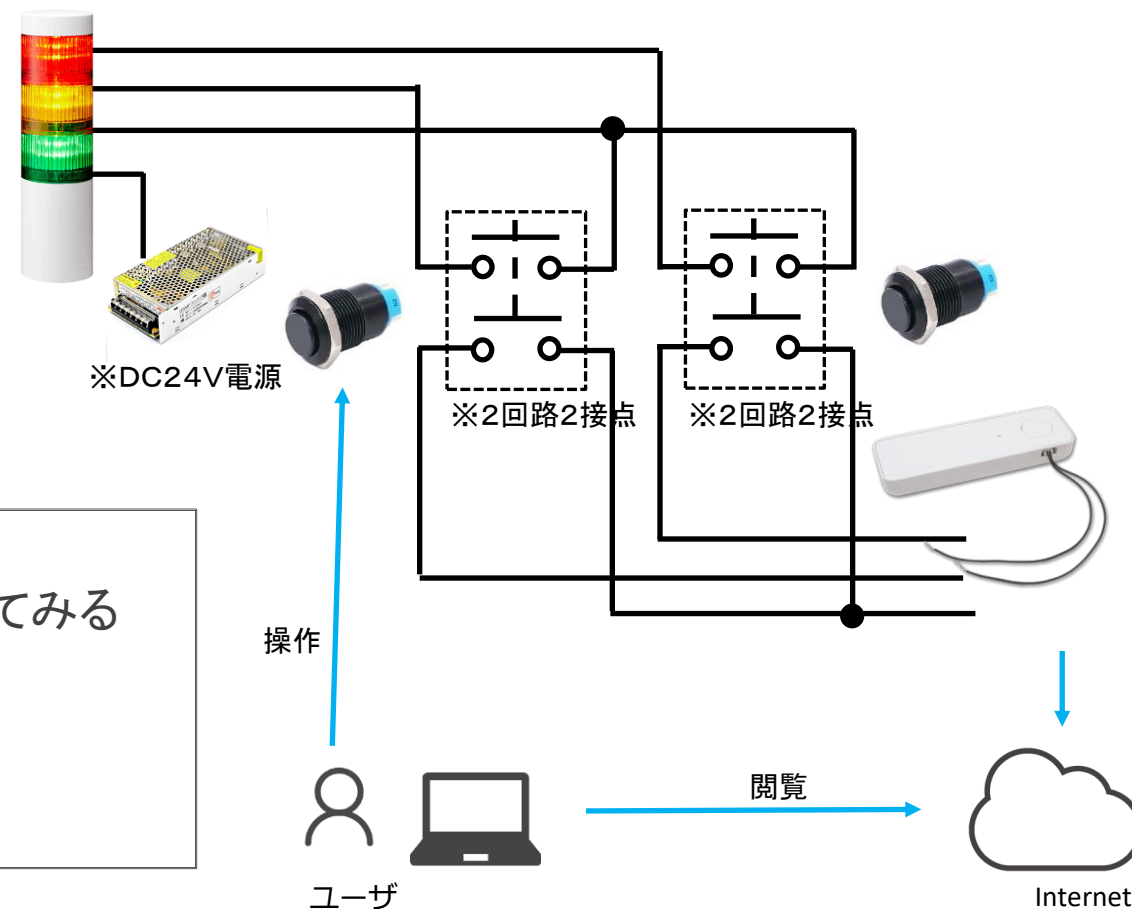
- 実際の機器で応用する方法を考える(パトランプを外部からPLCで制御する場合)



