Wifi 搭載マイコンとデータベース PC を用いた スケーラブルな計測システムの開発

○真野篤志 A)

A) 計測・制御技術支援室　シンクロトロン光技術グループ

概要

WiFi(IEEE 802.11.b, g, n)通信機能を搭載したマイクロコンピュータ(マイコン)である Espressif System 社製 ESP シリーズを利用し、開発環境には ArduinoIDE を用いて、マイコンが直接リレーショナルデータベースにアクセスし、測定値を記録していくというシンプル且つ拡張自由度の高い測定システムの原型を開発した。この開発したシステムにおける、基本構成や、現段階での立上げ方法などについて報告する。

1　背景

近年の無線通信システムの発展や普及は著しく、関連機器・素子の低廉化も急速に進んでいる。特に 2015 年以降に Espressif System 社より順次発売されたマイクロコンピュータ(マイコン)モジュール ESP シリーズは、WiFi(IEEE 802.11.b, g, n)通信機能を内蔵して、モジュール単品で約 400 円、開発ボードでも約 2000 円と非常に安価なものである。また、マイコンでは内蔵ソフトウェアの開発が必須であるが、その開発環境として、メーカー純正の ESP-IDF のみならず、ArduinoIDE や MicroPython といった既に広く普及済みのものにも対応していたので、他のマイコン開発経験を生かせるという画期的な物であった。これにより高機能な無線通信を利用したマイコン開発が非常に手軽にできるようになった。

特に利用者数の多い Arduino IDE に対応している意味は大きく、Arduino 公式リポジトリに蓄積された膨大な機能を用意に導入・利用できることを意味する。本報告では、そのうちの 1 つである「MySQL Connector Arduino」ライブラリを用いることで、システムを開発することができた。

2　データベースを利用する意義

2.1　データベースとは

データベースとは、検索や蓄積を目的として所定の形式に整理された情報の集まり、または、その管理システムを指す。データベースには、情報の管理方式によって階層型やカード型といった様々な種類があるが、一般にデータベースといった場合はリレーショナル型と呼ばれる方式を指すことが多い。本報告においても、データベースはリレーショナル型を前提として記載している。

2.2　リレーショナルデータベースの特徴

リレーショナルデータベースは Microsoft Excel などの表計算ソフトウェアと非常によく似たデータ構造(表 1.)を持っているが、全くことなるものであり、混乱を招くことも多い。大きな違いは、表計算ソフトウェアでシートに相当するテーブルにおいて、入力可能な内容・書式などが厳密に固定されている点と、様々なデータの処理はテーブルそのもので実施するので

表 1. 表計算ソフトウェアとデータベースの退避

表計算	データベース
ファイル	スキーマ (データベース名)
シート	テーブル
列	カラム(列)
行	レコード(行)

はなく、クエリとよばれる専用の処理機構を利用して行う点が挙げられる。

リレーショナルデータベースではリレーションと呼ばれる機能が最大の特徴で、指定するカラムの値が同じレコードを結合できる。これにより、複数テーブルに分散したデータの連携が容易にできる。(図 1.)

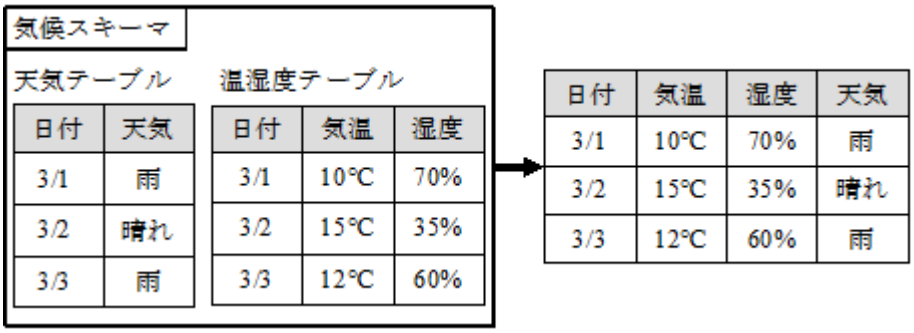

図 1. リレーション機能の例(日付カラム基準で結合)

