マイコンを用いた電力データ無線通信システムの構築

山村 健斗*、吉田 雅史**

Establishment of a wireless communication system

Kento YAMAMURA and Masafumi YOSHIDA

Abstract: We have developed a real-time monitoring system for electric vehicle power data. Using a microcontroller and low-loss sensors, we measured power consumption and transmitted the data to a cloud MQTT broker via a mobile router acting as an access point. The data subscribed from the broker was then written to a database server on a PC and visualized through real-time graphing.

Key words: MQTT, a wireless communication system

1. はじめに

宇部工業高等専門学校(以下、宇部高専)の電気工学科の有志で結成された自主活動グループ E-Project では、昨年度に 2025 年 CQ EV ミニカート・レース 九州大会(CQ 出版社主催)に出場した。競技 30 分間、ピットクルーはドライバーの体調はもとより、電動カーマシンの状態をリアルタイムで管理しなければならない。昨年度では電話を用いてドライバーの声で速度やバッテリー電圧電流を管理してドライバーに運転の指示を送った。しかし、この手法の場合に、ドライバーに計器を逐次見てもらう必要があり、集中力の低下、最悪の場合前方不注意による事故の発端になりうる可能性がある。そのため、自動でデータを送受信する機器が必須となる。そこで我々は、消費される電力データをリアルタイムで監視するシステムの構築を目指した。本稿では、その現状について初期結果ならびに今後の展望について報告する。

2. システムの概要

図 1 に本研究にて構築する電力データの流れを示した。電力データを取得して送信する送信側およびそのデータを受信する側にて構築される。青い線は電流電圧の測定、点線は無線通信、黒い線は電力データの流れをそれぞれ表している。電力データはホール効果センサとマイコンによって測定される。マイコンはモバイルルータをアクセスポイントとして、クラウド MQTT ブローカー[1]へデータを送る。送られたデータは時系列データベースに書き込み、グラフ化ソフトウェアで可視化する。

3. システムの詳細

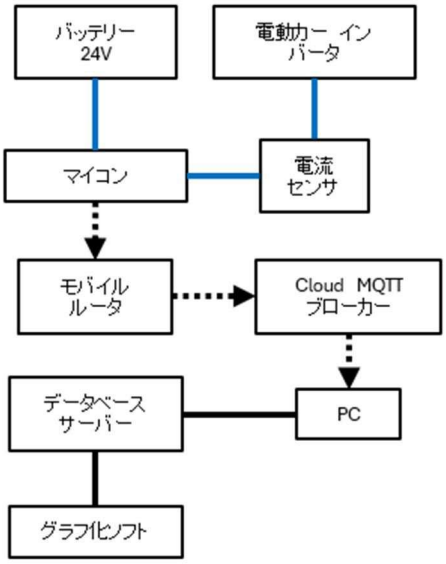

図 1: 電力データの流れ.

3. 1 電力測定

図 2 は、電動カーのバッテリー電圧とバッテリーからインバータへ流れる電流の測定に使用した回路図である。図 2 の青い線に相当する役割を担っている。バッテリーから供給される電圧約 24~25 V をスイッチングレギュレータ(R-78E5.0-1.0)で 5 V に降圧し、ホール効果センサ(ACS758)やマイコンに供給した。マイコンには ESP32-WEOOM-DA を用いた。電圧の測定には 34 ピン、電流の測定にはホール効果センサから出力される電圧を 35 ピンでそれぞれ読み取った。その際、ESP32-WEOOM-DA のロジックレベルである 3.3 V を超えないよう分圧して測定した。測定に用いたホール効果センサは抵抗値が 100 μΩと非常に小さく損失が小さい。

(2026 年 2 月 6 日受理)

責任著者：吉田雅史
* 宇部工業高等専門学校 電気工学科 3 年
** 宇部工業高等専門学校 電気工学科

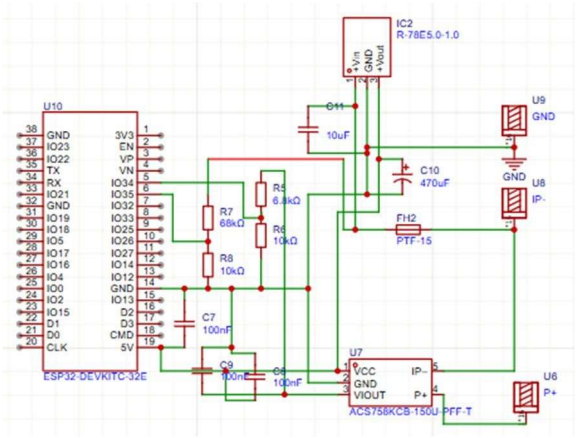

図 2: 回路図.

