Vol.2019-IOT-46 No.16
2019/6/14

# 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

大野 浩之[1,a)]　松浦 智之[2,b)]　森 祥寛[1,c)]

**概要**：著者らが提案する「ものグラミング」は，ものづくりのためのプログラミング方式と当該方式を用いた実装環境である．今般，これまでの「ものグラミング」を見直し，IoT 環境への適用も念頭に置いて「マイクロコントローラと POSIX コンピュータのそれぞれが担当する機能」の選択と集中を行った．その結果得られた IoT 開発環境は，単純ながら拡張性に富み，安全と安心の確保も容易になった．本報ではこの新しいものグラミング方式（ものグラミング 2）の考え方を述べ，既存の他方式との比較，今後の課題，さらにすでに国内外で展開している普及啓発活動について報告する．

**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

# Monogramming2 - IoT Development Method Based on POSIX Centricism

Hiroyuki Ohno[1,a)]　Tomoyuki Matsuura[2,b)]　Yoshihiro Mori[1,c)]

***Abstract:*** The “Mono-gramming” is a programming and implementation method for MAKERs who are developing new devices using microcontrollers. This time, we reviewed the method, and we chose the selection and concentration of functions to be provided. The new method separates the role for microcontrollers and POSIX based computers. It is simple, easy to extent, safe and secure. We name the method “Monogramming 2”. In this report, we discuss abount the benefit of the method, future plans, deployment in Japan and other countries.

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

## 1. はじめに

「ものグラミング」は，著者らが提唱し研究開発と普及啓発に取り組んでいる「ものづくり」に適したプログラミング手法である [1]．本報では「ものグラミング」の第二世代にあたる「ものグラミング 2」について，その背景にある考え方，それに基づく実装，普及啓発活動の現状を報告する．

[1] 金沢大学 総合メディア基盤センター
Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan
[2] ユニバーサル・シェル・プログラミング研究所
[a)] hohno@staff.kanazawa-u.ac.jp
[b)] richmikan@richlab.org
[c)] mori4416@staff.kanazawa-u.ac.jp

「ものグラミング 2」の詳細は 3 節で述べるが，その特徴を POSIX[3] 系 OS のシェルにおける 1 行プログラム（ワンライナー）で表現すると以下のようになる．

```
$ cu -l /dev/ttyX < NP | script1.sh \
     | cu -l /dev/ttyY | script2.sh > NP
```

ここで，NP は mkfifo コマンドで作成した名前付きパイプ，cu は古くは UUCP 時代から POSIX 系 OS で利用されている，シリアルポートを介した通信を提供する cu コマンド，scriptN.sh は任意のシェルスクリプトである．また，このワンライナーを動かすコンピュータは少なくとも二つ

のシリアルポート (ttyX と ttyY) を持つ必要があり，各シリアルポートには．単機能のマイクロコントローラ（以下，組み込みマイコンあるいは単にマイコンと表記，たとえば Arduino が接続され，センサを使って計測したりアクチュエータを使って操作しているものとする．

このワンライナーにより，(1) 最初の cu コマンドによって /dev/ttyX から読み出された 1 台目のマイコンが生成したデータは，script1.sh での処理を経て 二つ目の cu コマンドによって /dev/ttyY に到達し 2 台目のマイコンに送られる．(2) 2 台目のマイコンは受け取ったデータをもとに何らかの動作を行い，その結果をワンライナーが動く機器に送り返す．(3) 送り返されたデータは二つ目の cu コマンドによって /dev/ttyY から読み出され，script2.sh での処理を経て名前付パイプ NP に出力される．(4) 名前付きパイプの反対側は，最初の cu コマンドの入力に繋がっているので，最終的には /dev/ttyX に到達し，/dev/ttyX に接続されたマイコンに送られる．(5) データをは受け取った 1 台目のマイコンはデータをもとに何らかの動作を行い．ワンライナーが動く機器に結果を送り返し (1) に戻る．

すなわち，このワンライナーにより，ttyX に接続された装置と ttyY に接続された装置とは，script1 と script2 を介して双方向接続している．

本報では，このワンライナーで特徴づけられる「ものグラミング 2」が，ものづくりに適した簡単に素早くかつ安全なプログラミングを提供することを示す．

## 2. 背景と関連研究

### 2.1 背景

ものグラミングは，もともとは趣味で電子工作を楽しむものづくり愛好家に，自分の作品とクラウド上のサービスを連携させるプログラムを簡単に手早くかつ安全に構成してもらうことを念頭に著者のひとり（大野）が提案した手法である．

