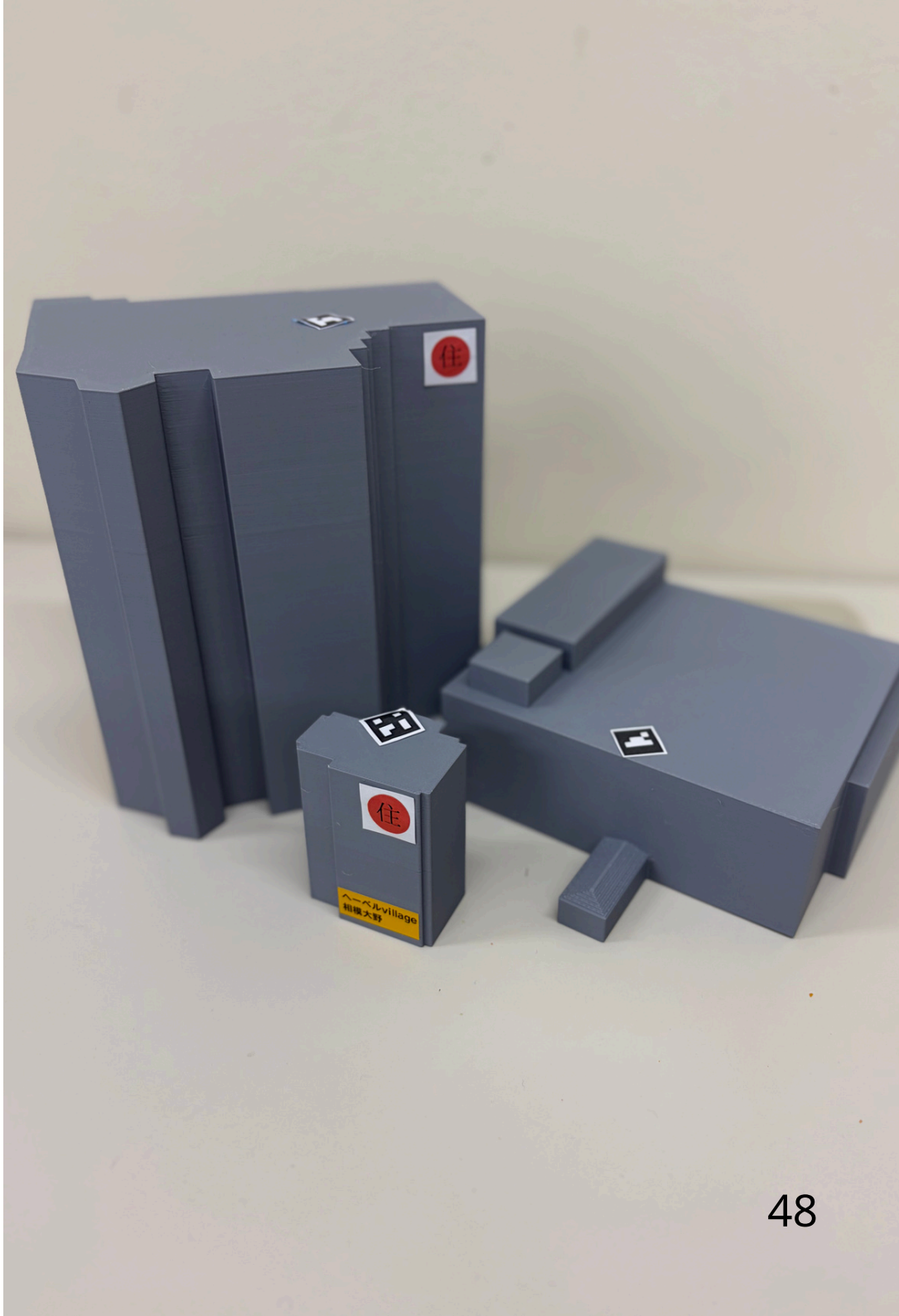
# 物体検知

RFIDとは、タグに記録された情報を非接触で読み書きする自動認識システムです。

中村

石黒



# RFIDとは

RFIDとは、タグに記録された情報を非接触で読み書きする自動認識システムです。
身近な例だと、ユニクロの無人販売システムに使われています。

HTTPS://SMARITE.CO.JP/MEDIA/UNATTENDED_VENDING_SYSTEM#INDEX_ID4


無人販売システムを種類別で解説-ユニクロ

**RFIDタグの例**

タグは小型かつ安価で，任意の場所へ容易に取り付け可能な特徴を持ちます。

## 目的

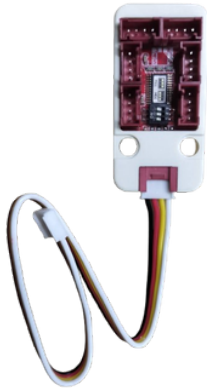
複数のRFIDのタグをリーダーで読み取る。
そのために、RFIDリーダーを複数接続する。

## 問題点

そのままリーダーを単体で接続しても、複数のタグを読み取ることはできません[2]。
それなら、リーダーを複数繋げてそれぞれでタグを読み取ってしまいましょう。


## 用意するもの


・M5Stark Core2本体×1



・PaHUB[2]×1


・RFID リーダーモジュール[3]×4[4]

・RFIDタグ[5]


・USBハブ[6]

・PC[7]