また、クエリ機能では、グループ化と集計という機能もあり、指定するカラムの値が同じレコードをグループとして分類し、その各グループに属するレコードの平均や合計などの集計を図 2.の様に容易に行うことができる。

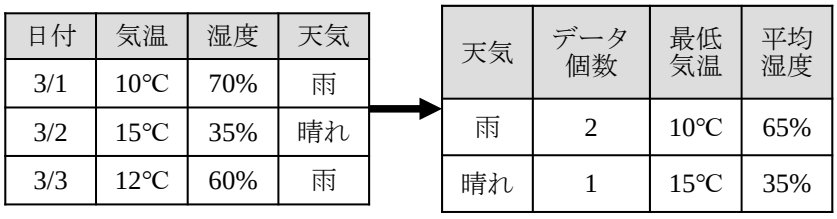

図 2. 集計機能の例(天気カラム基準で気温最低値と湿度平均値を集計)

これらの機能により、リレーショナルデータベースを用いることで、複数のセンサーからの長期間のデータを連携し、集計する分析を容易に実施できるため、データ蓄積にリレーショナルデータベースを採用した。

3　システムの基本構成

測定システムについて、インターネットなどによく掲載されている事例を紹介し、本報告のシステムとの対比を以下に述べる。

3.1　データ分散配置型(マイコンでのデータ保持)

測定システムにおいて、最もシンプルな構成が、データベースサーバーなどデータの集約記録を担うものを置かずに、各マイコンに SD メモリーカードなどを取り付け、そこにデータを記録する方式である。(図 3.)

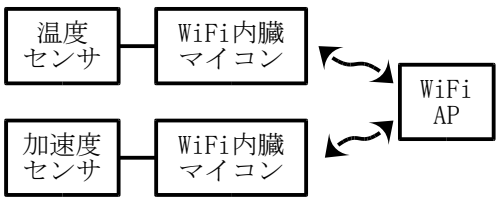

図 3. マイコンへデータ配置

マイコンソフトウェアを開発するだけで済むため、初期の開発負担は非常に軽いものとなる。だが、データへのアクセスの為には、各マイコンの IP アドレスを台帳等で管理する、DHCP サーバーによる特定 IP アドレスの割り振りを行うといった煩雑な管理が必要になる。このため、マイコンがバグや通信経路不良でアクセス不能となった場合に、その情報をきちんと後に伝える工夫をしないと「いないもの」とみなされてしまうという問題が生じる。

また、複数のマイコンに保存されたデータの連携分析を行う場合に、各マイコンにアクセスしてデータを

ダウンロードし、その後にデータ結合処理をして、分析を実施となり、非常に手間がかかる。

3.2　中継サーバー設置型

インターネットにおける測定・記録システムにおいて、最もよく見られるのが、図 4.の様に中継サーバーを設置する方法である。

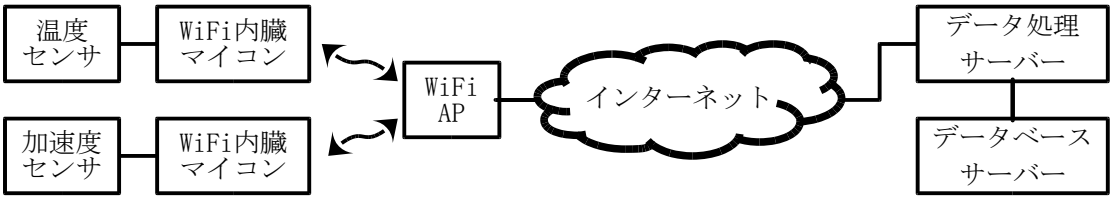

図 4. 中継サーバー設置型システム

データ処理サーバーは Web サーバーを利用する例が多い。マイコン側に Web ブラウザの簡易版と呼べる WebClient ライブラリを利用し、HTTP(Hyper Text Transport Protocol)における POST メソッドにより Web サーバーに測定値を通知する。Web サーバーは、PHP や CGI といったプログラミング機構によって、受信した測定値情報を解析・整形し、データベースへの書き込みを行う。この他に、Telnet を利用し、独自のメッセージ構造で通信を行うパターンもある。