趣味で行うものづくりで製作するものは作品であって工業製品ではないのが普通なので，ハードウェアの信頼性や安全性，さらに製品寿命といった問題についてはあまり配慮しない場合が多い．しかし，業務で製品化する場合には設計開始当初から信頼性や安全性に十分配慮しなければならない．この違いのため，趣味のものづくりで行われているものづくりの方法論は，業務での製品化には通用しないことが少なくない．一方，ソフトウェアの構成方法は，安全安心なプログラムが短時間で簡単に生み出せるのなら，電子工作愛好家向けに提案された手法なのか当初から業務を念頭において生み出されたのかといった違いは問題にならない．そこで，「ものグラミング」手法を実際にさまざまな現場で試してみると，趣味のものづくりだけでなく，業務で IoT デバイスを使った装置を短時間で試作したり，実際に運用したりする状況でも有効に機能することに気づいた．

一般論として．ソフトウェアは (1) 簡単かつ素早く開発でき，(2) 完成後も長期に渡って必要な性能を維持でき，(3) 開発後 10 年あるいは 20 年という長い時間が経過していても修正が必要になったら即座に修正できることが望ましい．しかし，実際には時間の経過とともに，開発したソフトウェアを取り巻く OS やミドルウエアやアプリケーションの仕様や実装が変化するため，(2) や (3) の実現は容易ではない．このような状況の下，テキスト形式のデータ処理を行う分野で (1) から (3) を満たすことに成功したのが，著者の一人（松浦）が主導的に進めている「POSIX 中心主義」に基づくプログラミングである [2].「ものグラミング」は POSIX 中心主義の影響を強く受けながら生まれた．そのため，センサから得た情報を処理してクラウド上のサービスに送ったり，クラウド上のサービスからの指示をアクチュエータに渡すといった IoT 分野ではありがちな処理を行う際には，POSIX 中心主義から極力逸脱しないようにしている．

POSIX 中心主義に基づくプログラミングでは，開発効率と処理効率の両立，互換性の増加，インストール・メンテナンスコストの抑制を実現するため，POSIX の仕様から逸脱しないように書かれたシェルスクリプトを用いる．ものグラミングにおいてもこれに習い，例えばセンサからデータを受けるのであれば，データを受け取った直後からデータ処理は全て POSIX 中心主義に基づいて書かれたシェルスクリプトで行うし，何らかのデータをアクチュエータに送るのであれば，アクチュエータにデータを送る直前までは，POSIX の仕様から逸脱しないように書かれたシェルスクリプトに担当させる．こうすることで， IoT 機器を取り扱うプログラミングでも，これまで POSIX 中心主義で対応してきたデータ処理と同様の高い互換性と長い持続性が確保できる．これが「ものグラミング」の根底にある考え方である．

### 2.2 用語

本報における「ものグラミング」以外の重要な用語には以下がある．

**ものづくり** ものを作ることがものづくりであるが，本報で特段の断りなく「ものづくり」と記した場合には，趣味で行う何かを作る活動であって，かつ電子回路を用い，さらに電子回路中には 1 台以上の組み込みマイコンが含まれる作品を作る行為を指している．

**POSIX 機** POSIX 系 OS が稼働するコンピュータの総称．Raspberry Pi のような組み込みコンピュータでも POSIX 系 OS が動いていれば POSIX 機であるし，Apple 社が提供する macOS が動くコンピュータも POSIX 機である．また，Windows10 搭載機であって，WSL(Windows Subsystem for Linux) が動いてい

れば，これも POSIX 機である．[*1]もちろん，BSD 系 UNIX や各種 Linux を搭載したデスクトップパソコンも POSIX 機である．

**POSIX 中心主義**　上述のとおり

**マイコン・組み込みマイコン**　現代的なマイクロコントローラを搭載した組み込み用コンピュータは，「ものグラミング 2」で好んで採用している 8bit 系 Arduino と比べるとはるかに多機能・高性能な機種も多い．これらは，単体で WiFi, Bluetooth I2C, SPI, UART 等のインタフェースを持ち，32bit CPU と数 MB かそれ以上のメモリを持つ場合がある．一方，Arduino UNO R3 に搭載されている ATmega328PU は 8bit CPU で，約 30KB のプログラム格納領域と 2KB RAM 領域を持つにすぎない．通信インタフェースも限定的であり，WiFi や Bluetooth 等は搭載していないため，拡張シールドを用いない限り実用的な対外通信方式は UART のみとなる．

