実世界と動画を連携した IoT プログラミング学習支援システム

川谷 知寛[1]　塚田 浩二[1]　栗原 一貴[2]

概要：近年，IoT が一般化している中で，Arduino や Raspberry Pi などの IoT デバイスの動作や仕組みを他者と共有する機会が増えてきている．この際，ソースコードやテキストのみではデバイスの振る舞いを伝えることは困難であるため，オンラインでのチュートリアル動画が広く利用されている．しかし，チュートリアル動画は，従来のテキストと比較すると，ソースコードとのリンクが弱く，動画上のデバイスの挙動の詳細理解が難しいという問題がある．そこで，本研究では，Web 上のチュートリアル動画に対して，時間軸に応じたスクリプト言語を埋め込むことで，特定のタイミングで IoT デバイスを制御できるシステムを開発する．本稿では，システムのコンセプトと機能，及び実装例を紹介する．

1. はじめに

近年，IoT(Internet of Things) のような，身の回りのものとインターネットを繋げたアイテムが一般化しつつある．この中でも，手軽に IoT デバイスに触れることができる Arduino や Raspberry Pi などを個人で利用する場面も少なくはない．また，小学校でのプログラミング教育が必修化 [6] され，授業でもブロックプログラミングを使用して回路やセンサに触れる機会もある．さらに，コロナ禍に伴い大学でのプログラミング演習のオンライン化も急速に進められており，オンラインでのプログラミング教育支援の必要性も増えている [5].

IoT プログラミング学習の特徴として，コードを書いて PC 上で実行する従来のプログラミング学習と違い，マイコンなどのデバイスの物理的な装置の振る舞いがあることが挙げられる．そのため，テキストや写真などの静的コンテンツではデバイスの振る舞いを伝えることは困難であるため，オンラインでのチュートリアル動画が広く利用されている．また，IoT プログラミング学習では，動画の内容やソースコード，ユーザの手元にあるデバイスや回路と言った複数の要素を意識して学習していく必要がある．したがって，初学者はこの複数の要素のどれを意識すればいいのか・どこから見ればいいのか，分からずに困ってしまうということがある．

さらに，チュートリアル動画は，ソースコードとのリンクが弱く，さらには動画とデバイスのリンクがないという問題がある．前者については，動画を見ながら対応のソースコードを探すのは見る場所が増えてしまうため効率が悪くなる．一方，動画上でソースコードを解説する場合は，製作者の手間もかかる上に，ユーザは直接コードを編集できないため，柔軟性に欠ける．後者については，ユーザは動画を見ながら自分の手元でデバイスを動かせないため，センサをどのくらい傾けたら数値が変化するかなど感覚が掴めない．このような理由から，ユーザは動画を視聴してデバイスの挙動を軽く理解することはできても，手元のデバイスでの再現を行うのは簡単ではないと考える．

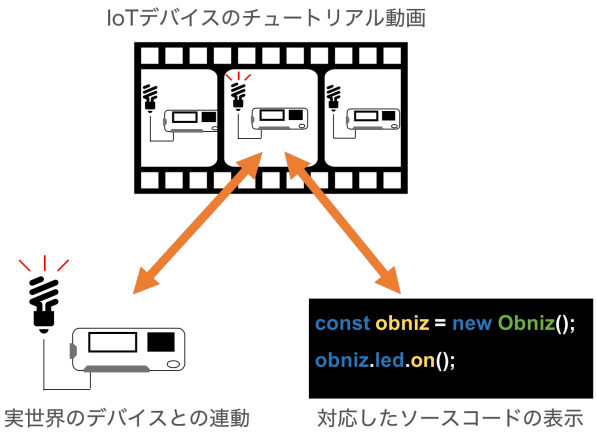

図 1　システムの概略図

そこで本研究では，IoT デバイスのチュートリアル動画にスクリプト言語を埋め込むことで，実世界の IoT デバイスと Web 上でのソースコード提示を制御し，動画視聴者

1　公立はこだて未来大学
2　津田塾大学

の学習を支援するシステムを提案・開発する．例えば，図 1 のような動画上のデバイスの動きに対応して，実世界のデバイス操作とソースコードの提示を行うことができる．システムを用いることで，動画視聴者のデバイスに関する理解度の上昇やモチベーションの向上といった効果が期待できる．

2. 関連研究

時系列コンテンツの特定時刻にスクリプトを実行することは，Adobe Flash の時代から広く用いられてきたが，その用途は主にアニメーションへのインタラクティブ性の付与等に限定されていた．Sikuli[7] は，プログラムに画像を使用して，自動化補助や検索を行えるようにしたシステムである．動画にスクリプトを関連づけて拡張する例として，Text Alive[1] や Songle Sync[2] がある．これらは，解析された音楽情報を基にした動画アニメーションの作成，スマートフォンなどの様々な実世界のデバイスと連動させることができる．