この方式では、マイコンソフトウェアが送信するメッセージ書式とデータ処理サーバーが解析・整形しようとする書式が整合している必要がある。例えば、新たにセンサー種類を追加する場合には、マイコンソフトウェアを開発するだけでなく、データ処理サーバーの改造を行う必要がある。このような複数個所の整合を取りながら、運用と改造を繰り返すと不整合を発生させる事故が起きやすくなる。このタイプのシステム運用においては、中途半端な処理共通化が発生しやすく、新規センサー追加時に共通化部分を書き換えてしまうと、システムが破綻する恐れすらある。

3.3　データベースサーバー直結型

今回、開発したのがこの図 5.のようなタイプのシステムである。

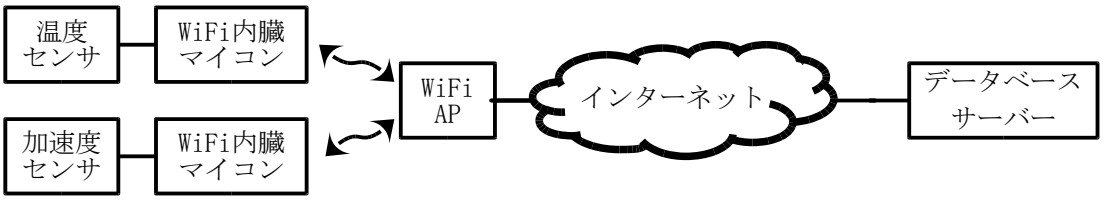

図 5.データベースサーバー直結システム

マイコン自身にデータベースクライアント機能を持たせ、センサーから得られた情報を直接データベースサーバーに書き込みを行う。これにより、センサーの違いについてはマイコンソフトウェア部分の処理内容変更のみで対応可能となり、運用が極めてシンプルにできる。ハードウェア開発を伴うプログラミング教育用途に開発されたマイコン規格である Arduino のおいては、「MySQL Connector Arduino」ライブラリという Arduino のマイコンにデータベースクライアント機能を付加するライブラリが公式リポジトリに登録されており、僅かな GUI 操作で開発環境に組み込むことができる。このライブラリを利用するとネットワークに接続可能な Arduino 対応マイコンであれば、非常に簡単に MySQL，MariaDB，PostgreSQL といった主要なオープンソースのリレーショナルデータベースサーバーに接続、データ読み込みや書き込みといった各種処理を

行うことができる。

4　テーブル自動追加機能

4.1　機能の概要

データベースに記録するうえで、「どのテーブルに書き込むか」というのは重要になる。1 つのデータベースサーバーに複数のマイコンを接続するときに、適切なテーブル選択を行うことができないと、データ書式の不一致によるエラーでマイコンがフリーズしたり、複数マイコンのデータが混在記録され、判別不可能になるといった問題が発生する。今回開発したシステムにおいては、マイコン自身が書き込むべきテーブルを識別し、必要があれば作成するという「テーブル自動追加機能」を組み込んだ。この機能により、データベースサーバー側の設定を殆ど変更することなく、接続するマイコン側の多様性に対応できるようにした。

データベースのテーブル名にはマイコンの MAC アドレス(Media Access Control address)を利用した。MAC アドレスは、原則として 1 つの通信機器に対し、一意な値が設定されるため、マイコン側で個別に記録先を設定しなくても、重複なく記録先テーブルを特定可能となる。