これ以外の　OS，通信方式，電子機器に関連する用語は，それぞれの開発者による説明や定義をそのまま受け入れ，改めて定義することなく用いている．

### 2.3 関連研究

POSIX 機であるか否かにかかわらず，マイコンと USB シリアル接続したパソコンからマイコンを制御する方法にはさまざまな方法がある．実用化の域に達し公開されている方法の一つに Firmata がある．Firmata は Arduino をパソコンから制御するため，Arduino にあらかじめ専用のスケッチを書き込んでおき，特定のコマンドを受付可能にしておく．そしてパソコン側ではこの Arduino にコマンドを送ることで，Arduino のアナログ入出力，デジタル入出力を操作する．Firmata はプロトコルが規定され公開されているので，誰でもパソコン上のインタフェースを準備できる．ただし，Firmata ではマイコン側はパソコン側の周辺機器になってしまうため，単体でも操作可能な上にマイコンとも連携できるという運用はできない．Firmata が対応していない操作はできないし，プロトコルがバイナリベースなのでテキストベースのプロトコルと比べると効率は多少良いが，開発時の操作性は劣る．

## 3. ものグラミング

ものグラミングは，著者らが 2016 年から 2017 年にかけて研究開発に取り組んだ「ものグラミング 1」と 2018 年から取り組んでいる「ものグラミング 2」がある．どちらも「ものづくりのためのプログラミング」であるが，両者は実現方法が異なり，「ものグラミング 2」は「ものグラミング 1」の後継ではない．よって現時点でも両者とも研究開発続いている．本報は「ものグラミング 2」の報告であるため，単に「ものグラミング」と表記した場合は「ものグラミング 2」を指す．

### 3.1 ものグラミング 1

ものグラミング 1 では，POSIX 機に Raspberry Pi を用い，同機の拡張ピンヘッダから利用できる GPIO，UART, SPI, 1-Wire 等を用いてセンサやアクチュエータを利用する形態をとる．Raspberry Pi の拡張ピンヘッダを使うのはありふれた方法であるが，これらのインタフェースを利用する際にデバイスに直接アクセスするプログラムを書き起こすのではなく，それぞれの I/O 方式に対応した基本機能を提供するコマンド（たとえば，I2C バスを介してデータを読み取る i2cget コマンド, 同バスを介してデータを書き出す i2cset コマンド，デジタル I/O ピンを制御する gpio コマンド等）を用意し，これをシェルスクリプトから利用する点に特徴がある．たとえば I2C バス上にある I2C デバイスのアドレスが 0x36 であるセンサの 1 番目のデータを読み取る場合には i2cget コマンドを用い，以下のようなコマンドを実行するかシェルスクリプトから呼び出す．すると読み出した結果が標準出力に現れる．

```
$ i2cget -y 1 0x36 0x01
```

ひとたびセンサからのデータがテキスト形式で標準出力に現れれば，あとはシェルスクリプトで必要な処理を施せる．逆方向，すなわち I2C デバイスにデータを送る i2cset コマンドや，1 ビットずつ独立してデジタル入出力を実施する gpio コマンドも同様の使い方をする．

コマンドを介してセンサやアクチュエータにアクセスするこの方法は，処理速度は犠牲になる．この処理速度の低下が問題となるか否かはどのような用途で使うかに依存する．もし数秒ないし数分に一度室内の気温を計測してクラウドにデータをアップロードするといった用途なら，シェルスクリプトからコマンドを起動し，コマンドを介してセンサにアクセスする方式でも処理速度に特段の問題はない．

### 3.2 ものグラミング 2

「ものグラミング 2」の特徴は，ものづくりのためのプログラミングにおける「適材適所」と「選択と集中」を徹底したことである．このため，センサやアクチュエータを操作する際には，Arduino UNO のようなシンプルな組み込みマイコンを採用し，通常はセンサやアクチュエータの数だけ用意する．そして一つのマイコンにはセンサやアクチュエータのうちの一つだけを接続する．センサを担当することになったマイコンはセンサからの値を読んだらシリ

[*1] 2019 年現在，WSL は他の POSIX 機と異なる挙動を見せることがある．これらのうちのいくつかは WindowsOS のサブシステムであることに起因するが，いくつかは単なる不具合である．不具合は，年に数回のペースで行われている更新を経るごとに改善されている

アルポートを介して POSIX 機にデータを送る作業だけを行う．アクチュエータを担当することになったマイコンは POSIX 機からデータを受け取ってアクチュエータを動かす作業のみを担当する．なお，いずれのの場合もマイコンと POSIX 機との通信はテキスト形式で行う．POSIX 機上での POSIX 中心主義に基づくデータ処理との親和性を確保するためである．

