IoTデバイスに電源と環境情報と例外発生を供給する方式とこれを用いた開発支援環境の提案 ーRaspberry Com*PoTE と Raspberry Workbench の設計と実装ー

大野 浩之[1,a)]

概要：一般に，IoT デバイスを運用するには，安定した電源供給が必要である．また，各デバイスに正確な時刻を共有させたい場合が多い．さらに、多数の IoT デバイスを運用する場合には，特にその開発時においては遠隔からの一斉再起動を行いたい場合が少なくない．そこで，多数の IoT デバイスに対し，外部から「電源」「時刻等の環境情報」さらに「例外状態の発生」を安定して提供するしくみを設計し，これをRaspberryCom*PoTE（ラズベリー・コンポート）と名付けた．本報では，この方式の概要と，この方式を用いた IoT デバイス開発環境（Raspberry Workbench）について報告する．

キーワード：IoT, ものづくり, ものグラミング, Raspberry Pi, Arduino, POSIX, POSIX 中心主義, 有線 IoT

Design and Implementation of RaspberryCom*PoTE and Raspberry Workbench to Supply IoT Devices Electrical Power, Environmental Information, and Exceptions

Hiroyuki Ohno[1,a)]

Abstract: Generally, stable power supply is considered necessary for stable operation of IoT devices for a long period of time. Depending on the application, it may also be necessary to enable a forced restart from a remote location. In addition, when operating a large number of IoT devices, it may be necessary to supply an accurate time. Therefore, we designed a system that provides power and time from outside to IoT devices and provides stable notification of exceptional conditions. We named it `RaspberryCom*PoTE`. In this report, we report on the design, implementation and operation of `RaspberryCom*PoTE` on the premise of operating in an indoor environment of the size of a university laboratory.

Keywords: IoT, Mono-gramming, Monogramming, Raspberry Pi, Arduino, POSIX, POSIX centrics approach, Wired IoT

1. はじめに

IoT デバイスを長期間に渡って安定運用するには，安定した電源供給が必要である。このため，さまざまな方法が考案されている．たとえば，IoT デバイスの消費電力を極力小さくした上で，内蔵した電池が当該デバイスの想定運用期間中は寿命を迎えず給電可能にする方法はその一つである．この方法であれば，電源まわりは電池から給電のみを想定すればよく，電池交換のための機構や外部からの給電や充電方式については考える必要はなくなる．しかし，この方法が利用できる IoT デバイスの消費電力は現状では

[1] 金沢大学 総合メディア基盤センター
Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan
[a)] hohno@staff.kanazawa-u.ac.jp

それほど大きくはできない．たとえば，数時間に一度の頻度で，センサを使って得た計測結果を無線ネットワーク経由で母機となる機材に通知し，通知が終了したら直ちに深い休眠状態（以後「ディープスリープ」）に入るといった用途であれば，電池交換なしに数年間の連続運用を可能にした例がある．一方，母機からの非同期の指示にしたがってアクチュエータを動かすといった用途では電池交換なしに数単位の連続運用は現状では困難である．

上述の例では，組み込み用のマイクロコントローラ（以後「組み込みマイコン」）を用いるのが普通であるが，近年広く普及している手のひらサイズのコンピュータである Raspberry Pi の場合は、アイドリング状態であっても 5V 300mA 程度の電力を消費し，組み込みマイコでは容易に実現できるディープスリープには入れない．このため，同機と同程度の大きさの IoT デバイス用としては比較的大きめなバッテリーを組み合わせた場合でも，外部からの定期的な充電または常時給電なしに年単位運用を継続することはできない．

このような消費電流が大きい用途では，太陽電池等で必要な電力得てを自給する方法もあるが，十分な明るさを確保できない室内や夜間のような状況での運用には適さない．最近では電場や磁場を介して電力を無線伝送する例もあるが有効な場面は未だ限られている．結局，外部から有線によって電源を供給する従来の方法を必要とする場面は残っており，この場面についての技術的検討も意味を持っている．

電源供給方法の議論とは別に，開発段階の IoT デバイス多数を最終的な設置形態を想定しつつ長期間に渡って無人連続運転を行う場合には，当該 IoT デバイスのファームウェアを遠隔から更新したり，任意のタイミングで強制的に再起動するしくみがあると便利である．