4.2　処理のフロー

テーブル自動追加機能の処理フローは以下の図.6 のようになる。

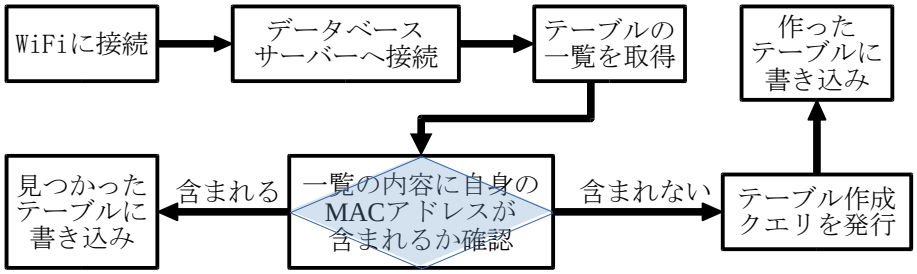

図 6. テーブル自動追加機能の処理フロー

マイコンは、WiFi に接続し、更にデータベースサーバーへと接続する。そして、データベースサーバーに対し、データベース内のテーブル一覧を取得するクエリ(SHOW TABLES;)を発行する。取得した一覧内の各テーブル名と自身の MAC アドレスを比較し、一致するテーブルがあれば、そのテーブルを書き込み先に設定する。

逆に、一覧内に自身の MAC アドレスと一致するテーブルが含まれなければ、自身の記録するデータ形式に応じたテーブル作成クエリ(CREATE TABLE 〜;)を発行し、テーブルを作成して、その作ったテーブルを書き込み先に設定する。

このフローでは、マイコンソフトウェアのアップデートや、マイコンの別センサーへの再利用などにより、同一 MAC アドレスで異なる形式のデータを記録しようとする場合には、既存の自身の MAC アドレスと一致するテーブルに対し、書き込みを行おうとしてしまい、エラーを発生させてしまうという問題がある。このため、マイコンソフトウェア改変後の再接続時には、既存テーブルの名称変更か、別のデータベースへの移動を事前に行っておく必要がある。

なお、データベース内に、マイコンの MAC アドレスとそのマイコンの使用上での識別名、用途を対応づける台帳的なテーブルを作成しておくと、リレーション機能の利用によって、使用上での識別名からマイコンを特定する処理を自動化できて便利である。

5　使用したマイコン(Espressif System 社 ESP シリーズ)

Arduino 規格ソフトウェアが稼働するマイコンをネットワーク接続する方法は、主に 2 つある。1 つは旧 Atmel 社(2016 年に Microchip Technology 社が買収)の AVR マイコンシリーズを用いた純正 Arduino、または、それと同等構成の Arduino 互換機と呼ばれるマイコンボードに、Arduino 規格の拡張ボードである Ethernet シールドや WiFi シールドを取り付ける方法。もう一つは、WiFi での通信機能を内蔵し、且つ、Arduino 規格のソフトウェアが稼働するマイコンを利用する方法である。前者は、マイコンボードで 1,000～4,000 円、拡張ボードであるシールドが 3,000～10,000 円と比較的高価になり、通信速度も低く制限される。後者だと、Espressif System 社の ESP シリーズマイコンが単体で 500 円程度、USB-UART 変換チップを搭載し、直接パソコンにつないで開発可能なマイコンボード製品で 1,500 円程度と安価で、基板サイズも小さく、通信速度も速くできる。このため、今回は、ESP シリーズマイコンをシステムに採用した。

5.1 ESP-WROOM-02

ESP-WROOM-02 は、ESP シリーズが WiFi 内蔵マイコンとして爆発的に普及する切っ掛けとなった SoC(System on Chip)である ESP8266 を搭載した Espressif System 社の純正モジュールである。

初期の製品のため、対応する無線通信は IEEE 802.11 b/g/n のみで、GPIO(汎用出入力)ピン数も 16 本と少なく、UART・SPI・I2C・ADC などの有線通信用ユニットも各 1 個と少ないなど、後続の ESP32 シリーズにすると性能はかなり見劣りし、価格は大して変わらないという短所を有する。一方で低機能、低性能故に消費電力が小さく、マイコンボード製品は基板サイズが小さいという長所を持っている。特に、起動時の突入電流が小さいために、電源の制約が緩いという点は非常に大きな利点となる。また、図.7 に秋月電子製の ESP-WROOM-02 使用マイコンボード「AE-ESP-WROOM02-DEV」を使用した温湿度測定システムの外観を載せる。このように一般的な 5 穴ブレッドボードを用いても、基板外部に配線を取り廻す余裕が残る程度に基板サイズが小さいという利点もある。