同じ POSIX 機に接続されたセンサやアクチュエータを互いに連携させたり，クラウドを介して他のネットワーク上の他の機材やサービスとも安全かつ確実に連携させるといった高度な機能の実現には，POSIX 系 OS を搭載したコンピュータを活用，マイコンには担当させない．

POSIX 機上で用いるソフトウェアは，POSIX 標準コマンド群とこれらを活用する POSIX 中心主義に基づくシェルスクリプト，そして慎重に選んだ少数の新規コマンド類に限定する．ものグラミングでは，これまでの POSIX 中心主義にもづくプログラミング同様，ruby や python といった汎用性が高く多くの利用者を獲得している人気のスクリプト言語は原則として用いない．長期にわたる互換性や保守性が保証できないのが理由であるが，やむを得ぬ事情がある場合にも POSIX 中心主義が主張する「交換可能性担保」を実現してから用いる．

通常，センサやアクチュエータは単独で動作が完結することはなく，相互に連携する相手が必要がある．たとえば，温度センサから読み取った温度情報は，液晶表示パネルに表示したり，アクチュエータの操作に反映させたりする．最近は機能を大幅に向上させた高機能マイコンが安価に利用可能になり，機種によっては 1 台のマイコンに複数のセンサやアクチュエータを接続した上で，自身も液晶表示器やタッチパネルを搭載し，さらにクラウド上の他の機器やサービスとも WiFi や有線 LAN 上に確保した TCP/IP 通信で連携できる場合が多い．すなわち，1 台で数多くの要求を満たすようなマイコンが普通に存在する．しかし「ものグラミング 2」ではこのような 1 台の組み込みマイコンで何でも実現するようなことは決して行わない．「ものグラミング 2」の方向はむしろ逆で，「適材適所」「選択と集中」を徹底させるため，仮にセンサが一つしかなくその値を読み取って液晶表示器に表示するだけであっても，単純構成のマイコン 2 台（温度センサ担当と液晶表示器担当）と POSIX 機 1 台の合計 3 台を投入する．この方法は，POSIX 中心主義に基づくこれまでのプログラムの延長にあり，2 台のマイコンの動作は，以下の動作を行うだけのきわめて単純なものとなる．

**センサ担当のマイコン：** センサの値を読んでシリアルポートに出力する，

**表示器担当のマイコン：** テキスト形式のデータを受け取って液晶表示機に表示する

現在では，多くのセンサやアクチュエータや表示器は製造者あるいは先行する実装者から動作確認用のサンプルプログラムが Arduino 用に提供されていることが多い．上記のプログラムが要求する機能は，サンプルプログラムと同じか多少の改造で実現できる．C 言語や C++言語で記述するなら数 10 行で書き切れる規模である.，また，センサや表示器が変更になってもこの程度のプログラムであれば，再実装は容易である．また，POSIX 機の作業は，cu コマンドで一方の tty ポートからデータを読み取り，シェルスクリプトを通したのちに，もう一方の tty ポートに別の cu コマンドを用いてデータを送るだけである．すなわち以下のワンライナーで実現できる．

```
$ cu -l /dev/ttyX | script.sh | cu -l /dev/ttyY
```

シェルスクリプト script.sh は，センサからテキスト形式で得られた値を加工して表示器を担当するマイコンが要求する形式のテキストデータに変換する．テキスト処理はシェルスクリプトの得意とする分野であり，変更は容易である．たとえば，センサが温度センサであって，表示器に摂氏（° C）の温度を表示していたとする．これを華氏（° F）表示に変更したい場合も，マイコン上のプログラム変更は不要で，シェルスクリプトの変更だけでよい．現在時刻も表示したい，表示時刻を日本時間ではなく指定した地域の時間帯を用いたい，得られた気温を他の機材に MQTT を介して送りたい，といった要求にもシェルスクリプトの変更だけで対応できる．そして POSIX 機であれば，これらの変更を遠隔地から SSH を介して安全に実施することも容易である．