さらに，多数の IoT デバイスを連携して運用する場合には，それぞれの IoT デバイスが正確な時刻を保持しているか必要に応じて利用できるとイベントを記録する際に適切なタイムスタンプを押すことができて好都合な場合が多い．IoT デバイスの規模が比較的大きければ，RTC（リアルタイムクロック）を持つ方法があり，さらに IP 接続性が確保できるなら，NTP プロトコルを用いて時刻同期を行って RTC を更新する方法が実用的であるが，小型のセンサと最小規模のマイコンの組み合わせなどでは RTC の搭載は可能でお IP 接続性の確保 NTP プロトコルの実装が難しい場合がある．このとき，RTC を搭載していても基準となる時刻と同期できなければ，タイムスタンプとしては約に立たないことに注意が必要である．

このような背景の下, 大学の研究室程度の規模の室内環境での運用を前提に,「IoT デバイスに電源 (**Power**) と時刻等の環境情報 (**Time**) と例外状態の発生 (**Exception**) を外部から安定供給する共通 (**Common**) の複合的 (**Complex**) な機構とそのための構成要素 (**Component**) 」を設計実装し，これを RaspberryCom*PoTE (ラズベリー・コンポート) と名付けて著者の周囲での供用を開始した．さらに RaspberryCom*PoTE を用いた IoT デバイス開発環境を試作し，これを RaspberryWorkbench と名付けた．

本報告では，RaspberryCom*PoTE の設計および RaspberryWorkbench の実装について報告する．

2. 背景

本報告では，大学の研究室のような規模の室内環境で IoT デバイスを長期に渡って安全安心かつ簡単に取り扱える環境の整備に主眼を置く．

これまでも著者は，RaspberryGate や RaspberryGuardian を設計実装して，IoT デバイスへの不適切なアクセスを防御するためのセキュリティゲートウェイの構築と運用機構を実装したり [1][2] ，I2C ベースの近距離有線ネットワークを提案して室内環境での IoT デバイスの運用環境を提供したり [3],「ものグラミング」と名付けた IoT 環境向けプログラミング手法を提案したりしてきた [5].

今回は，著者が 2017 年に Raspberry Leaves として報告した「I2C ベースの近距離有線 IoT ネットワーク」の運用経験を踏まえ，これを一部を簡素化した上で新たな機能を追加したしくみを検討した．具体的には，2017 年に提案した方式（以後「2017 年方式」）から長距離 I2C 通信機能を削除し，これに伴い I2C バスから基準信号を抽出する機能も削除した．代わりに以下の機能を改良あるいは追加し，簡単で使い勝手のよいしくみの実現を目指した．

- (改良) 安定した電源供給
- (強化) 例外状態（ハードウェア・リセット，ハードウェア割り込み）の通知
- (新規) 環境情報（時刻等・後述）の通知

長距離 I2C 通信機能は，2017 年方式には存在したものの本研究では削除したが，通信方式を差動型に変更して通信距離の延長と安定を確保した I2C 通信方式を現在評価中である．この成果は次世代の RaspberryCom*PoTE に取り込む予定である．

3. 関連研究

3.1 2017 年方式 (Raspberry Leaves)

本研究で提案する RaspberryCom*PoTE に最も近いのは，著者が取り組んだ 2017 年方式である．RaspberryCom*PoTE は 2017 年方式の延長上にある．

2017 年方式では 8 芯ケーブルの使用を前提とし，そのうちの 4 芯で電源と有線通信機能（実体は駆動電圧を 12V に上げて通信距離の延長を実現した I2C バス互換の通信方式）を提供し，残りの 4 芯を 2 つ目の I2C バスにしたり，1-Wire を使った構成管理機能に割り当てたりした．2017 年方式を RaspberryCom*PoTE と比べると，電源を供給す

るという考え方は同じであるが，RaspberryCom*PoTE では I2C バスベースの双方向有線通信は提供していないし，I2C バスを持たないため，I2C バスのロックアップ回避・回復機能も持たない．一方，RaspberryCom*PoTE は有線通信のため RS-485 を採用しているので低速な双方向通信は可能である．しかし，現時点では以下で述べる「環境情報」を提供するためだけに用いており，結果的に一方向通信になっている．