# Arduino IDEのインストールと基礎設定(8)

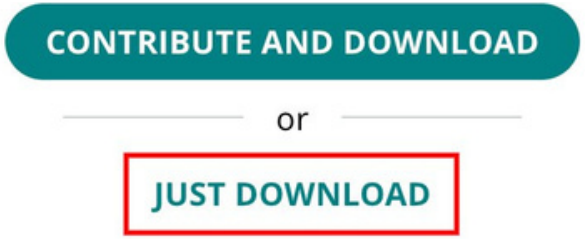
## ダウンロード

まずはARDUINO公式サイトのソフトウェアページからダウンロードしましょう。ページ上部の「DOWNLOADS」から、自分のOSにあったものを選択します。

比較的最近のMACの場合は「MACOS APPLE SILLICON...」で問題ありませんが、INTEL製CPU搭載のMACの場合は「MACOS INTEL...」をダウンロードします。

ダウンロード

**寄付確認画面**



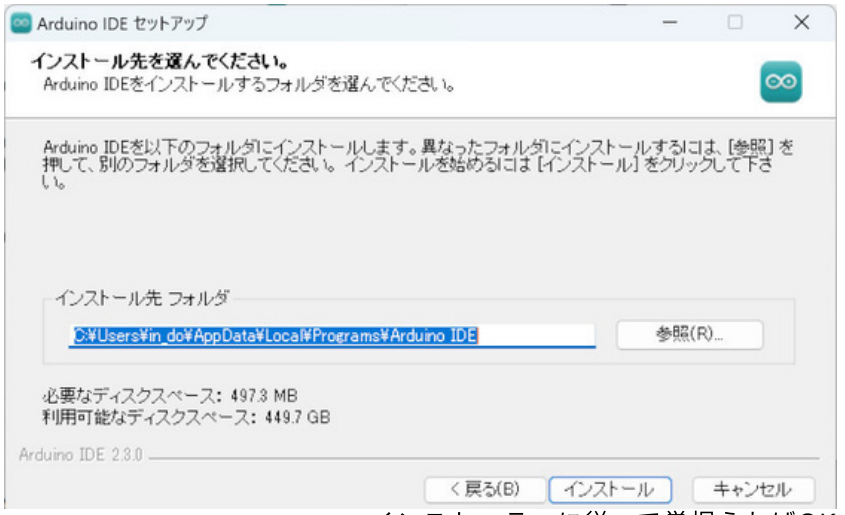
**メールマガジン登録画面**

寄付とメールマガジン登録の画面が出るので、不要な場合はいずれも「JUST DOWNLOAD」をクリックしてダウンロードします。

## インストール（Windows）

ダウンロードしたEXEファイルをダブルクリックするとインストーラーが起動するので、手順に従ってインストールしましょう。「同意する」、「次へ」、「インストール」と順にクリックしていくだけでOKです。

インストール完了後は自動的にARDUINO IDEが起動します。初回起動時は、USBドライバーなどのダウンロード、インストールが自動で行われるので、許可するようにしてください。



インストーラーに従って巣据えればOK

## インストール（Mac）

ダウンロードしたDMGファイルをダブルクリックすると以下のようなダイアログボックスが開きます。一般的なアプリと同様、ARDUINO IDEアイコンを右のAPPLICATIONSにドラッグ&ドロップすればインストール完了です。

下部ドックのLAUNCHPADからARDUINO IDEを起動できます。

## インストール（Linux）

ダウンロードしたAPPIMAGEファイルを適当なディレクトリに移動しておきます。ダウンロードディレクトリでも使用できますが、他のダウンロードしたファイルと混在していると分かりにくいと思います。

その後、APPIMAGEファイルに実行権限を付与します。
APPIMAGEファイルのある場所でターミナルを開き、以下のコマンドを実行します。なお、バージョン番号部分は実際のファイル名に合わせてください。

```
chmod 755 arduino-ide_2.3.0_Linux_64bit.AppImage
```

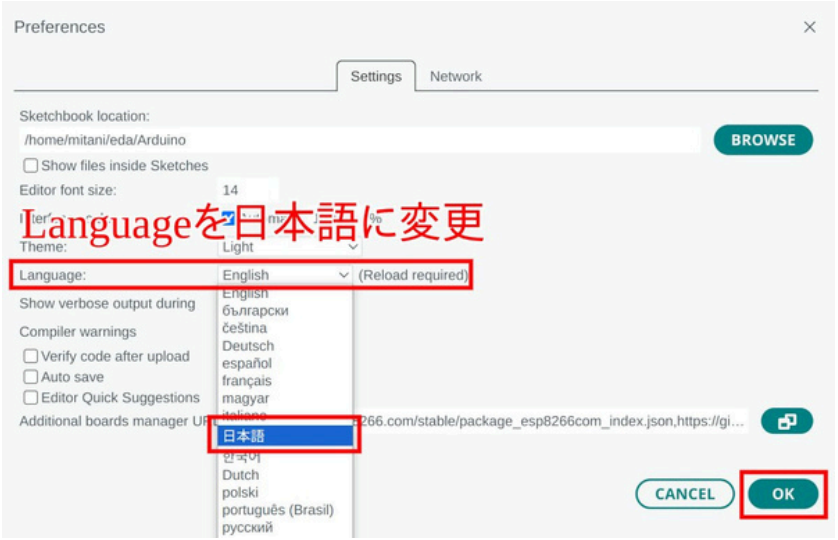
APPIMAGEをダブルクリックしてARDUINO IDEを起動できます。なお、うまく行かない場合は以下のようにコマンドでも起動できます。

```
./arduino-ide_2.3.0_Linux_64bit.AppImage
```

## 日本語表示

初期状態で英語になっている場合は、以下の手順で日本語に変更できます。
ARDUINO IDEメニューの「FILE -> PREFERENCES」をクリックして設定画面を開きます。(MACの場合は上部メニューの「ARDUINO IDE -> PREFERENCES」)
LANGUAGEを日本語に変更し、右下のOKボタンを押します。