さらに，安全性について考えるとこの方法には以下の特長がある．組み込みマイコンに必要な機能は限定的である．このためソースコードは短く見通しがよい．C 言語や C++言語で 20 行程度で完結し，ソースコード全体がパソコンの 1 画面に収まることも珍しくなくソースコードレビューが容易に行える．プログラムはシリアルポートとテキストをやり取りするだけなのでセキュリティ上の脅威が入り込む余地が全くない．ネットワークを介した他の機材との通信は，暗号化等の安全確保を含めてすべて POSIX 機が担う．実際には Raspberry Pi のような小型のカードサイズのコンピュータを用いることが多いが，Raspberry Pi は小さくても POSIX 機である．POSIX 機であれば，OS の基本機能や多種多様なライブラリやサービスの安全確保については常に議論されており，問題が見つかれば直ちに議論され修正が配布される．組み込みマイコンは独自の拡張回路や素子が接続される独自の機材となることが多く，その安全確保は開発者以外は対応できないことがある．一方，POSIX 機は，汎用的なコマンドを組みあわせたシェルスクリプトを POSIX という汎用的な OS の上で動かすだけなので，組み込みマイコンと異なり，開発者以外とも

情報交換可能な汎用的な環境の汎用的な安全確保の問題となる．マイコンと POSIX 機会で担当する機能を分け，それぞれに得意な作業に特化させるのが「ものグラミング 2」における選択と集中であり，これが情報セキュリティ上の安全確保に寄与している．

このように，センサの値を液晶表示機に表示するために 3 台のコンピュータを投入する方法は一見すると高コストに思えるが，高い柔軟性と安全性を確保できる．さらにこの方法は POSIX 中心主義に沿っているあめ，20 年以上先であっても動作し，容易に改修でき，安全性であることを担保できる．したがって，中長期的な視点で見れば無駄でも高コストでもなく，保守性に優れた性能価格比に優れた方法である．

あらためて，図 1 に上記の例がが提供する標準的な接続形態を示し，ものグラミング 2 における標準的な実装についてまとめる．

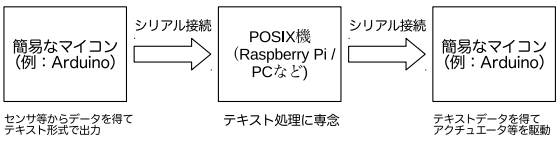


**図 1** ものグラミング 2 における標準的な機器構成

この図では，POSIX 機を中心に左右に「簡易なマイコン」を配置して USB 接続している．情報の流れは左から右への一方向である．

この図が意味することは以下の 3 点である．

(1) センサを接続した左側のマイコンは，センサから情報を得てこれをテキスト形式で POSIX 機に情報を送る．

(2) POSIX 機は左側のマイコンから受け取ったテキスト形式の情報を処理して右側のマイコンに送る．これは POSIX 機が得意とするテキスト処理である．

**例:** 「温度がある値を超えたらモータを動かす」のであれば，左側のマイコンから受け取ったテキストを分析して条件が成立したら，モータを動かす指示をテキスト形式で生成して右側のマイコンに送る．

(3) アクチュエータを操作するための指示をテキスト形式で受け取った右側のマイコンは，実際にアクチュエータを操作する．

これを本報冒頭のワンライナーと同じ記法で表現したのが以下となる．これは 3 つのコマンドがパイプで接続されており，図 1 における，左側のマイコン（POSIX 機とシリアル接続している），POSIX 機，右側のマイコン（同）という並びに対応している．これは直上のセンサと液晶表示を用いた事例と全く同じ内容のワンライナーであり，異なるのは標準入力からテキストを読み込んで標準出力にテキストを出力する script.sh の中身だけである．

```
$ cu -l /dev/ttyX | script.sh | cu -l /dev/ttyY
```

図 1 におけるマイコンは POSIX 機とシリアル接続しているが，通信は左側から右側への一方向である．こうすることで，左側のマイコンはシリアルポートにデータを書き出すだけで読み込まず，右側のマイコンはシリアルポートからデータを読み込むだけで書き出さない．これによりどちらのマイコンのプログラムもとても簡素になる．しかし，2 台のマイコンがそれぞれ POSIX 機を介して双方向通信したい場合もコマンドラインはそれほど複雑にはならない．図 1 における 2 台のマイコンの通信が双方向になると，センサを担当するマイコンは計測したデータを POSIX 機に送り出すだけでなく，POSIX 機から指示を受け取れるようになり，計測間隔や前処理の有無等を変更可能になる．アクチュエータを担当するマイコンは，POSIX 機からの指示に従ってアクチュエータを動かすだけでなく，動かした結果を POSIX 機に送ることができ，POSIX 機はアクチュエータの動作結果を把握できる．この双方向性を実現するスクリプトでは /dev/ttyX の双方向通信には script1.sh を /dev/ttyY の双方向通信には　と script2.sh を用意し，script1.sh と script2.sh はプロセス間通信させる．script1.sh, script2sh はシェルスクリプトなので，内部で様々なコマンドを起動でき，プロセス間通信は容易に実現できる．すなわち データの流れは以下になる．

```
/dev/ttyX → script1.sh →  /dev/ttyY
/dev/ttyY → script2.sh →  /dev/ttyX
script1.sh ←→ script2.sh
```