両方式とも IoT デバイスに時刻情報を供給することは重要だと考えているが，2017 年方式では基準時間信号と称する正確なクロックを提供しようとしているのに対して，RaspberryCom*PoTE では時刻そのものを適切なタイミングで供給している．また，RaspberryCom*PoTE では時刻を「環境情報」の一つと位置づけており，時刻以外にも IoT デバイス群が設置された環境に関する諸情報（気温，湿度，気圧，明るさ等）を供給・共有できる．さらに，RaspberryCom*PoTE では必須の機能として実現しているハードウェア割り込みやハードウェア・リセットは，2017 年方式ではオプション機能と位置づけ，1-Wire を使った構成管理機能に委ねているため必須機能ではなかった．

以上をまとめたのが表 3.1 である．RaspberryCom*PoTE は 2017 年方式の開発経験を踏まえているが完全な後継ではない．

機能	2017 年方式	ComPoTE	備　考
電源供給	✓	✓	
有線通信	✓(I2C)	✓(RS-485)	
基準クロック	✓	ー	
現在時刻	ー	✓	
環境情報	ー	✓	時刻以外は準備中
構成管理	✓(1-Wire)	ー	オプション
割り込み	ー	✓	
リセット	ー	✓	

表 1　2017 方式と RaspberryCom*PoTE の比較 (✓ が該当ありを示す)

3.2　有線 IoT

RaspberryCom*PoTE は「有線 IoT」と呼ばれる分野に属する．この分野では，既存のイーサネット技術を元に，最長 1km の距離で最大 10Mbps のデータ伝送速度と 1〜50W の給電を 1 組の撚対線で実現する，IEEE802.3cg の標準化作業が行われている．標準化後には大いに注目される技術になると予想できるが，現時点では当該規格に 10 挙した製品を入手できる状況ではない．2017 年方式も RaspberryCom*PoTE も室内のような小規模な領域での利用を前提としている．RaspberryCom*PoTE には通信機能が搭載されておらず，通信が必要な場合には WiFi や Bluetooth などの通信方式を併用する必要があるし，電力供給のための電源電圧も 12〜15V 程度であり，IEEE802.3cg が 50W を供給するために DC60V を想定していることと比べると RaspberryCom*PoTE が適用できる用途や範囲は IEEE802.3cg の適用範囲より小規模になる．一方，RaspberryCom*PoTE には環境情報の提供や例外状態の通知など IEEE802.3cg にはない機能を有しているため，この機能を必要とする用途であれば RaspberryCom*PoTE に優位性がある．

3.3　PoE

通信用の配線を使い，通信しつつ電力を供給する方式としては PoE（PowerOverEthernet）が普及している．PoE の基本機能は 2003 年に発効した IEEE 802.3af で規定されており，これに従えばカテゴリ 5 以上の UTP ケーブルを使って電力を供給できる．通常，電力送出側の機器は直流 48V 前後で送電し，受電側は最大で約 13W の電力を取り出せる．PoE は，有線環境で電力供給と通信を両立させる方式としてはもっとも成功している．RaspberryCom*PoTE を PoE と比較すると，通信機能も電力供給も劣るが，PoE にはない環境情報の提供や例外状態の通知などの特定目的に特化した機能を有する．また，電力供給のために PoE が利用する 48V 前後の電圧は電子回路の開発者にとっては高圧であり，開発中の接触事故や短絡事故などの際に周辺に与える損傷は，15V 前後の電圧を用いる RaspberryCom*PoTE の場合より大きい．

3.4　PLC

PoE とは逆に，交流電力線を使って電力を供給しつつ通信を重畳する方に PLC（PowerLineCommunication 電力線通信）がある．

日本においては周波数の低い　50Hz や 60Hz の交流電力線に 450kHz 以下の帯域の信号を重畳して通信を行う方式がまず実用化され，その後 2006 年には 2〜30MHz の帯域も利用できるようになった．後者は高速電力線通信と呼んで前者と区別する場合がある．前者では 9600bps 程度，後者では 100Mbps を超える通信速度を提供する機器が実用化されている．