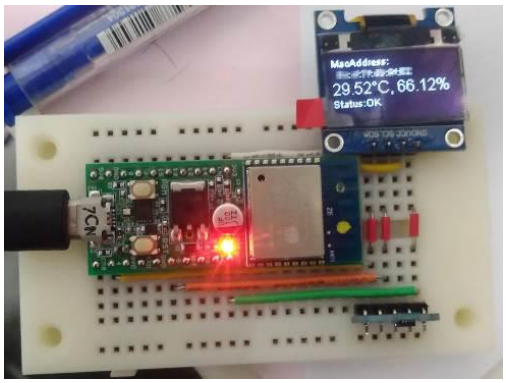

図 7. ESP-WROOM-02 のマイコンボードの例

5.2 ESP32 シリーズ

ESP32 シリーズは、ESP8266 より高機能化した後継製品群で複数の SoC が発売されている。また、メーカー純正モジュールだけでなく、メーカー純正のマイコンボードも発売されている。

高機能化の内容は、プロセッサのデュアルコア化と動作周波数の向上、対応無線規格が IEEE 802.11 b/g/n だけでなく Bluetooth v4.2 BR/EDR と Bluetooth Low Energy を追加、GPIO ピン本数の増加、各種有線通信ユニット等も複数個に増加、DAC・パルスカウンタユニットなど ESP8266 未搭載機能の追加など、非常に多岐にわたる。その代償に、消費電力の増大と、マイコンボードボード製品の基板サイズ増大を招いている。特に突入電流の増大が激しく、メーカー純正のマイコンボードであっても、モバイルバッテリー駆動では保護回路が作動する時がある。また、基板サイズの増大も取り扱い性を大きく低下させる。図 8.に示すように一般的な 5 穴ブレッドボードでは、基板の外側には片側に辛うじて 1 列穴が残るだけとなる。

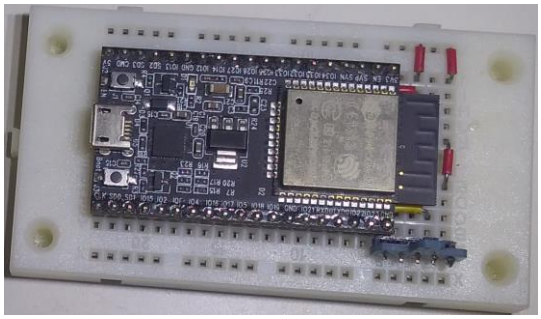
図 8. ESP32 シリーズのマイコンボードの例

6　システム立上げの主な流れ

今回開発した測定・記録システムを立ち上げる上で、主な流れと要点について述べる

6.1　データベースサーバー

データベースサーバーは、マイコンソフトウェアで使用する「MySQL Connector Arduino」に対応するものである必要がある。MySQL，MariaDB，PostgreSQL といった主要オープンソースソフトウェアでは動作は確認済みだが、本報告では最も使われそうな Ubuntu Linux を OS とし、MariaDB を使用する場合で例示する。(図.9)