これをさらに整理し，以下の流れにしたのが本報冒頭のワンライナーである．

```
/dev/ttyX → script1.sh →  /dev/ttyY
/dev/ttyY → script2.sh →  /dev/ttyX
```

この場合，/dev/ttyX から script1.sh　を経て /dev/ttyY に向かう流れは，本稿で繰り返し示したように パイプを使って以下のように簡単に書けるが，同時に /dev/ttyY から /dev/ttyX へ向かう逆方向の流れの記述は一工夫必要である．

```
$ cu -l /dev/ttyX | script1.sh | cu -l /dev/ttyY
```

「ものグラミング 2」では，あらかじめ mkfifo コマンドで名前付きパイプ（ここではパイプ名を NP とするが実際に

は適切なフルパス名を用いる）を用意した上で 以下のようにしている．

(1) cu コマンドで /dev/ttyY から読み込んだデータは通常のパイプで script2.sh の標準入力に送る
(2) script2.sh の出力は通常のパイプではなく名前付きパイプに送る．
(3) /dev/ttyX を担当する cu コマンドの標準入力が上記の名前付きパイプから読み込むことで script2.sh の出力を /dev/ttyX に送り込む

この部分を記述すると以下になる．

```
$ cu -l /dev/ttyX < NP
$ cu -l /dev/ttyY | script2.sh > NP
```

これらをまとめたのが本報冒頭のワンライナーであり，「ものグラミング 2」の特徴を表現している．

```
$ cu -l /dev/ttyX < NP | script1.sh \
    | cu -l /dev/ttyY | script2.sh > NP
```

## 4. 比較

### 4.1 ものグラミング 1 とものグラミング 2 との比較

ものグラミング 2 の，ものグラミング 1 に対する主たる長所と短所を挙げる．

長所　ものグラミング 1 では，全てのセンサやアクチュエータを POSIX 機に接続するが，センサやアクチュエータの中には，POSIX 機よりマイコンに接続した方が効率よく接続できる，あるいはマイコンでないと接続できない場合がある．

長所　センサやアクチュエータなど，実機に必須な機能はマイコンに，通信やセキュリティといった汎用的な機能は POSIX 機にといった役割分担が明確にでき，それぞれの機材はそれぞれの役割に集中できる．

短所　POSIX 機単体では対応できず必ずセンサやアクチュエータの数だけ Arduino UNO のようなマイコンを必要とする．しかしこの点については，直上の長所を得るために必要である．よって本件は大きな欠点ではない．

### 4.2 多機能・高性能マイコンに全てを担当させる方法との比較

IoT デバイスを操作する装置には，多機能・高性能な組み込み用マイクロコントローラを使う事例が増えている．近年の 32bit マイコンであれば，多数のセンサデータを受け取り，高速で処理した後に WiFi ネットワークを介してクラウド上のサービスに TCP/IP で送信したり，クラウドからデータを得て多数のアクチュエータを制御する，といった処理を 1 台でこなせる場合が少なくない．これらのマイコンではもはやアセンブラ言語を直接使うことはなく，場合によっては C 言語で書くことすらなく，micro Python のような高機能なスクリプト言語で対応することがある．しかし，ものグラミングでは，単一のマイコンになんでも詰め込むのではなく，単機能のマイコンと POSIX 系 OS を搭載したコンピュータを組み合わせて対応する．

この比較では，ものグラミング方式に多くの長所がある．

長所　マイコンが単機能なためソフトウェアも単純になる

- 単純なプログラムなのでセキュリティ上の問題が生じうる箇所がほとんどない
- 単純なプログラムなのでソースコードの全貌を用意に把握できる
- 当該装置に機能の追加・改修が生じた場合，POSIX 機側の改修で対応できることが多い
- センサやアクチュエータは Arduino 向けのサンプルコードを用意していることが多い．このコードはそのままあるいは最小の修正でものグラミング用に活用できる．
- 将来，プログラムを改修しようとした際，もとになるプログラムが見つけられなくても，同等のスケッチを再現することは容易．