一方，RaspberryCom*PoTE ではハードウェア割込信号やハードウェア・リセット信号を直流の電源線に重畳するが，その信号は数 Hz あるいはそれ以下の速度で電源電圧を変動させることで実現している．すなわち供給する電力は直流である．PLC 技術は交流電力線に信号を重畳しているため，参考にはなる点もあるが RaspberryCom*PoTE とは異なる方式である．

3.5　DCC

PoE では，送信側は通信用の交流信号に電力用の直流を加えて送り，受信側では本来の信号と電力供給用の直流を分離して用いている．

PLC は，送信側は電力供給用の交流電力線に通信のための高周波信号を乗せて送り出し，受信側は交流電力を受電しつつ，重畳した高周波を取り出して複合し，受信した情報を得る方式である．すなわち周波数の低い交流電力線に周波数の高い交流信号を重畳している．

これらに対して，送信側は送出したい情報を PWM 変調してこれをある程度の振幅の交流として送り出すことで，信号そのものに電力搬送の役割も担わせ，受信側は受け取った受け取った信号を復調して情報を得るとともに，これを整流することで電力とする方法も考えられる．この，信号線に電力搬送も担わせる方式の一つに，鉄道模型の分野で模型をデジタル信号で遠隔操作する目的で使われている DCC (Digital Command Control) がある [4].

DCC では，制御装置から 2 本のレールに ± 12V 程度の振幅を持つ 数 kHz の PWM 信号を流しており，レール上の模型はこれを復調することで制御装置からの指令を受け取り，整流することで電力を得ることができる．DCC は，想定する運用環境の大きさも，運用に供する機材の数も，RaspberryCom*PoTE が想定する環境に近い．このため DCC を利用することも検討したが，作動通信方式を導入するとはいえ基本は I2C バスである 2017 方式との統合も視野に入れているため，DCC の採用は見送った．

4. RaspberryCom*PoTE の設計と実装

本節では，RaspberryCom*PoTE の設計方針を明らかにし，さらに実際の設計と実装について述べる．

4.1 方針

電源供給・環境情報提供・例外状態通知 3 つの需要を満たす方法を設計実装するにあたり，以下の方針をとった．なお，現時点では環境情報として提供しているのは時刻のみである．

(1) 配線に用いる線数（配線材の芯数）はできるだけ少なくするなど，全体として簡素であることを重視する
(2) 既存の類似の諸方式との互換性確保には固執しない
(3) IoT デバイス同士，IoT デバイスと IoT デバイスの母艦となる計測制御用コンピュータとの間で発生する実際の通信については積極的には提供しない
(4) 開発の初期段階から日常の研究開発環境に投入して常時利用し，問題点があれば直ちにフィードバックして改善・変更する方式をとる

これらのうち第 3 項については，後述するように RaspberryCom*PoTE では，環境情報の提供に RS-485 による有線通信を用いているので，RaspberryCom*PoTE が提供する機能を用いて IoT デバイス同士や IoT デバイスと計測制御用コンピュータが相互に通信することは可能である．しかし，このような機材間の通信は，頻度，通信量，実時間性への要求などが利用形態によって全く異なる．このため，RS-485 で対応可能か否かは利用形態ごとに個別に検討する必要がある．また，利用目的や利用形態が異なる複数の装置が一つの RaspberryCom*PoTE を利用して電源や環境情報の供給を受けることは可能であるが，目的も形態も異なる装置が一つの RS-485 ネットワークを利用すると回線の利用を調停する必要が生じ，簡素に実現することはできない。一方，現在では機材間の通信は RS-485 より高速な WiFi, Bluetooth, その他の方法で実現できるのが普通であり，無理に RS-485 の利用を強制する必要はない．そこで，機材間の実際の通信は RaspberryCom*PoTE に含めないことにした．

この方針の下，RaspberryCom*PoTE では 4 芯ケーブルを用いることとし，このうち 2 本で電源供給と例外状態通知を，残りの 2 本で環境情報の提供を実現することとした

ところで，拡張した I2C バスを持つ 2017 年方式では 8 芯ケーブルを使うことを前提としている．この 8 芯ケーブルを，今回提案する RaspberryCom*PoTE で利用すると 4 芯が残る．この残った 4 芯を用いれば，2017 年方式の実装後に別途実装と評価を行った「差動方式の I2C バス」(4 芯必要) を導入できる．機材間の通信を含めて全てを有線で構成したい場合は，通信速度が I2C 通信の速度に拘束されてしまう問題があるが，この差動方式の I2C バスの採用が有力である．すなわち，8 芯ケーブルを用いれば，2017 年方式が提供していた有線通信と RaspberryCom*PoTE とを融合させた別の方法を検討する余地がある．このことについては考察にて言及する．