また，IoT デバイスと Web を連携する仕組みとしては，obniz Board[*1]や m5stack[*2]といったプラットフォームがある．これらは電気回路と直接接続可能なマイコンであり，クラウドを介して，Web 上からソースコードの記述や実行が可能なソフトウェアを備えている．このように，時系列コンテンツへのスクリプトの組み込みや，Web と実世界のデバイスを連動させる仕組みは存在していたが，両者を統合した試みはほとんど行われてこなかった．

そこで本研究では，時系列コンテンツである動画にスクリプトを埋め込み，Web と実世界のデバイスの連動を組み合わせたシステムを作成し，動画視聴者の理解度の向上と動画とソースコードのスムーズな連携を行う．

3. 提案システム

3.1 システムコンセプト

システムのコンセプトを図 2 に示す．大きく分けて，Web 上の要素である動画とソースコード，実世界の要素である IoT デバイスの 3 つの要素がある．ここで，動画の時間軸を中心として，ソースコードの表示や実世界の IoT デバイスの挙動を連携して制御する．ユーザは，Web 要素（動画とソースコード）を見ることによってデバイスの挙動と対応するソースコードの関係の理解しやすくなる．また，動画と連動する手元のデバイスを見ることによって，動画だけでは伝わらないデバイスの挙動を理解することができると考える．

3.2 システム要件

本研究では，チュートリアル動画視聴時における動画とデバイスの連動と，動画とソースコードのスムーズな連携ができるシステムの構築を目指す．そこで，以下の 2 点のシステム要件を満たす必要があると考えた．

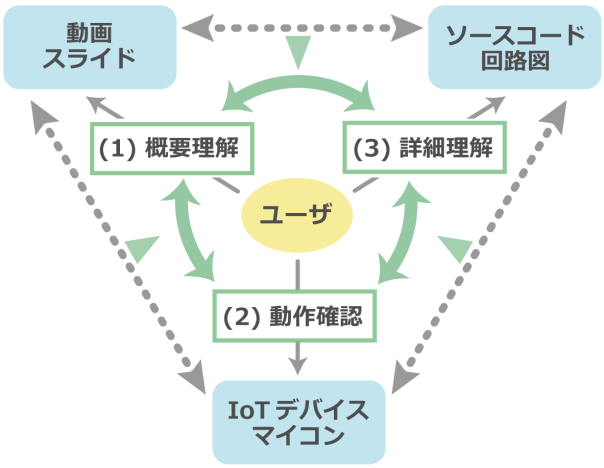

図 2 システムのコンセプト図

1. ユーザの手元でデバイスが動作する：動画上でデバイスの挙動を見るだけでなく，実際に手元のデバイスが同じ挙動をしたり，ユーザによる入力で動くことが詳細理解には必要だと考える．ユーザは手元のデバイスでプログラムを即時に実行することで，センサの感度やアクチュエータの駆動範囲等を体験的に理解することができる．また，動画と連動して手元のデバイスが動くという体験によって，IoT デバイスの初学者の学習意欲の向上，維持にもつながるのではないかと考えている．
2. デバイスの動作タイミングに合わせたソースコードの表示：1 章で挙げた通り，動画上のデバイスの動作がソースコードのどの部分実現されているか確認するのは難しい．そこで，動画上でデバイスが動いたタイミングで，Web 上のテキストエリアに対応するソースコードをコピーなどの編集操作が可能な状態で自動的に表示することにより，動画とソースコードの連携を行えると考えた．

4. 実装

本研究では，動画視聴者が使用する閲覧システムと動画製作者が使用する編集支援システムを作成した．この章では主に本研究のメインシステムである閲覧システムについて述べる．編集支援システムについては，5.2 で述べる．

閲覧システムのスクリーンショットを図 3 に，全体的なシステム構成図を図 4 に示す．システムは誰でもアクセスしやすいように，Web アプリで実装した．実装には，フロントエンドに HTML, CSS, JavaScript 等を用いた．バックエンドには，Google のクラウドサービスである Firebase を用いて，ホスティングとデータベースの管理を行った．

閲覧システムは，Web アプリ，YouTube 等の動画共有サービス，obniz 系マイコンといった要素から成る．Web アプリはソースコードエディタやデータベースを備えつ

[*1] https://obniz.com/ja/
[*2] https://m5stack.com

つ，外部からチュートリアル動画や動画に埋め込むスクリプトを読み込んでページに表示する．ソースコードエディタは，指定した言語の色付けなどを自動で行う CodeMirror という JavaScript のライブラリを用いた．