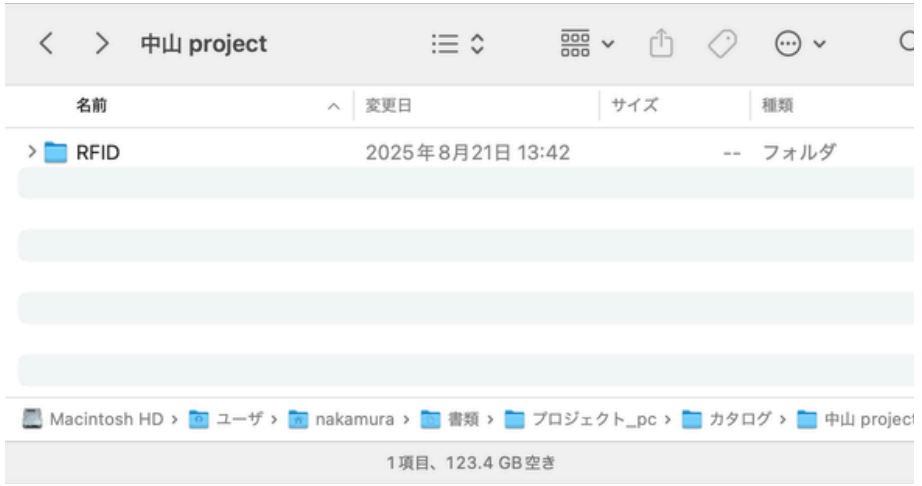
以上でARDUINO IDEのインストールと設定は完了です。



## 作業フォルダの作成

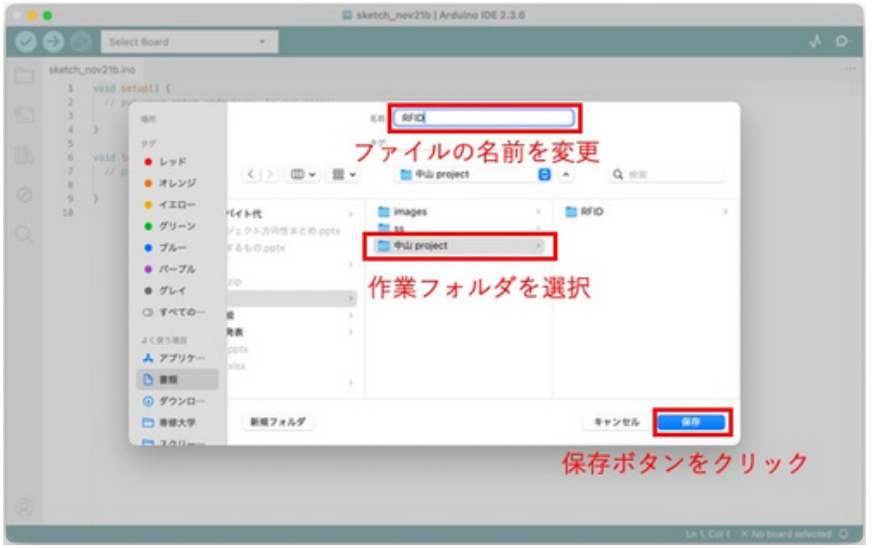
デスクトップ(自分の作業しやすい場所)に任意のフォルダを作ります。

そのフォルダの名前をわかりやすい名前にしましょう。(写真では中山PROJECT)
その中にARDUINO IDEで作成したプログラムを保存していきます。




## 作業フォルダに保存

あらかじめプログラムを作業フォルダに保存しておきましょう。ARDUINO IDEを起動すると、すでに新しいスケッチが生成されているはずです。

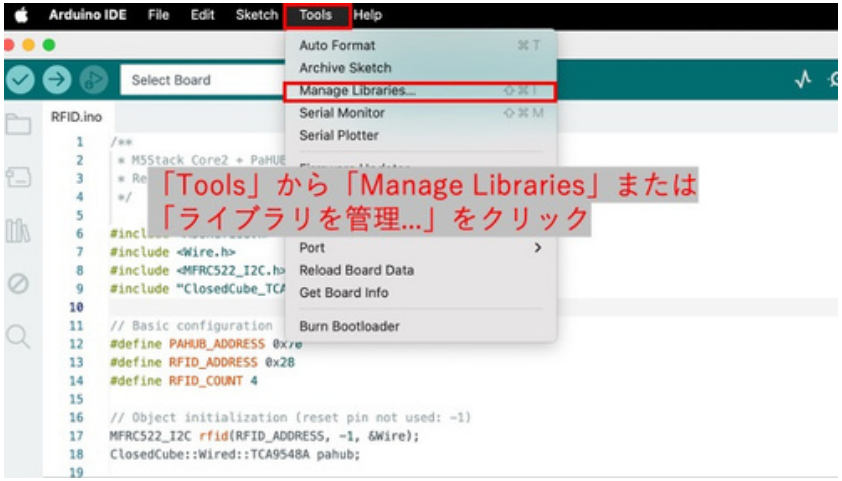




## ライブラリインストール

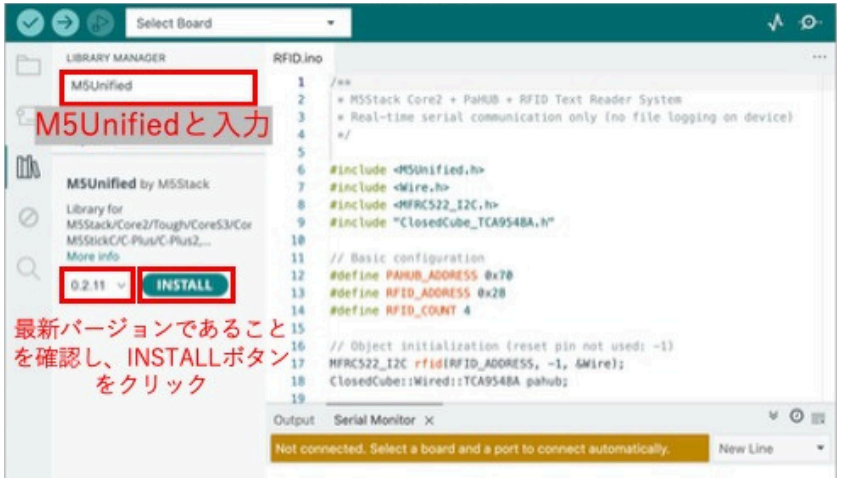

### コード貼り付け

保存ができたら、コードを記述してください。プログラムコードは長文になるためページの最下部に記載します。そこからコピーしてARDUINOにペーストしてください。

## インストール

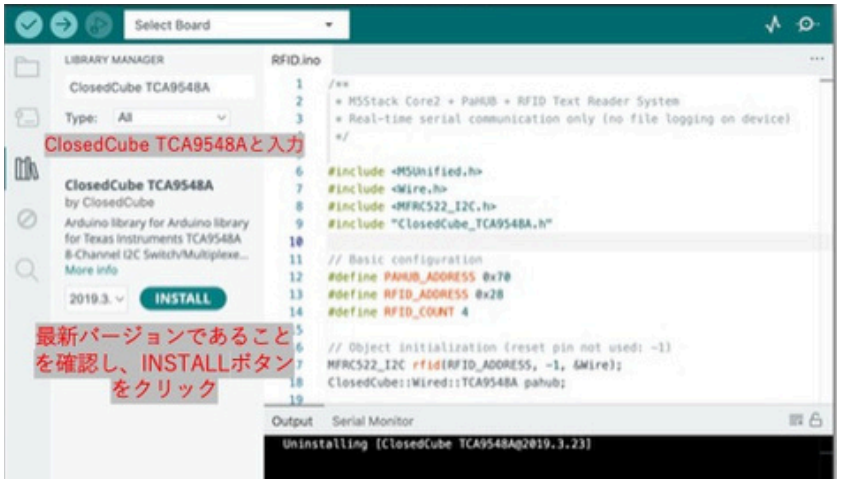