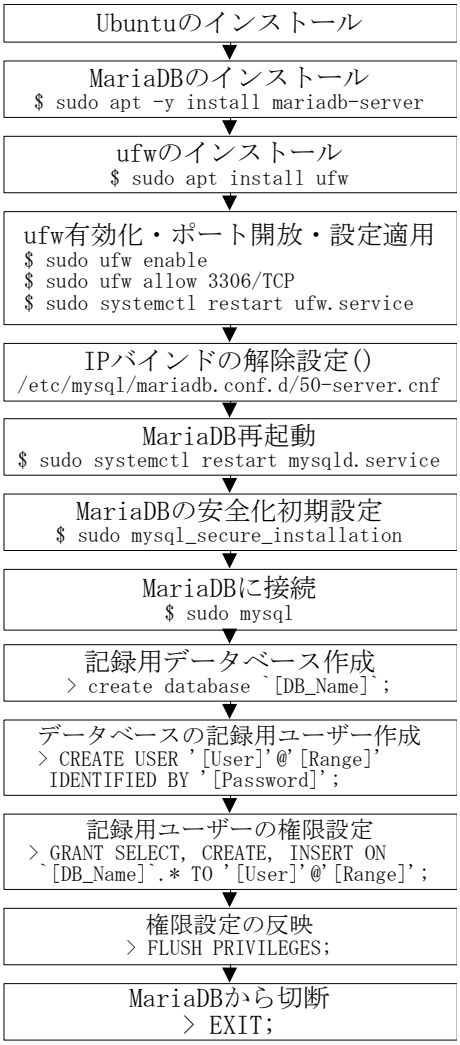

図 9. MariaDB 立上げの主な手順
コマンド行頭が「$」は Ubuntu への、
「>」は MariaDB への入力を示す。

まず、Ubuntu Linux をサーバーとなる PC にインストールする。Ubuntu Linux は、GUI(Graphical User Interface)で起動する Desktop 版と CLI(Command Line Interface)で起動する Server 版があるが、Linux やコマンドライン入力に不慣れな場合は Desktop 版を推奨する。理由は、Web ブラウザによる検索しながら作業を進め、コピー&ペーストで作業ができること、一部設定項目に関しては GUI によるマウス操作で実現できるので、事前知識を要しないこと、ローカル環境作業でも、端末画面を複数起動でき、他の設定を参照しながら作業を進めやすいことなどが挙げられる。短所としては、サーバー向けソフトウェアが同梱されず、手動インストールを要すること、サーバー用途で不要なソフトウェアがインストールされるので、若干のリソース(CPU・メモリ・ストレージ)を浪費すること、不要ソフトウェアも更新対象となるので、更新作業の時間が増大することなどが挙げられる。

OS インストール後は、SSH サーバー(OpenSSH)をインストールし、リモート操作を可能にすることが望ましいが、紙面の制限から本報告では割愛する。同様に、Linux の基本的な操作(管理者権限やファイル編集方法)も割愛する。

OS が利用可能になった後は、Linux の標準的ファイヤーウォールソフトウェア iptables の操作簡易化ソフトウェア(フロントエンド)である ufw をインストールする。これにより、ファイヤーウォールの管理が非常に簡単になる。インストール直後はファイヤーウォールを使用しない設定となっているので、「ufw enable」コマンドで有効化をする。そして、MariaDB が接続を受け付けるポート(通常は 3306/TCP)を開放する「ufw allow 3306/TCP」コマンドの実行と、設定内容の有効化の為に、ファイヤーウォール関連ソフトウェアを再起動する「systemctl restart ufw.service」コマンドを実行する。SSH を利用する場合は、その利用ポート(22/TCP)を開放する設定も、ここで行っておく。なお、標準ポートは攻撃対象とされやすいため、可能であれば、ポート番号を変更して運用することが望ましいと言える。

ファイヤーウォールの設定終了後は MariaDB サーバーの設定を行う。まず、MariaDB・MySQL・PostgreSQL については、「バインド(bind-address)」と呼ばれる機能があり、アクセスを特定の IP アドレスからのみに限定している。初期は「127.0.0.1」が指定され、同一 OS 内からのみアクセスを受け付ける状態となっている。外

部からのアクセス受付には、この機能を無効化する必要がある。それには Ubuntu Linux での MariaDB の場合には、「/etc/mysql/mariadb.conf.d/50-server.cnf」に書かれた「bind-address = 127.0.0.1」の行の先頭に「#」を付けてコメントアウトするか、値を「0.0.0.0」に書き換える必要がある。また、運用ポートを変更する場合は、「#port = 3306」の行の先頭の「#」を削除して、コメントアウト解除をした上で、値の部分を書き換える。設定ファイルの書き換えと保存が終わったら、「systemctl restart mysqld.service」にて MariaDB を再起動し、設定を反映する。