4.2 設計と実装

上述の方針に基づいて設計実装した RaspberryCom*PoTE の概要を図 1 に示す．

図 1 RaspberryComPoTE 全体構成図

以下では上図に記された 3 つの系，すなわち電源供給系，例外状態通知系，環境情報提供系の設計と実装について述べる．

4.2.1 電源供給系

電源供給系は，単なる定電圧電源ではなく二つの異なる電圧間を必要なタイミングで遅延なく遷移する機能が必要になる．これは，RaspberryCom*PoTE では，例外状態の発生を電源電圧の変動で表現するためである．2017 年方式にも存在するこの機能は同方式と同様に非安定電源機能

と呼んでいる．非安定電源機能は，安定化電源装置の出力電圧を外部から意図的に変動させる機能である．この電位の変動は，受電側（RaspberryCom*PoTE を利用する各 IoT 機器側）においては，4 芯ケーブルを経由することで生じる電圧降下のために相応に低下した状態となるが，重要なのは二つの状態に対応した電位そのものではなくその電位差である．この電位差はほとんどの利用形態で送電側と受電側とで同じになる[*1]．

なお，各機器が最終的に必要とする電源は，各機器ごとに用意する昇降圧型のレギュレータで電圧を調整するため，送出時点での電源電圧の変動と 4 芯ケーブルを経由することで発生する電圧降下の影響は，個別の IoT 機器には及ばない．

現在の実装では，電源供給系自体への入力電圧は直流 16V，出力電圧（送出電圧）は通常時が DC 13.5V 前後，例外状態通知状態では 9V 前後 としている．

例外状態通知のための電源電圧の変動速度は 2-4Hz 程度なので通信としては超低速な部類になるが，定電圧電源装置にスイッチングレギュレータを採用すると数 Hz の速度であっても電圧を任意のタイミングで素早く変化させることは難しい．これはスイッチングレギュレータでは，二次側に容量の大きなコンデンサを取り付けるのが普通で，この容量により出力電圧の設定を変更しても追従するのに秒単位の時間がかかるためである．そこで今回は，スイッチングレギュレータより変換効率は劣るが，設定電圧の変化への追随が速いシリーズ型の定電圧電源回路を採用し，LM338 5A 可変レギュレータ [7] を用いて実装した．電源供給系の構成を図 2 に示す．

図 2 電源供給系

各 IoT 機器側に用意する昇降圧型のレギュレータには，4.5〜18V の入力電圧を許容する 3W 級絶縁型昇降圧型 DC-DC コンバータの MCWI03-12S05[8] を採用し，最終的には安定した DC5V 最大 600mA 供給している．さらに，各 IoT 機器には，電源電圧変化の検出に特化した専用 IC の M51957B(RNA51957B)[6] を配置し，次項で述べる例外状態を検出可能にした．

[*1] 高電位状態での供給電流を I_H ，低電位状態での供給電流を I_L ，線路の直流抵抗を R とすると，受電側の電位差は厳密には $R \times (I_H - I_L)$ だけ小さくなるが，この差は今回の用途にとっては十分に小さい

電源供給系の実装にあたっては，付加機能として設定以上の電流を検出した際に直ちに給電を停止し，音と光で警報を発報する機能も用意した．

通常，定電圧電源装置は，シリーズ方式かスイッチング方式かによらず，規定値以上電流が流れ続けた場合には一時的に給電を停止するか供給電圧を大きく下げ，過電流状態が除去されると元の電圧での給電を再開する機能がある．RaspberryCom*PoTE においても，最終的には長時間無停止無人運転を行うのでこのような機能は必要となるが，この機能が働く電流値は多くの場合装置ごとに決まっていて任意の電流値には設定できない．さらに，現在はまだ開発中のため試行錯誤を伴う運用がある．このような状況では，あらかじめ決めた以上の電流が意図せず流れた場合には給電を直ちに完全に停止するとともに，光と音で警報を発するなどして開発者に知らせ，原因を除去するまでは自動での通電再開はできないようにする機能があるのが望ましい．