次に必要なライブラリを順番にインストールしていきます。
まずはライブラリマネージャーというツールを開きます。




ツールを開いたら、M5UNIFIEDと検索し、BY M5STACKの
最新版をインストールします。


次に、MFRC522_I2Cと検索し、BY KKLOESENERの最新版
をインストールします。



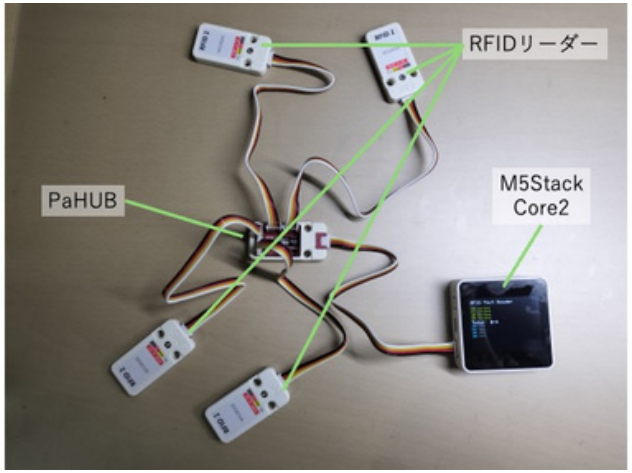

次に、CLOSEDCUBE TCA9548Aと検索し、インストールし
ます。

## システム設定


次は用意したマイコン、PAHUB、リーダー、USBハブを準備してください。まずは、マイコンをPCと接続します。


次にマイコンとPAHUBを接続します。

---

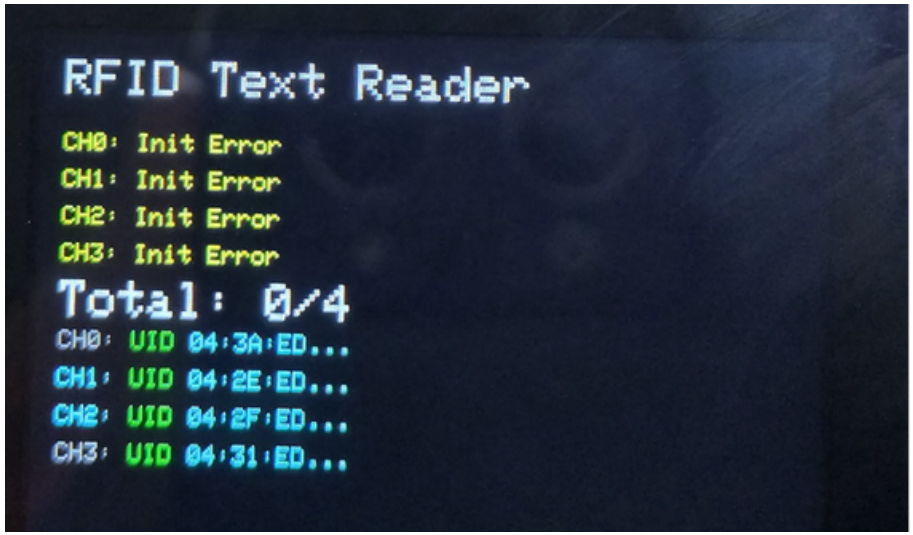



同じ要領でリーダー4つをPAHUBの0〜3と書かれているところにそれぞれ接続してください。以下が全てを接続した際の全体像です。

## ボード設定

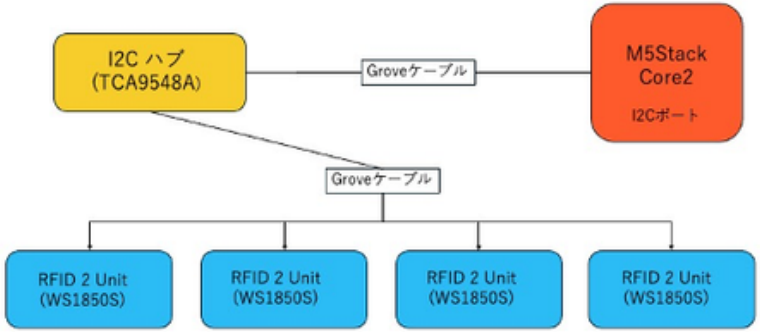
接続したら、ボードが設定できます。「SELECT BOARD」の欄に、「LILYGO T-DISPLAY」と出るので、それをクリックしてください。