動画に埋め込むスクリプトとしては，YouTube の字幕ファイルに JavaScript を埋め込んで，特定のタイミングで実行する srt.js というフレームワークを使用する．srt.js については 4.2 章で詳しく述べる．図 5 の使用例にあるように，動画上で LED が光ったタイミングで，ソースコードエディタに対応するコードを表示しつつ，ユーザの手元にある LED を光らせるといった流れになっている．

図 3　閲覧システムのスクリーンショット

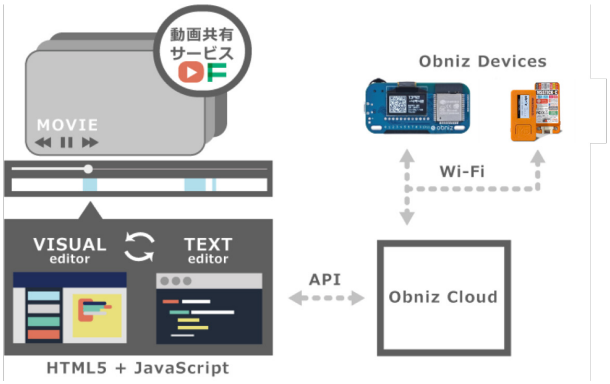

図 4　システム構成図

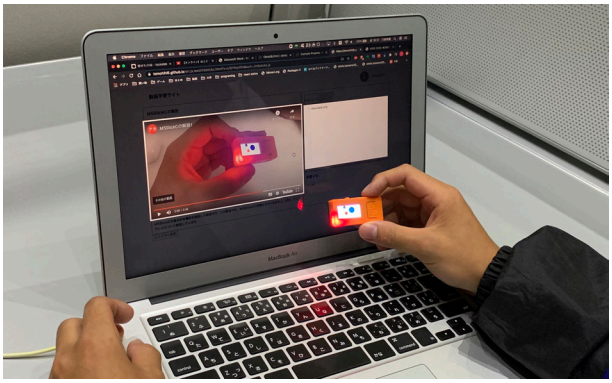

図 5　閲覧システムの使用例

4.1　使用する IoT デバイス

チュートリアル動画の題材として，前述した obniz 系マイコンの基本動作を解説したチュートリアル動画を作成する．obniz 系マイコンとは，obnizOS が導入された ESP32 型マイコンのことである．これらは，マイコンの ON/OFF やセンサ，IO 制御などをクラウドの API 経由で行えるため，Web 上から操作することが可能になっている．また，デバイス毎に登録されている ID によって即座に Web と繋げることができる．さらには，Web 上のエディタでコードを書き即時実行したり，node.js を使用してローカル環境での実行も可能である．obniz 系マイコンを使う理由としては，JavaScript によって Web 上から操作が可能なため，後述する「srt.js」との相性が良いことが挙げられる．

4.2　srt.js

動画の特定時刻と連動して Web 要素の操作や，IoT デバイスを操作する手法として，JavaScript のフレームワークである「srt.js」[4] を使用する．この「srt.js」は，動画の字幕情報を記述するフォーマットである srt 形式のファイルを拡張し，時系列に応じて JavaScript の実行を指示するフォーマットである．これによって，動画の特定のタイミングで対応するコードを実行することで動画やそれに伴うコンテンツを拡張することができる．動画上の何分何秒という時間を設定して，そのタイミングで実行するコードを記述していく（図 6）．書くタイミングで通し番号を設け，そこから改行までを 1 ブロックとしている．この機能を使って，デバイスを動かすタイミングやソースコードを表示するタイミング，ハイライトの切り替えなどを操作する．また，YouTube の動画を操作するための YouTube Player API を標準で実行できる．例えば，動画の再生や停止，指定の時間までスキップなどが行える．この機能によって，手元のデバイスに注目させたい時や，ソースコードを見て欲しい時などに動画の管理を行うことができる．閲覧システムでは，YouTube の動画 ID に応じた srt ファイルを外部データベースから読み込んでいる．

```
2
00:01:22,000 --> 00:01:23,000
displayBtnPressed("B");
const doc = editor0.getDoc();
doc.setValue(
`//ボタンBが押されたら
obniz.buttonB.onchange = (flg) => {
    //flgはtrueで押された，falseで離された
    if(flg) {
        obniz.display.clear();
        obniz.display.print("ボタンBが押されました")
    }
}`
);

3
00:01:25,500 --> 00:01:25,900
displayBtnPressed("A");
vars.btnAfunc[index] = function() {
    displayBtnPressed("A");
}
vars.btnBfunc[index] = function() {
    displayBtnPressed("B");
}
```