3. 2 無線通信

マイコンは Wi-Fi が使える ESP32-WROOM を用いた。マイコンはモバイルルータをアクセスポイントとして電力データをクラウド MQTT ブローカー（shiftr.io Cloud[1)]）に一秒間隔で送信する。データが送られると数値を WebSocket でブロードキャストし、時系列データベース(Influxdb3)に書き込む。書き込まれたデータはグラフ化ソフトウェア(Grafana)で可視化される。

MQTT とは、（Message Queuing Telemetry Transport）の略であり、IoT 向けのメッセージングプロトコルである。軽量なパブリッシュ/サブスクライブモデルでブローカーというサーバーが間に入り、小型のマイコンでも使用可能である。

4. 実験

実験では、ブレッドボードを用いて図 2 の回路を組んだ。25 m ほど離れた場所でデータの送受信を試験したところ、良好な通信を確認できた。これより、クラウド MQTT ブローカーにデータを送信する特徴から、クラウドに接続できる環境があれば、距離によらず通信が可能であることが明らかになった。以下に実験で観測したデータを詳述する。

4. 1 クラウド MQTT ブローカー

図 3 は動作時のクラウド MQTT ブローカーでのメッセージのやり取りを示す動作画面である。マイコンで測定した電力データがメッセージとしてクラウド MQTT ブローカーに送られており、そのメッセージを PC で受信する流れを可視化している。マイコンを EV_Sensor_ESP32ppp、PC を C1 で、さらにメッセージを黒い点で表している。動作時は黒い点が動くことでメッセージがいつ送信され、受信されたのかというタイミングを確認することができる。図 3 では PC がクラウド MQTT ブローカーからデータを受信するタイミングを切り取った。

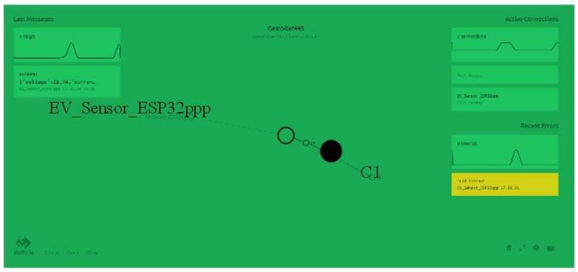

図 3: クラウド MQTT ブローカーの動作画面.

4. 2 時系列データベース

以下にクラウド MQTT ブローカーから送られたメッセージがどのように時系列データベースに書き込まれるかについてコードを交えて示す。

コード 1 はクラウド MQTT ブローカーから送られたメッセージ（バイト列）を文字列に変換している。変換されたデータをコード 2 で表示させる。その結果のシリアルモニタ画面を図 4 に示す。時間の表示を確認すると、設定した 1 秒間隔で電力データが送られていることがわかる。コード 3 ではデータにテーブルとタグをつけている。これにより電動カーが複数台に増えたとしても電動カーごとにデータを管理することができる。

```
1. payload_str = msg.payload.decode("utf-8")
   data = json.loads(payload_str)
2. current_time = time.strftime("%Y-%m-%d %H:%M:%S")
            voltage = data.get('voltage', 0.0)
            current = data.get('current', 0.0)
            wattage = data.get('wattage', 0.0)

            print(f"[{current_time}] V: {voltage:.2f} V | I:
        {current:.2f} A | P: {wattage:.2f} W")
3. point = Point("power_measurement")¥
                .tag("ev_id", "car_001")¥
                .field("current", data.get("current", 0.0))¥
                .field("voltage", data.get("voltage", 0.0))¥
                .field("wattage", data.get("wattage", 0.0))
```

```
2026-01-07 18:21:54] V: 23.91 V | I: 7.19 A | P: 172.00 W
2026-01-07 18:21:55] V: 23.91 V | I: 7.88 A | P: 188.40 W
2026-01-07 18:21:56] V: 23.91 V | I: 7.74 A | P: 185.10 W
2026-01-07 18:21:57] V: 23.91 V | I: 7.67 A | P: 183.50 W
2026-01-07 18:21:58] V: 23.90 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:21:59] V: 23.91 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:22:00] V: 23.91 V | I: 7.67 A | P: 183.50 W
2026-01-07 18:22:01] V: 23.91 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:22:02] V: 23.91 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:22:03] V: 23.92 V | I: 7.88 A | P: 188.50 W
2026-01-07 18:22:04] V: 23.90 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:22:05] V: 23.91 V | I: 7.67 A | P: 183.40 W
2026-01-07 18:22:06] V: 23.93 V | I: 7.67 A | P: 183.60 W
```

図 4: 時系列データベースに書き込まれたデータ.