設定できたら、コンパイルしてみましょう。左上の右矢印（→）ボタンをクリックしてください。
もし予期せぬエラーが発生しましたら、コードとエラー文をCLAUDE.AIに読み込ませると細かい修正にも対応できます。活用してみてください。正常に読み込まれればディスプレイに以下の表示がされ、タグを読み取れるようになります。



## システム動作イメージ

・PAHUBで複数接続を可能にする[9]。
・メインマイコン（M5STACK）がPAHUBにチャネル0をONに指示
→そのチャネルに接続されたRFIDを問い合わせ
→次にチャネル1に切り替えて別のRFIDを問い合わせという順番で巡回する。
・今回は、4チャンネルのリーダーを接続し、並列的にタグを読み取る。




**デバイス構造**

## 参照

(1)アドレス衝突が起きるため
(2)I2Cマルチプレクサ、TCA9548A相当の8CH/4CHモジュール
(3)I2C対応
(4)今回は4台想定
(5)推奨：13.56MHZ NFCタグ（ISO 14443A準拠）
(6)M5STACK と PC 接続用
(7)シリアルログ受信・CSV 保存用
(8)HTTPS://WWW.INDOORCORGIELEC.COM/RESOURCES/ARDUINOIDE%E8%A8%AD%E5%AE%9A/ARDUINO-IDE%E3%81%AE%E3%82%A4%E3%83%B3%E3%82%B9%E3%83%88%E3%83%BC%E3%83%AB%E3%81%A8%E8%A8%AD%E5%AE%9A/
(9)I2Cアドレスの競合を回避できる拡張モジュールで、複数リーダーを切り替えて利用することが可能

## プログラムコード

こちらのQRコードを読み込むとプラグラムのコードが見れます。

# 制作ガイド：物体検知・WEB可視化編

本ガイドでは、物理的な模型の動きをデジタル空間に同期させる「デジタルツイン」の構築手順を解説します。

## 0 事前準備：ファイル構成の確認

開発を始める前に、プログラムとデータの「居場所」を整理します。本システムでは、データの加工状態や役割に合わせてフォルダを厳密に分けて管理します。(GITHUBに内容があります。)

```
.
├── SERVER.PY          # 全体の司令塔（AR検知 & 通信）
├── SCRIPTS/           # ★データ変換ツール群
| ├── CONVERT_BUILDINGS.PY  # GMLから建物JSONを生成
| └── CONVERT_ROADS.PY     # GMLから道路JSONを生成
├── DATA/              # 地理データ集約フォルダ
| ├── RAW/             # PLATEAUの加工前GMLファイル
| ├── PROCESSED/       # 変換済みの軽量JSON（WEB表示用）
| | ├── BUILDINGS_YUEN.JSON # 背景（動かさない建物）
| | ├── BUILDINGS_MARKERS_ONLY.JSON # 模型（動かす建物）
| | └── ROADS_YUEN.JSON  # 道路データ
| └── MASTER/          # システム定義ファイル
|   └── BUILDING_MARKER_MAPPING.JSON # マーカーと建物の対応表
├── CONFIG/            # システム設定（自動生成）
| └── CALIBRATION_SETTINGS.JSON # カメラ補正値
└── TEMPLATES/
   └── INDEX.HTML      # 3D地図表示画面
```

