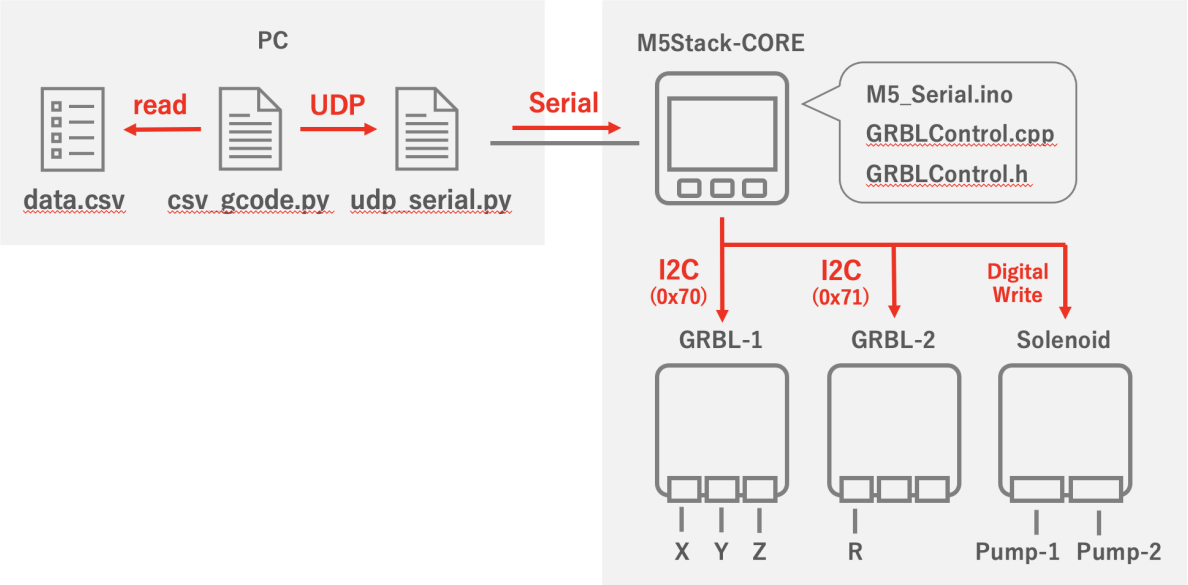
ChipMounter System

全体のシステム概要図



概要

- 制御するものは5つ
  （X-StepMotor, Y-StepMotor, Z-StepMotor, 回転軸Motor, 吸ポンプ, 吐ポンプ）
- CSVで指定した部品位置に部品を配置する
- 作成したGcodeを、udp_serial.py → M5_Serial.ino → GRBLの順に流すことでステップモータ等を制御している

各コードの役割

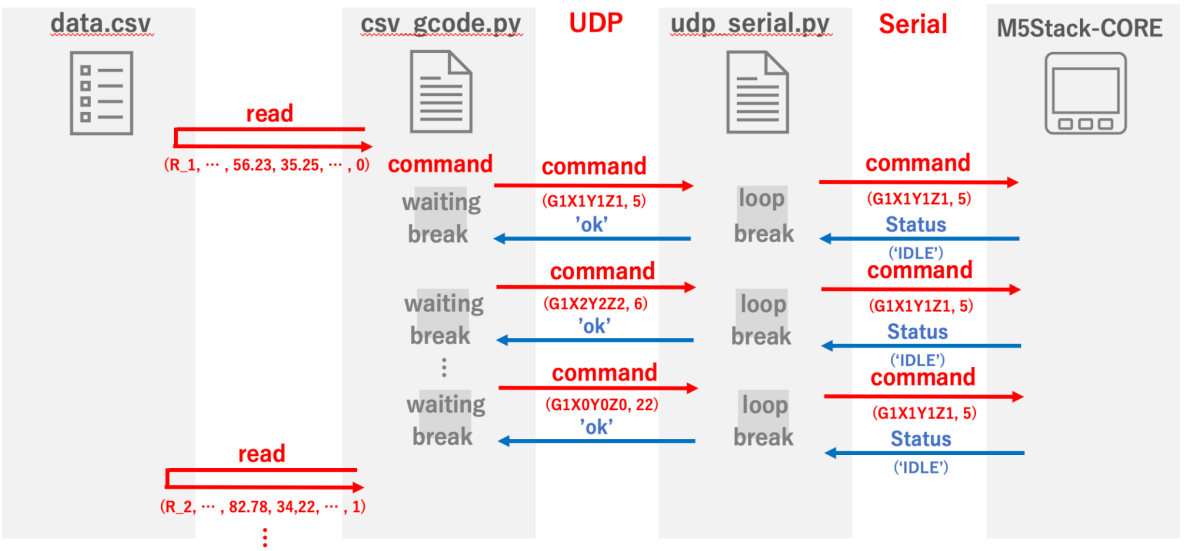
- csv_gcode.py
  = POSファイル（CSV形式）から、部品位置を抽出して、Gcodeを作成
  = 作成したGcodeをUDP通信で送信
- udp_serial.py
  = M5Stackとのシリアルを常時開通させ、Gcodeの送信、GRBLステータスの受信
- M5_Serial.ino
  = Gcodeの受信、適切なモジュール（GRBL-1, GRBL-2, Soleoid）に送信
  = 画面表示などのUI管理