この機構を望んだのは，意図せぬ誤接続誤接触で電源系がショートしたのにそれが短時間だったため直ちに通電が回復してしまい，同じことが何度も発生したのに異常に気づくのに時間を要した経験が何度かあったためである．RaspberryCom*PoTE においては，4 芯ケーブルが長くなるとケーブルの抵抗成分が無視できない大きさになり，引き回しによっては 1 Ωを超えることがある．このような場合，IoT デバイス側で電源まわりがショートしても非安定電源側ではたかだか 5-10A 程度の電流に留まってしまう．このような場合，レギュレータの保護回路が働くまでに数 10 秒の時間を要したり，発熱はしてもは保護回路は全く働かず給電を続けたりする．つまり，過電流検出と電流制限をレギュレータの機能に任せてしまうと，レギュレータの規定値を上回る電流を流さない限り保護回路は働かない上，多くの場合でケーブルの直流抵抗成分が供給上限以下の電流に留めてしまう．例えば，今回用いた LM338 は，データシートによれば 8A 以上の電流が一定時間（過渡的には 12A 以上）流れないと保護回路が働かない．一方，開発中の IoT デバイスに誤電流が流れた場合，1A かそれ以下の電流が流れても異常状態ということがあり得る．上述の理由により 1A 程度の電流は LM338 にとっては正常の範囲であって過電流ではないので保護回路は働かないが，任意の値に設定できる過電流検出回路とそれに連携した給電停止回路を用意することで，異常状態検出機能とレギュレータの給電限界とを切り離して対応できる．加えて，給電停止となったら音と光で警報を発する機能を用意すれば，原因を除去するまでは給電停止と警報の発報が繰り返され，不用意な通電継続による回路の損傷を回避できる．この回路の概要を図 3 に示す．

図 3 過電流検出・警報

4.2.2 例外状態通知系

すでに言及しているように，RaspberryCom*PoTE では，接続した IoT デバイスを一斉に再起動したり主としてハードウェア割り込みを発生させる，「例外状態」の発生を通知できる。

現時点ではハードウェアリセット 1 種類とハードウェア割り込み 1 種類を提供しているが，これらの状態を通知するために上で述べた電源供給の非安定能を利用した．実装を簡単にするため，供給電圧が通常状態の約 13.5V から例外通知状態の約 9V への降下を検出した時点でハードウェア割り込みを発生させ，それが一定時間（たとえば 0.8 秒）以上継続した場合にはさらにハードウェア・リセットを発生させる回路を用意した．これにより電源電圧降下を 2〜5Hz の速度で繰り返せばハードウェア割り込みは繰り返し発生し，ハードウェア割り込みハンドラ側でこの回数を数えることで，割り込み処理の挙動を順次変えることもできる．

図 4 例外状態通知系

（※メモ：図）ここにリセット・割り込み回路の図（ブロック図（と回路図？））（図 4）を入れる

4.2.3 環境情報提供系

著者は，運用中の IoT デバイスの周囲の諸情報を環境情報と呼んでいる．環境情報の筆頭は時刻情報である．時刻情報は IoT デバイスが何らかの情報を記録する際のタイムスタンプとなり，何らかの動作をする際のタイミングの基準となる．時刻情報以外には気温や湿度を把握したい場合があるだろうし，用途によっては気圧や明るさ磁場電場なども必要かもしれない．

これらの情報は，IoT デバイスによっては，デバイス自体が自ら取得して利用できる場合があるが，小型の IoT デバイスではそのような機構が省かれることもある．たとえば，時刻であれば実時間時計（RTC - Real Time Clock）を搭載するだけでは正確な現在時刻を取得するには不十分である．そのため時刻同期の仕組みが必要であるが，パソコンでよく利用される NTP サーバへアクセスして NTP プロトコルを用いて時刻同期をするには，IoT デバイス側に TCP/IP プロトコルスタックを用意し，IP 到達性を確保するための物理層とデータリンク層を用意しなければならない．IoT デバイスをできるだけ小規模に作ろうとした場合，NTP プロトコルによる時刻同期はソフトウェア領域を圧迫し新たなハードウェアの追加を必要とするなどオーバーヘッドが大きくなる．たとえば，Arduino UNO を用いた計測系を作る場合，Arduino UNO のプログラム領域はわずか 30kB 程度でありここに TCP/IP プロトコルスタックに加え、DHCP, DNS, NTP などのサービスを書き加えると 30kB の領域の可搬を消費してしまう．