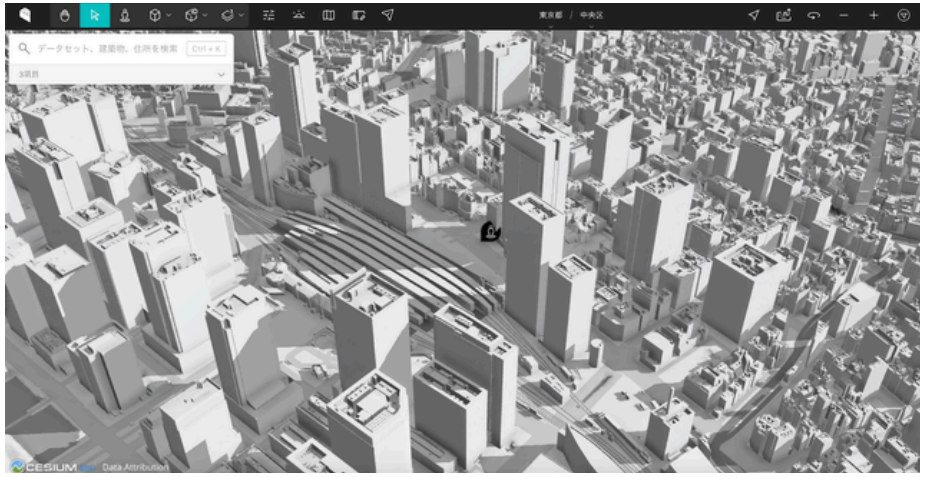
# 1 データの仕分けとIDの特定

本システムの核心は、PLATEAUのデータを**「動かすもの（模型）」と「動かさないもの（背景）」**に切り分ける点にあります。

## 1.1　建物ID（gml_id）の特定

PLATEAU VIEW にアクセスします。

https://plateauview.mlit.go.jp/



左上の検索窓に欲しい建物がある地区（今回は多摩区）を入力し、
建築物モデルを選択してください。

表示された建物モデルの中から、動かしたい建物を選択してください。右手に表示された詳細情報からIDの部分をコピーしてメモ帳などに控えます。

別の街から持ってくる建物（名所など）がある場合も、同様にIDを控えます。

# 2 データの取得とツールによる変換

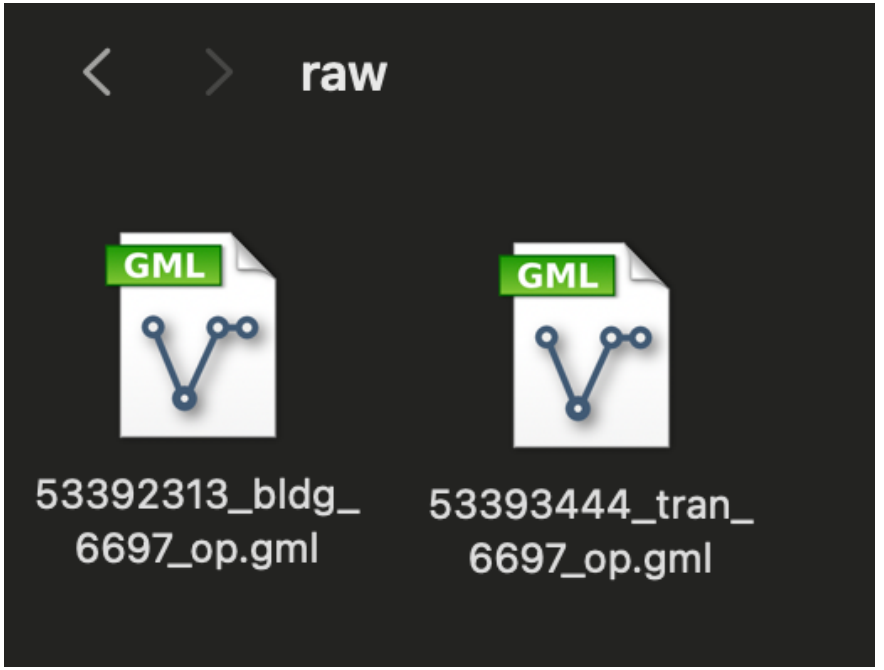
仕分け情報を元に、生データをシステム専用のJSON形式へ加工します。

## 2.1 PLATEAUデータの配置

1．PLATEAUのオープンデータからCITYGMLをダウンロードします。

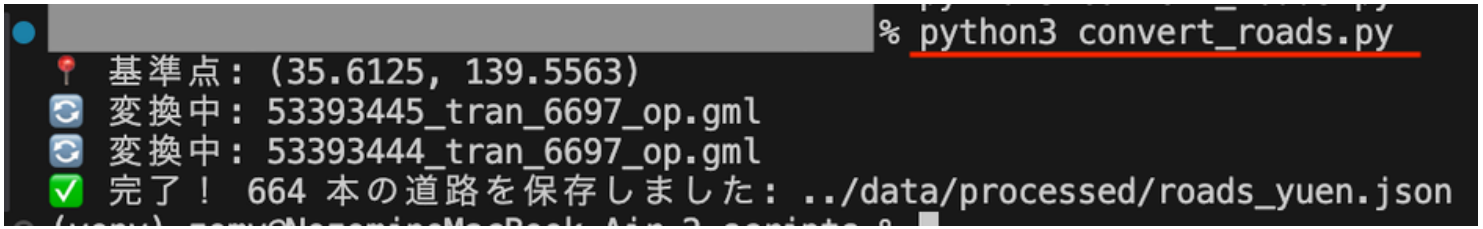
https://plateauview.mlit.go.jp/

2．表示したい街のタイルが入っている.gml ファイルを
data/raw/ へ格納します。
地区によって異なりますが、川崎市のデータではbldgと書いてあるファイルには建物データが入っており、tranと書かれているファイルには道路データが入っています。