続いては、MariaDB の初期設定から推奨される安全確保設定への変更を行う「mysql_secure_installation」コマンドを実行する。実行される内容の詳細は、紙面の制限で割愛する。各自「MariaDB mysql_secure_installation」にて検索して頂きたい。1 点注意が必要なのは、最初の設定項目である「Switch to unix_socket authentication [Y/n]」で「Y」を入力とすると、MariaDB 管理用の初期ユーザー「root」の MariaDB へのログイン認証が「unix_socket」と呼ばれる方法となり、通常のパスワード認証ができなくなる。この場合はローカルの端末から「sudo mysql」など OS の管理者権限を持つ状態でデータベースクライアントを起動し、MariaDB に接続する必要がある。この機能を用いる場合は、別に、パスワード認証にて管理権限を持つユーザーの作成が必要となる。

MariaDB の安全確保後は、記録用データベースを「create database」コマンドでデータベースを作成し、「CREATE USER」コマンドでマイコンのアクセス用ユーザーを作成、「GRANT」コマンドで、SELECT・CREATE・INSERT の 3 種のクエリ実行権限を付与する。その後、「FLUSH PRIVILEGES;」にて設定変更の反映を行う。最後に「EXIT;」コマンドにて MariaDB との接続を切断し終了となる。

6.2　マイコンソフトウェア

まず、Arduino の開発環境 ArduinoIDE をインストールする。そして、ボードマネージャから ESP シリーズのマイコン用設定を呼び出せるようにする。「環境設定」ウインドウ(Windows 版ならメニューバーの「ファイル」-「環境設定」で呼び出せる)にある「追加のボードマネージャの URL」に ESP-WROOM-02 であれば「 http://arduino.esp8266.com/stable/package_esp8266com_index.json 」を、ESP32 シリーズであれば「https://dl.espressif.com/dl/package_esp32_index.json」を追記する。その後、ArduinoIDE を再起動する。

次は、「ボードマネージャ」ウインドウ(Windows 版ならメニューバーの「ツール」-「ボード」-「ボードマネージャ…」で呼び出せる)にて、使用するマイコンを選択し、必要なファイル類をインストールする。なお、ESP-WROOM-02 使用時には、本報告記述時点の最新版である Ver.3.0 系統では、「MySQL Connector Arduino」ライブラリでのデータベースへの接続がエラーで停止するため、Ver2.7 系統を選択する必要がある。その後、使用するボードを選択し、必要に応じて各種パラメーターを設定する。

続いて、「ライブラリマネージャ」ウインドウ(Windows 版ならメニューバーの「スケッチ」-「ライブラリをインクルード」-「ライブラリを管理…」で呼び出せる)にて、「MySQL Connector Arduino」ライブラリを選択、インストールする。これで、ESP シリーズでデータベースサーバーに直接アクセス可能なソフトウェア開発環境が整う。

マイコンに書き込むソフトウェアの実装の流れとしては、以下の様になる。

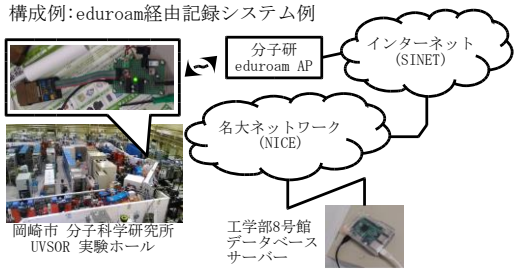

図 10. WPA-Enterprise 方式での利用例

まず、WiFi への接続が必要であるが、これはインターネット上の多くの記事があるので割愛する。なお、ESP32 においては、WiFi として一般的な WPA-PSK 方式のみでなく、eduroam や nuwnet1x といった WPA-Enterprise 方式の WiFi への接続も容易にできる。この機能を用いて、図