（※メモ）引き続き書く

・環境情報提供サーバの時刻提供機能

時刻情報の提供と取得は以下のように行う

まず時刻提供用の機材（環境情報サーバ）を用意する．必要なの以下の機能である．なおカッコ内には必要とな具体的な機能である．

- NTP サーバに到達できる機能（TCP/IP プロトコルスタック，IP 到達性）
- NTP サーバと時刻を同期を行い，内部のマイクロ秒カウンタの現在値とその時点での UNIX 秒（1970 年 1 月 1 日 0 時 0 分 0 秒の UTC を 0 秒とする整数値）とのオフセットを求める機能（SNTP プロトコル）
- RS-485 回線に文字列を出力する機能（RS-485 ドライバ）

この機能を使い，以下の手順で時刻情報を RS-485 回線に送出する

(1) 内部のマイクロ秒カウンタの下 6 桁がゼロになるタイミングの一定時間前（通常は 100m 秒程度）に改行文字を送出（これが「予報」となる）
(2) 内部のマイクロ秒カウンタの下 6 桁がゼロになるタイミングでプリアンブルとなる “T” を送出
(3) 続いてこのタイミングにおける UTC 値を送出る機能

このため，環境情報サーバは ESP-8266 を用い Arduino 開発環境上で C++言語を用いて実装した（図 5）．

図 5 環境情報提供系

（※メモ：図）環境情報サーバの構成（図 5）

・環境情報提供サーバが提供する時刻との同期方法

RaspberryCom*PoTE を利用する IoT デバイスは RTC は搭載せずとも，Arduino UNO クラスの 8 ビットマイコンでも無理なく実現されている タイマ割り込みで更新するマイクロ秒カウンタが必須となる．そして以下の手順で，自らのマイクロ秒カウンタと UTC とのオフセットを決定する．

(1) UART から 1 バイト以上の読み出しが可能かを確認する．読み出せる文字がなければ他の作業に制御を移す．
(2) もし 1 バイト以上の読み出しが可能なら 1 バイトずつ読み出し改行文字が現れるまで読み飛ばす．
(3) 全て読んでも改行文字が現れなければ 1 に戻る
(4) 改行文字が得られたら，一定時間以内（通常は 100m 秒程度）に次の文字が届き，それがプリアンブル（文字”T”）がくることを期待してビジーループで待機
(5) 1 バイ以上の呼び出しが可能になったらその時点でのマイクロ秒タイムカウンタの値 T0 を取得
(6) プリアンブルがこなければ T0 を破棄して 1 に戻る．
(7) プリアンブル以後の文字を取得できるだけ取得してそれが UNIX 秒として妥当な文字列なければ T0 を破棄して 1 に戻る
(8) UNIX 秒として妥当な文字列が取得できたらこの文字列を UNIX 秒を示す整数値に変換する．5 で取得した T0 は，この UNIX 秒に対応するマイクロ秒カウンタの値である．

この方法には，時刻ずれを生じる要素として以下が考えられる．

(1) 環境情報提供サーバのマイクロ秒カウンタから得られる UTC の推定値と実際の UTC のずれ（NTP 時刻同期の誤差）
(2) 環境情報提供サーバがプリアンブルを出力してから，IoT デバイスが UART からプリアンブルを読み出すまでの時間
(3) IoT デバイスが 1 文字読み出してからマイクロ秒カウンタの値 T0 を取得するまでの時間

このうち 1 は通常は 1 ミリ秒より十分小さな値で数〜数 10 マイクロ秒に追い込める場合が多い．2 は、RS-485 回線の通信速度が RaspberryCom*PoTE が採用する 57600bps の場合，140 μ秒程度かかるが，ゆらぎのない固定値とみなせる値である．3 は 10 マイクロ秒かそれより十分小さい以下であり，実験によってゆらぎのない固定値を取得可能である．以上のことから RaspberryCom*PoTE による時刻同期は 1 ミリ秒より十分小さい値に追い込めることがわかる．