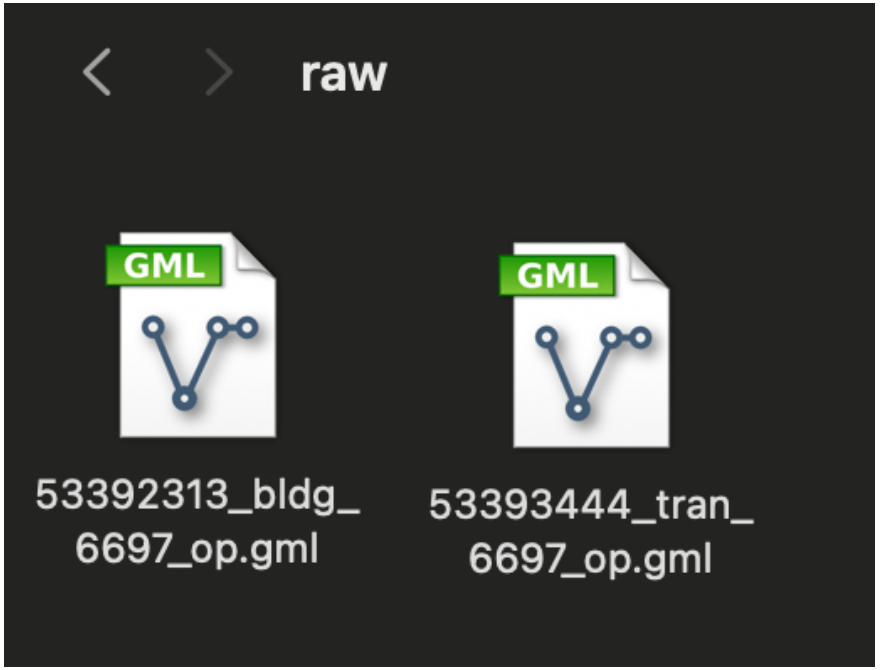



## 2.2 道路データの変換

1．python scripts/convert_roads.py を実行します。

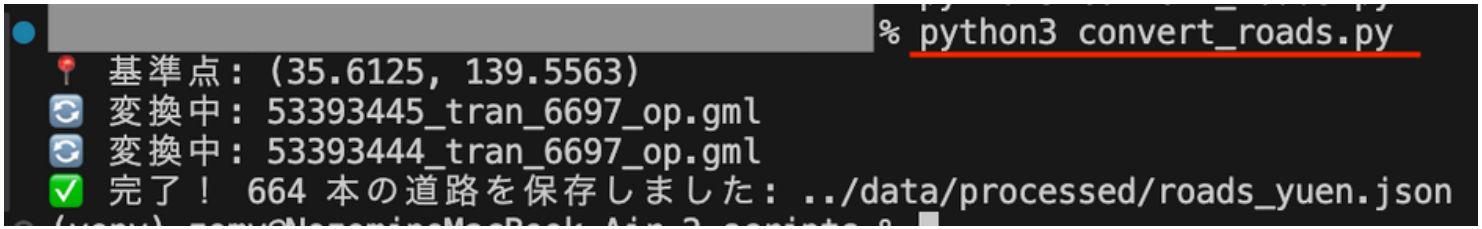


2．表示したい街のタイルが入っている.gml ファイルを
data/raw/ へ格納します。
地区によって異なりますが、川崎市のデータではbldgと書いてあるファイルには建物データが入っており、tranと書かれているファイルには道路データが入っています。

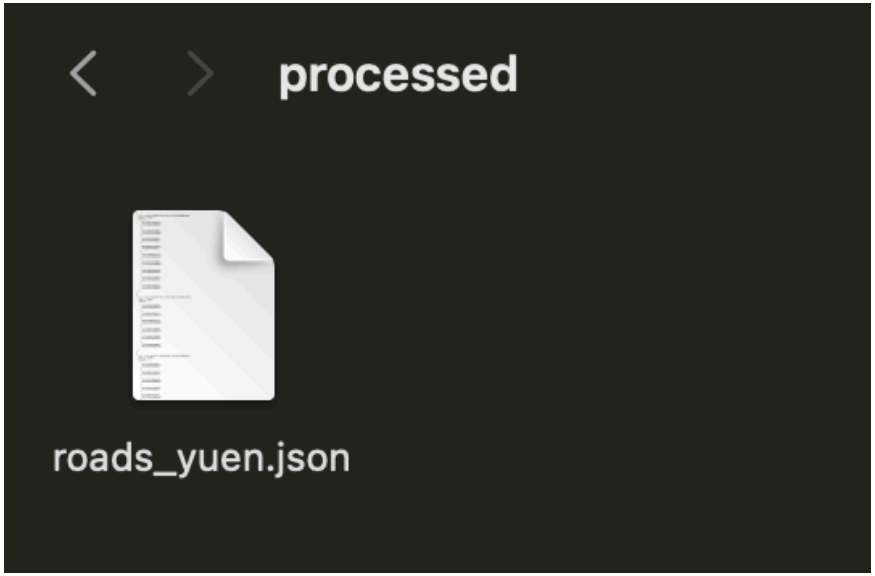



## 2.2 道路データの変換

1．python scripts/convert_roads.py を実行します。



2．中心点から半径800m圏内の道路網が
data/processed/roads_yuen.json として出力されます。

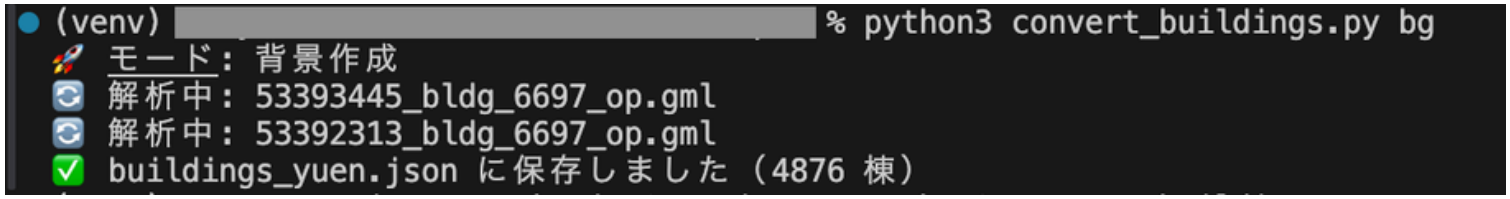



## 2.3 道路データの「分離」変換

convert_buildings.py を2回実行し、背景と模型を物理的に別ファイルに分けます。

- 手順A：背景用 (buildings_yuen.json) の作成

  a. スクリプト内の MOVABLE_IDS リストにメモしたIDを記入します。
  b. python convert_buildings.py bg これを実行します。
  c. 動かしたい建物が「消えた」背景データが生成されます。