パトランプに 응용してみる

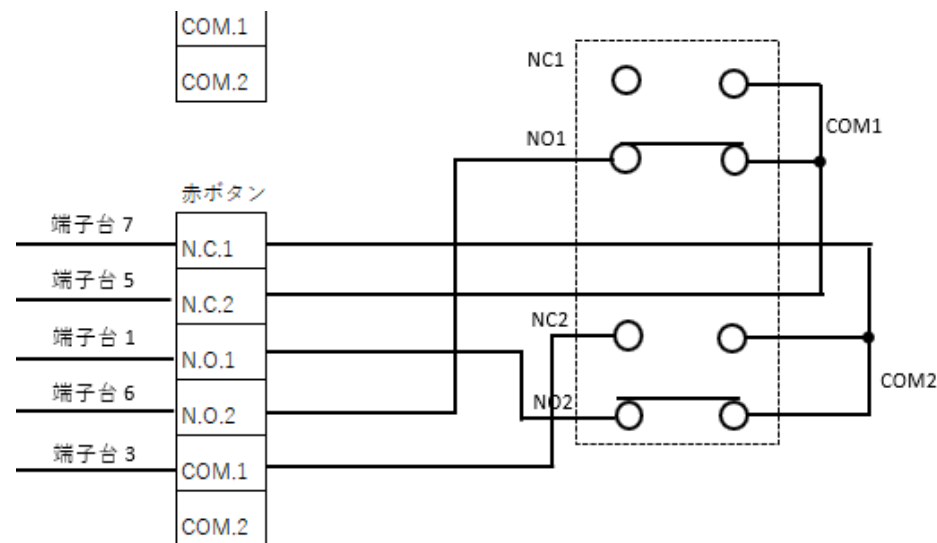
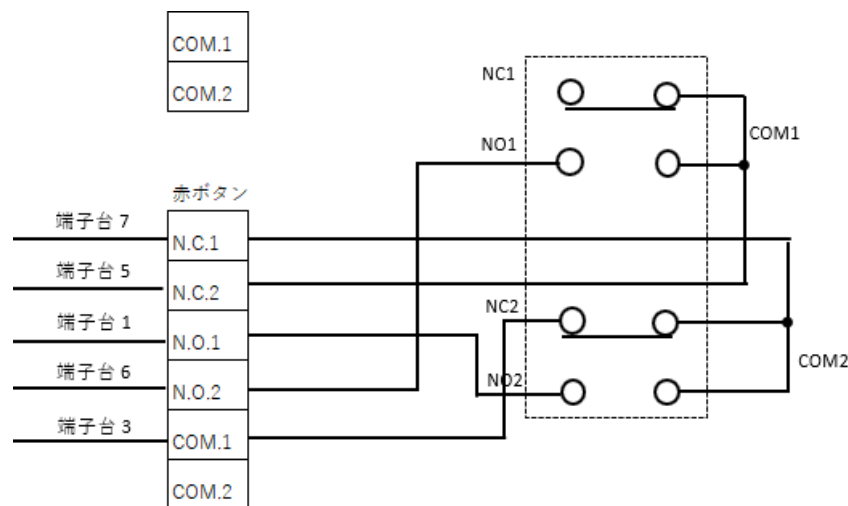
- 実際の機器で応用する方法を考える
 - PCLやリレー盤の機能をスイッチを利用して実現する
 - まずはボタンを押したらパトランプが点灯するよう配線する
 - 下記のステップで配線を行ってみる

- 常時「緑」点灯を前提とし、「赤」点灯時に「緑」消灯してみる
- 「黄」点灯時には「緑」と同時に点灯
- 「赤」点灯時にLTE-Mボタンの端子も短絡する



配線図を参照しながら配線を行う

- 添付の配線図を参考にして、赤ランプ点灯時にLTE-M Buttonからメールが飛ぶようにする
 - LTE-Mボタンの端子を短絡すると本体のボタンを押した場合と同じ挙動となる(メールが出る)
 - どの端子にLTE-M Buttonのコネクタを接続すればよいか考えてみる
- 【ヒント】赤いボタンを押すと赤ランプが点灯する…
 - 左: ボタンを推していない場合 右: ボタンを押した場合



さらなる活用

- 今回はアラート発報時にメール送信したが……
- システムと連携する
- スマートフォンなどにプッシュ送信する
- アラート検出時に所定の処理プログラムを走らせる