長所　クラウドとの通信を安全確実に行う部分は POSIX 機が基本機能を組み合わせて対応する

- 汎用の OS なのでセキュリティ

短所　POSIX 機あるいはマイコン単体では対応せず必ずセンサやアクチュエータの数だけ Arduino UNO のような単機能のマイコンを必要とする．

総じて，ものグラミングにおいては，プロジェクトに依存する部分，あるいは固有の素子の依存する部分は単機能のマイコンに，1 マイコンあたり 1 機能（あるいは 1 素子）で押し込めてしまい，それ以外は，POSIX 機上の汎用的なテキスト処理として対応する．これによって，機能の選択と集中を実現し，POSIX 中心主義に基づいたプログラミングを可能にする．その結果，安全と安心と長期運用性を確保する．

## 5. 実装

ものグラミング 2 は POSIX 中心主義に極力従うので，シェルスクリプトを書く際に利用できるコマンドも限定される．

マイコンが生成した文字列を POSIX 機がシリアルポートを介して受け取る際には cu コマンドを用いる．cu コマンド自体は POSIX が規定するコマンドではないが，UUCP による組織間通信が全盛だった 1980 年代前半から存在し，大きな変更は加えられていないため POSIX 系 OS であれば仕様や挙動の違いはない．また POSIX 機が生成した文字列をマイコンに送る際にも cu コマンドを使う．すなわ

ち cu コマンドはものグラミング 2 における POSIX 機とマイコンを繋ぐ唯一のインタフェースである．

POSIX 中心主義では，非 POSIX コマンドを使う場合には，交換可能性担保が必要である．ものグラミング 2 では必須となる cu コマンドが非 POSIX コマンドに該当する．

cu コマンドは重要な役割を担うが，もともと遠隔地にある POSIX 機にアクセスするためのコマンドであるため，ものグラミングでは不要な機能も多い．したがって，ものグラミングにとって必須の機能だけを実装した，cu コマンドより大幅に小さくしたコマンドを実装することは検討に値する．なお，既存のオープンソースなフリーソフトウェアの中では，jerm が cu コマンドの代替になる．jerm は余計な機能がなく，C 言語で書かれており，ソースコードの規模は cu の数分の一である．この jerm によって，cu コマンドの交換可能性が担保される．

POSIX 機上でテキストデータを処理するシェルスクリプト内でもっとも利用頻度が高いのは awk コマンドである．awk は POSIX コマンドである．

シェルスクリプトでの処理を終えてネットワーク上にデータを送る場合やその逆のには，nc コマンド（netcat コマンド）や mosquitto コマンド群（MQTT クライアント）を利用する．これらは POSIX コマンドではないが交換可能性担保は容易である．

この他，頻繁に利用するコマンドには，本報著者の松浦が開発し普及啓発中の，標準入力から読み込んだ文字列をツイッターに送る（またはその逆を行う）シェルスクリプト群の kotoriotoko（小鳥男）[4]，大野が開発中の Gmail からメールを送出する tegamiotoko（手紙男）等がある．

## 6. 普及啓発

これから IoT 分野でプログラミングを行う者にものグラミング 2 の特徴と有効性を伝えることは重要だと考え，普及啓発活動を開始した．この活動は国内だけでなく海外でも実施する方針で，一部は実現している．

### 6.1 日本での実践

2019 年 2 月現在，日本国内では，大学コンソーシアム石川のいしかわシティカレッジの講義「クラウド時代の「ものグラミング」概論」および金沢大学の学部 3 年生向け講義「計算科学特論」（いずれも 90 分 15 回の講義）で，ものグラミング 2 を取り上げ，演習中心に実施している．学生のは POSIX 系 OS が利用可能なノートパソコンの持参を指示し，マイコン 2 台とマイコンに搭載するセンサや LED ライト等は講義をする側が用意した．大学コンソーシアム石川での講義は，第二著者の松浦，第三著者の森も参画し，金沢大学の講義では森が参画し，講義の録画を含めた支援を担当した．録画した講義は今後の講義を計画する際の資料として用いている．