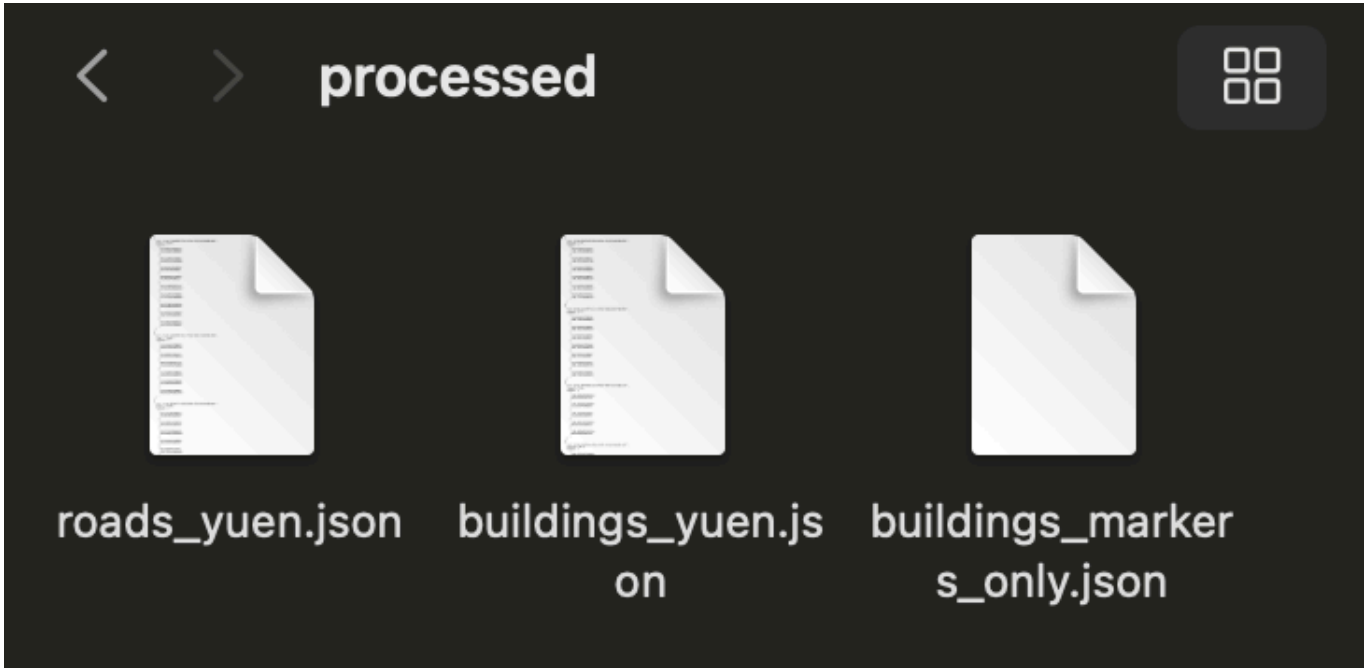


- 手順B：模型用 (buildings_markers_only.json) の作成

  a. 今度はpython convert_buildings.py mv 実行します。
  b. 特定の建物だけが、共通原点に基づき書き出されます。

```
(venv)                                          % python3 convert_buildings.py mv
🚀 モード: 模型抽出
🔄 解析中: 53393445_bldg_6697_op.gml
🔄 解析中: 53392313_bldg_6697_op.gml
✅ buildings_markers_only.json に保存しました（0 棟）
```

２の手順が全て終了し、data/processed の中にはroads_yuen.json,buildings_yuen.json,buildings_markers_only.jsonの３つのファイルが入っていればOKです。



# 3 物理セットアップとシステム起動

## 3.1 マーカーの紐付け

1. data/master/building_marker_mapping.json を開きます。

2. ARマーカーIDと、抽出した建物の gml_id を対応させて記述します。記述する際は写真のようなルールで記述してください。

```
"31": {建物と対応させたいARマーカーのIDを書いてください
  "building_id": "bldg_d779da7a-dd44-4806-ac8b-4774015a3b51",1で取得した建物IDを書いてください。
  "name": "ライフ",建物につけたい名前を書いてください。
  "usage": "122",建物の用途コードを書いてください。
  "area": 2045.77,建物の延べ床面積を書いてください。
  "jsonfile": "buildings_markers_only.json"おまじないのようなものです。すべての建物に書いてください。
},
```

## 3.2 サーバー起動と座標補正

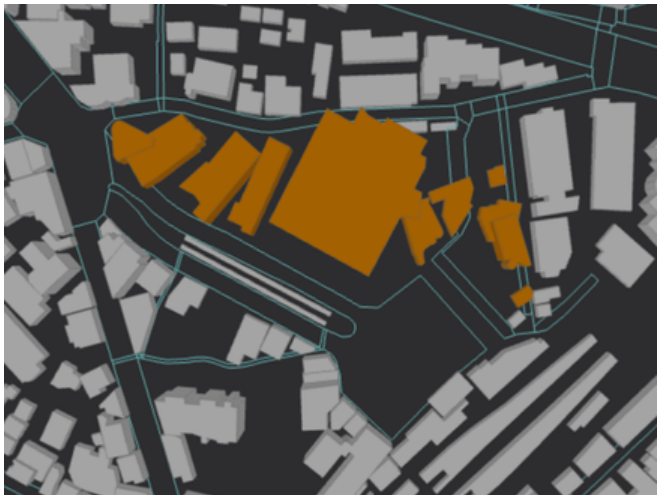
python server.py を実行し、カメラ映像を立ち上げます。

# 4 Web画面でのリアルタイム分析

## 4.1 3D地図の描画と同期

1．ブラウザで http://localhost:8000 を開きます。


2．ARマーカーつきの模型をカメラに映すと、対応したデジタル上の3D建物が滑らかに追従します。



# 5 資料とコードのダウンロード

本ガイドで使用したソースコードや、環境構築のための手順は、以下のGitHubリポジトリから確認・ダウンロードが可能です。

- GitHub URL/QRコード:
  https://github.com/nakayama-pro-2025/Tangible-GIS-NakayamaPro-2025