4. 3 グラフ化ソフトウェア

以下に,、時系列データベースに書き込まれた電力データがどのようにグラフ化ソフトウェア(Grafana)[2]で可視化されるかをSQLクエリを交えて示す。SQL クエリとは時系列データベースへ特定の指示を投げかける命令文である。

コード 4 は取得する項目を示すクエリであり、コード 3 の. Field で書き込んだデータ current と time を取得している。コード 5 はどのテーブルからデータを取得するかを示すクエリで、コード 3 によってつけられたテーブルを対象にしている。コード 6 は、コード 3 でつけられたタグに絞り込んでデータを取得し、かつ表示する時間範囲をしている。コード 7 は得られたデータを新しい順に並べるクエリである。コード 8 は送られてくるデータの制限であり、処理が重くなることを防いでいる。データは電流と電圧、および電力の 3 種類であるためデータ量が膨大で、10000 を超えると処理が重くなる可能性が高くなる。そこで 3 種のデータをそれぞれ 2000 件に制限して、合計 6000 件を上限とした。

以下のクエリをグラフ化ソフト(Grafana)で実行したグラフが図 5 である。横軸時間に対して、緑、赤、そして青のプロットがそれぞれ電流、電圧、電力を表している。図 4 で書き込まれた時刻 18:22:00 の電力データを図 5 にプロットで追記したところ、読みとった値と書き込まれた値が一致することが確認できた。それにより、時間と電力データを正しく取得できていることが確認できる。

```
4. SELECT
    time,
    current
5. FROM
    power_measurement
    WHERE
6. ev_id = 'car_001' AND $__timeFilter(time)
7. ORDER BY
    time DESC
8. LIMIT  2000
```

図 5: 電力データのグラフ

4. 4 今後の計画

本実験により本システムによる電力データの無線通信の実現可能性が示された。電動カーの消費電力監視にあたり、省スペースかつ消費電力を抑えることが求められる。この点において本システムは強力なルータを用いた local のネットワークを構築するより優れ、安価であるが最大の利点と言える。今後は、実機への実装を想定してブレットボードではなくプリント基板によるシステムの構築を考えている。図 2 に示した回路図から図 6 のようにプリント基板を現在設計している。100 A の大きな電流が流れる可能性のあるホール効果センサには両面に広いパターンをすることで抵抗値を下げ、熱を逃がすことができるように工夫した。設計基板が完成次第、その有用性を確認する。

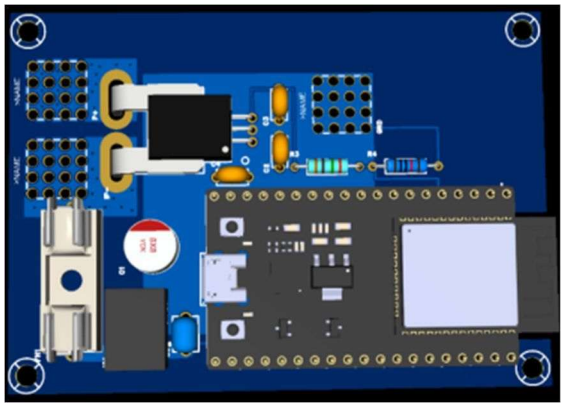

図 6: プリント基板設計図.

5. まとめ

本研究では、電動カートにて消費される電力データをリアルタイムで監視する無線通信システムの構築を目指した。MQTT プロトコルを使うことで、1 秒間隔で電流電圧、および電力の 3 種類のデータを 25 m 離れた先から取得することに成功した。電動カーの消費電力監視にあたり、省スペースかつ消費電力を抑えることが求められる。この点において本システムは強力なルータを用いた local のネットワークを構築するよりも優れた通信安定性を有しており、かつ低コストで実現できる。これらにより小型マイコンによる無線通信の実現の可能性を見出せた。

今後は設計したプリント基板を実際に電動カーに積み電力データを監視することを試みている。

※本研究は、「令和 7 年度 宇部高専 T&B 学生支援事業」により助成を受けた研究成果の一部である。

謝辞

本研究には、多くの方々にご支援いただきました。関係の皆様に心より感謝申し上げます。

参考文献

1) shiftr.io Cloud: shiftr.io 最終閲覧日 2026 年 3 月 2 日.
2) InfluxData: InfluxData 最終閲覧日 2026 年 3 月 2 日.