などのマイグレーションを(クラウドで)容易に行うことができる

今回のケースはリレーやセンサなどでも
応用することができます



ミニチュアリレー



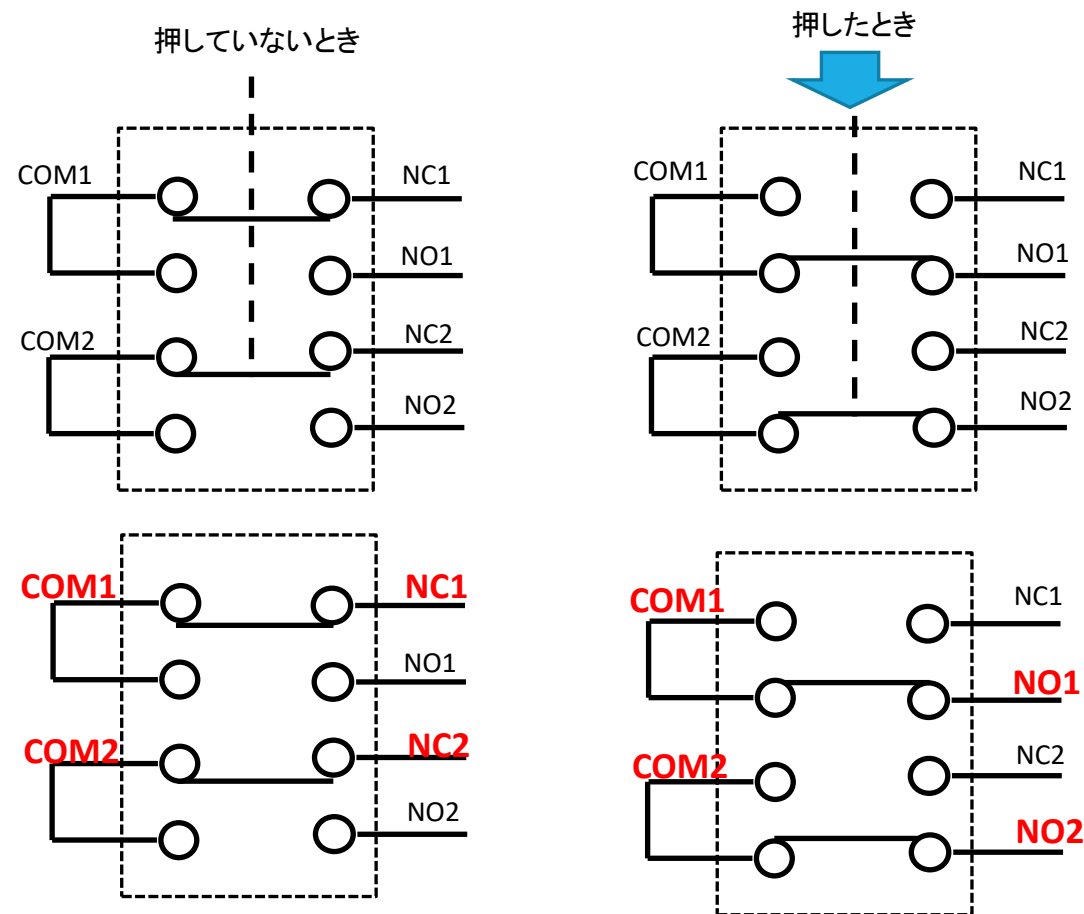
開閉器



マグネット(リード)スイッチ

【参考】2回路2接点スイッチの構造

- ボタンを押下すると2回路同時に操作が行えるスイッチを利用している(2極双投)
 - ボタンOFF時にON, およびボタンON時にONになる反対の動作を行う接点が2つセットになった回路
 - NC1・NO1側／NC2・NO2側の回路が互いに独立しているが、ボタン押下によって連動して切り替わる
 - 右図のように6接点を持つ
- ボタンOFF時は・・・
 - NC1とCOM1, NC2とCOM2が短絡した状態
- ボタンON時は・・・
 - NO1とCOM1, NO2とCOM2が短絡した状態



配線図

別紙