### 6.2 カナダでの実践

著者のひとり（大野）は，2018 年 5 月から 9 月までカナダ国ノバスコシア州立ダルハウジー大学にサバティカル滞在した．滞在中は滞在先の研究室でものグラミング 2 を実践し，この方法を主に大学院修士課程の学生に普及させるとともに，ものグラミングに関する 90 分の講義を 1 度実施した．さらに同年 11 月末から 2 週間に渡って再び訪問し，滞在中に 2 日間のものグラミングワークショップを開催した．このワークショップは両日ともに約 2 時間が割り当てられた．参加を希望した約 40 名の学生を 8 つのチームにわけ，各自にはノートパソコンを持参させた．初日は，ものグラミングに必要な環境構築のための説明と実習，さらにものグラミング方式の具体的な説明に充て，2 日目は参加者に実際に操作させた．参加者は，こちらが用意して配布した Arduino とアドオンボードを用い，ノートパソコンから cu コマンドで LED を点滅させたり，kotoriotoko にメッセージを送信することを体験した．このワークショップそのものは 2 日間で終了したが，参加チームのうちの一つが後日独自に課題を設定してものグラミング方式に則った作品を作って，報告してきた．報告の時点で著者はすでに帰国していたので，彼らは著者がサバティカル滞在した研究室で実機を用いたプレゼンテーションを行い，著者もビデオ会議システム経由で報告を受けた．当該作品は，2 台の Arduino を用い，グループの 1 名が自宅で飼う猫の餌の消費量をセンサで確認し，適宜 kotoriotoko で報告するというものであった．この作品は，Arduino 側では正味 30 行程度のスケッチで，またパソコン側では，10 数行程度のシェルスクリプトと 1 行のワンライナーで構成されていた．ものグラミング方式が理解され，実際に利用可能であることを理解したことが確認できた．

### 6.3 インドでの実践

ダルハウジー大学にサバティカル滞在を開始する 2 ヶ月前の 2019 年 3 月には，インド国ハイデラバード州のインド工科大学ハイデラバード校を訪問した．最大の目的は同所で開催された国際会議において基調講演を行うことであったが，ものグラミングについては，当該基調講演で言及し，さらに国際会議開催後には大学院生向けのゼミや学部生向けのハンズオンを行なった．ただし，ここで述べたものグラミングは「ものグラミング 1」であった．同校には 2019 年 3 月以降の再訪を検討している．再訪の際には再び講義やワークショップを行うことになっているが，そこでは「ものグラミング 1」ではなく「ものグラミング 2」を取り上げる予定である．

## 7. 評価と今後の展開

ものグラミングが，IoT に興味を持つ学生に訴求することはインドでの体験でもその熱心な応答からも感じてい

たが，カナダでは上記のように学生が自主的に課題を設定して問題解決を図り，報告にくるという大きな反応があった．上述のようにカナダの学生は，ものグラミング 2 をよく理解して問題解決に取り組み解決した．まだ一例ではあるが，ものグラミングの有効性が示されたと考えた．これが，引き続き海外でもものグラミングの普及啓発に取り組む契機となった．一方で，ものづくり等を行ってきていない学生等にとっては，ものグラミングに至るまでの前段の知識が不足している場合があり，その不足部分を補うための教育や教材等が必要であった．そこで，今後の取組の 1 つとして，授業やワークショップ等でスムースにものグラミングを学習し，ものづくりを行えるようにするための教材整備を行うこととした．これら教材は，YouTube 等に掲載できるような動画形式で作成することを検討しており，e ラーニングや反転学習用教材として使用する．併せて，英語対応等も行い，国内だけでなく英語圏を中心に世界展開を目指す．

## 8. おわりに

本報では，ものグラミングの新しい方式となるものグラミング 2 についてその概要を述べつつ普及啓発活動やそこで得られた知見，今後の展開について述べた．ものグラミングは，選択と集中の考え方により，マイコンにはマイコンにしかできない作業をさせ，一般性の高いデータ処理は POSIX 機上のシェルスクリプトで対応させる．これにより，POSIX 中心主義に則った高い互換性と長い持続性を IoT 機材のプログラミングにも導入できることを示した．

**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学生からは，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．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, 森, 北口, 中村, 松浦, 石山, 當仲，ものづくりのための「ものグラミング」と実践的教育環境の構築, DICOMO2016, 1335-1340, 2016-07.
[2] 松浦智之, 大野浩之, 當仲寛哲，ソフトウェアの高い互換性と長い持続性を目指す POSIX 中心主義プログラミング，デジタルプラクティス 8(4)，352-360，2017-10-15.
[3] What is POSIX?，The Open Group (オンライン)，入手先 ⟨https://collaboration.opengroup.org/external/pasc.org/plato/⟩（参照 2019-02-04）
[4] 秘密結社シェルショッカー日本支部，恐怖！小鳥男 (オンライン)，入手先 ⟨https://github.com/ShellShoccar-jpn/kotoriotoko⟩（参照 2019-02-04）.