データ処理の流れ

csv_gcodeの処理

処理の流れ

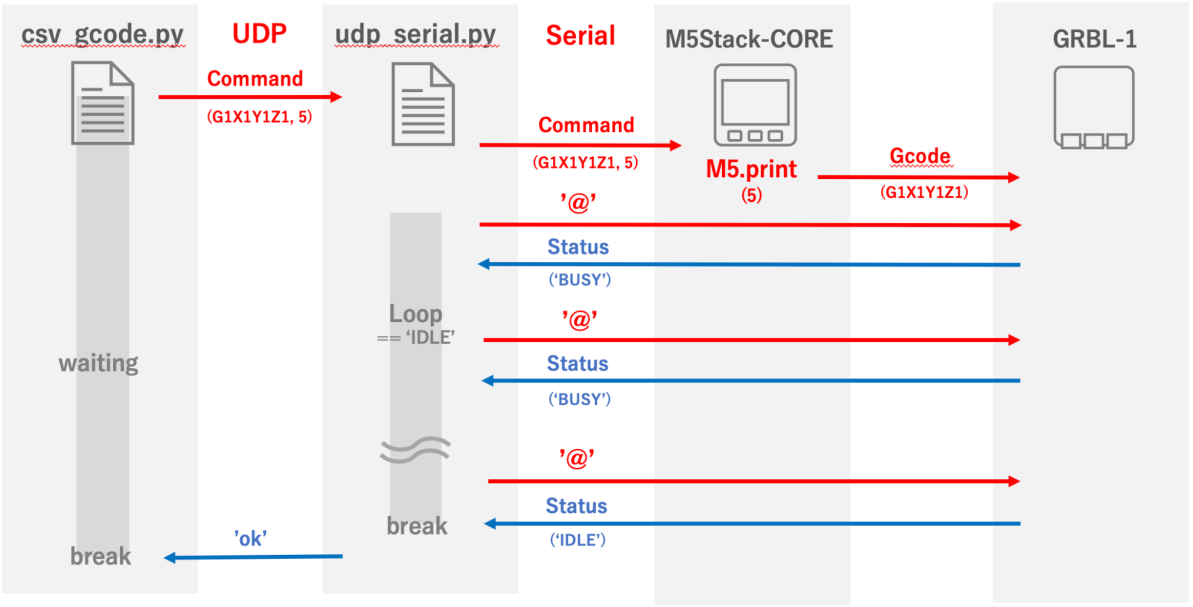
- CSVファイルから、POSデータ（部品の配置位置など）を読み取る
- POSデータからcommandを作成する
- 作成したcommandをudp_serial.py（UDP通信）、M5Stack-CORE（Serial通信）で受け渡す
- 処理成功の合図（'ok'）を待ってから、次のcommandを送信
- commandを全部送信し切ったら、次の部品に関するPOSデータを取得しにいく



udp_serial.pyの処理

処理の流れ

- csv_gcode.pyからcommandを受信する
- 受信したcommandをそのままM5Stack-COREに受け渡す
- スタータス確認コマンド（'@'）を送信して、現在のステータスを返してもらう
- ステータス'IDLE'を確認しだい、処理成功の合図として'ok'を送信する



M5_Serial.inoの処理

- 基本的には受信したcommandを適切な形にして各モジュールに送信する
- commandの内容で送信するべきモジュールの判断をしている
- commandには連番で番号がついているので、番号を取得することで進捗が分かる
- 進捗を元に画面に進捗率を表示している