・RaspberryCom*PoTE における時刻同期を実現するスケッチ

前目で述べた手順を Arduino UNO 用のクラスライブラリとして実装した．

（※メモ）SoftwareSerial や Serial1 を使った
（※メモ）どのくらいの大きさ？

・時刻以外の環境情報の提供と利用

（※メモ）時刻以外の環境情報

5. 評価 (0.75p / 6.25)

5.1 評価方法

(※メモ) 評価方法を提示する

5.1.1 電源供給系

(※メモ) 電源電圧の遷移が想定の速度でできるかを確認する

5.1.2 例外状態通知系

(※メモ) 電源電圧の遷移によってハードウェア割り込みとハードウェア・リセットができるか確認する．ハードウェア割り込みが一定の速度で繰り返された場合，ハードウェア・リセットを起こすことなくハードウェア割り込みを繰り返せるかを確認する

5.1.3 環境 (時刻) 情報提供系

(※メモ) 今回の評価は時刻同期に限定する．RaspberryCom*PoTE に Arduino UNO を接続して時刻同期させて上で毎正秒ごとに正秒パルスを発生させる．同時に GPS 受信機からも正秒パルス出力し両者の時差をマイクロ秒単位で計測する

5.2 評価結果

(※メモ) 評価結果を提示する

5.2.1 電源供給系

図 6 評価：電源供給系

5.2.2 例外状態通知系

図 7 評価：例外状態通知系

5.2.3 環境 (時刻) 情報提供系

図 8 評価：環境 (時刻) 情報提供系

6. 考察 (0.75p / 7.0)

(※メモ) 他の方法との比較

7. 今後の展開 (0.5p / 7.5)

今後の展開については以下の 3 つの方向がある

一つ目は，現在の RaspberryCom*PoTE の量的拡張で，著者の研究室周辺で利用可能箇所を増やすとともに，著者の研究グループ内外への展開も行いたい．

（※メモ）

二つ目は，現在の RaspberryCom*PoTE の質的拡張で，距離と安定性を増した I2C ベースの近距有線ネットワーク機能の追加を行う．

（※メモ）

三つ目は，

（※メモ）検討中

8. おわりに (0.25p / 7.75)

本研究では...

（※メモ）検討中

(※これ以降で 0.25p / 8.0)

謝辞

本研究を遂行するにあたり，カナダ国ノバスコシア州立ダルハウジー大学コンピュータサイエンス学部の Prof. Sampalli および同教授の研究室の大学院生からは「ものグラミング」の有効性について検討する際に多くの示唆を得た．この経験が，RaspberryCom*PoTE の開発を後押しした．また，ユニバーサル・シェル・プログラミング研究所の當仲寛哲所長をはじめとする同社の研究部門の方々との議論も有益であった．ここに記して感謝したい．

参考文献

[1] ※大野浩之，RaspberryGate，情報処理学会， ...
[2] ※大野浩之，RaspberryGuardian，情報処理学会， ...
[3] 大野浩之，データ通信機能に電源供給・回線復旧・基準信号供給機能を加えた I2C ベースの近距離有線ネットワークの提案，情報処理学会研究会報告，Vol.2017-IOT-38，No.12，2017-06-24.
[4] Digital Command Control https://en.wikipedia.org/wiki/Digital_Command_Control （参照 2020-12-21）
[5] 大野浩之, 森祥寛, 松浦智之，ものグラミング 2 –諸機能の選択と集中を徹底した POSIX 中心主義に基づく IoT 開発方式の提案–，情報処理学会研究会報告，Vol.2019-IOT-46，No.00，2019-06-14.
[6] RNA51957A,B 電圧検出リセットシステム IC（※準備中※），https://www.renesas.com/jp/ja/doc/products/linear/r03ds0010jj0600_rna51957ab.pdf （参照 2019-08-03）
[7] LM138 and LM338 5-Amp Adjustable Regulators（※準備中※）http://www.ti.com/lit/ds/symlink/lm338.pdf （参照 2019-08-03）
[8] MINMAX MCWI03 series（※準備中※）https://www.minmax.com.tw/ja/download/files/307/MCWI03_Datasheet.pdf （参照 2019-08-03）