10.に示すような構成で岡崎市の分子科学研究所 UVSOR の光源加速器に設置した Beroz 社製ビームロスモニターBLM の出力パルスを ESP32 マイコン内蔵パルスカウンタにて計数し、加速器運転時の電子ビーム損失量の情報を、分子科学研究所設置の eduram を経由して名古屋大学内に設置したデータベースサーバーへ記録することに成功している。実際のコーディングについては「ESP32 eduroam」をキーワードとしてインターネット検索すると見つけることができる。

次にデータベースへの接続となるが、これも「MySQL Connector Arduino」ライブラリの公式 Web サイトにあるサンプル[1]に沿えばよいので割愛する。

テーブル自動作成機能については、テーブル一覧取得に関する手続きはライブラリの公式 Web サイトにあるサンプル[2]に沿うことになるが、かなり差異が大きいので、その実装コードを図 11.に示す。

データの書き込みである INSERT クエリの発行は、SHOWTABLES クエリの処理よりも簡単で loop 関数中で INSERT クエリを記述した文字列を「cur_mem->execute()」によって実行するのみで済む。ライブラリの公式 Web サイトにあるサンプル[3]も参考になる。

```
//ESP マイコンの MAC アドレス
String chipidWiFi.macAddress();
//  テーブル名は小文字のみなので、小文字に変換
chipid.toLowerCase();
//  クエリ処理用の MySQL_Cursor 型変数の定義
MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
//  データベース内テーブル名一覧の取得
cur_mem->execute("SHOW TABLES");
//  列名一覧の取得
column_names *columns = cur_mem→get_columns();
//  取得した一覧の各行を可．能する配列
row_values *row = NULL;
//  テーブル作成が必要かの判断フラグ
bool need_table = true;
//取得内容の展開
do {
  row = cur_mem->get_next_row();
  if (row != NULL) {
    // 1 列しか返ってこないので行データ先頭が
    String table_name = String(row->values[0]);
    //  テーブル名と自身の MAC アドレスと比較
    if (table_name.compareTo(chipid) == 0) {
      //一致するものがあれば作成不要フラグセット
      need_table = false;
    }
  }
} while (row != NULL);
//  不用になったクエリ処理用変数をメモリから削除
delete cur_mem;
//テーブル作成処理
if (need_table) {
  //  クエリ処理用の MySQL_Cursor 型変数の定義
  MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
  //   クエリコマンド文字列の生成
  String query =
    "CREATE TABLE `%name` ("
    "`Time_Inserted` DATETIME "
    "DEFAULT current_timestamp,"
    "～ "  //センサーに応じた列の定義を記述
    ");";
  //   クエリコマンド中の" %name" を MAC アドレスに置換
  query.replace("%name", chipid);
  //   テーブル作成クエリの実行
  cur_mem->execute(query.c_str());
  //   不用になったクエリ処理用変数をメモリから削除
  delete cur_mem;
}
```

図 11. テーブル自動作成機能の実装コード

7　現在の課題と今後の予定

現在のシステム開発状況として、接続する WiFi やデータベースを直接ソースコード中に書き込む方式となっており、汎用性に劣る。このため、外部 UART 通信からのコマンド受付機能の実装と、接続情報のマイコン内蔵フラッシュメモリへの保存機能の追加について現在検討中である。

参考文献

[1] GitHub, “ChuckBell/MySQL_Connector_Arduino/Examples/Connect with WiFi”, https://github.com/ChuckBell/MySQL_Connector_Arduino/wiki/Examples#connect-with-wifi-connect_wifiino

[2] GitHub, “ChuckBell/MySQL_Connector_Arduino/Examples/ Basic Select”, https://github.com/ChuckBell/MySQL_Connector_Arduino/wiki/Examples#basic-select-basic_selectino

[3] GitHub, “ChuckBell/MySQL_Connector_Arduino/Examples/ Basic Insert”, https://github.com/ChuckBell/MySQL_Connector_Arduino/wiki/Examples#basic-insert-basic_insertino