図 6　srt.js 形式ファイルの記述例

4.3 Web アプリ

Web アプリの構成は図 3 のようになっており，YouTube から埋め込まれた動画表示部分，ソースコード表示部分，動画概要欄などが付属している．そのうち Web 上の要素である動画とソースコードにおいては，図 7 のように注目させたいコンテンツをハイライト表示させることができる．ハイライト表示を切り替えるには，前述した srt.js に切り替えたいタイミングでハイライト対象を指定する．手元のデバイスに注目させる方法として，obniz の画像を Web 上に設置し, その画像を点滅させる簡易的な手法を用いた．手元のデバイスに注目させる手法に関しては，実世界の要素であるため，本機能で注目させることはできない．そのため，現在様々な方法を検討中である．検討中の要素については，5.1 で述べる加速度センサを利用して動画の操作を行う機能や，6.2 の議論において述べる．

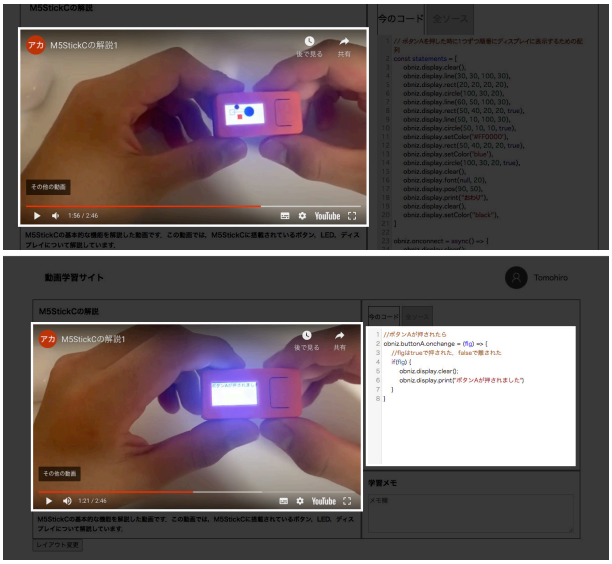

図 7 ハイライトの表示の例（上：動画のみ, 下：動画とソースコード）

次に，Web アプリの処理の流れを述べる．ページがロードされる際に URL に指定した YouTube から動画 ID に対応した動画を読み込む．その時，動画 ID に応じた srt ファイルを Firebase のデータベースから読み込んでいる．ページにアクセスした際に使用する obniz マイコンの ID を入力し，ID を元に obniz マイコンと Web アプリを obniz クラウドを介して接続する．接続を確認し動画を再生すると，srt ファイルが実行され，obniz のディスプレイへの接続確認，各センサや LED などの使用準備を行う．その後，動画の内容に合わせて任意のタイミングでソースコードやデバイスの操作，ハイライト切り替えを行う形になっている．

5. 応用例

5.1 IoT デバイスの状態に応じた動画の操作

手元のデバイスに注目しづらい要因の 1 つとして，ユーザが手元デバイスを触っている時でも動画の再生が止まらないことが挙げられる．そこで，デバイスに搭載されている加速度センサやジャイロセンサを利用して，ユーザがデバイスを手に取っている時は動画を停止し，手から離すと動画を再生する機能を作成した．実世界のデバイスの状態によって動画の再生状態が変化するため，本研究の目的の 1 つである “実世界と動画の連携” にもつながっている．しかし，この機能が使用できるのはデバイスに加速度センサが搭載されている場合に限られ，また動画を停止したままにしたい時でも，デバイスを触っていなければ勝手に再生されてしまうという制約がある．

5.2 動画編集支援システム

これまでは閲覧システムを使う側に注目してきたが，本章ではコンテンツを作る側の支援について述べる．まず，コンテンツの作成をする際は，

- 撮影者が事前にプログラムや IoT 作品を準備し，それらを解説しながら動画を撮影
- 完成した動画を YouTube にアップロードし，動画を見返しながら動画上の挙動に合わせて，対応するコードを記述
- 動画全体のコードを書き終えたら，srt 形式のファイルをデータベースにアップロード

という手順で行っていた．この srt 形式ファイルを作成するには，動画上でデバイスが動作した場合に動画を停止して，秒数や対応のコードをエディタで手動で記述していく必要があった．この作業は非常に手間がかかるため，これを補助する編集支援システムの開発を進めている．

編集支援システムのスクリーンショットを図 8 に示す．閲覧システムと同じく Web アプリであり，フロントエンド及びバックエンドにも同様の技術を使用している．画面下側にあるタイムラインは動画の時間軸と対応しており，タイムラインをダブルクリックすることで，クリックした場所に対応した動画の経過時間を自動で取得し srt ファイルを編集できる．画面右上にあるコードエディタは，左側に制作者が事前に作成したコードを表示し，右側に srt ファイルを記述できる．対応するコードをコピー&ペースト等で記述し，編集の時間短縮につながるようにしている．最後に “ファイルを書き出す” ボタンを押せば，専用のデータベースに srt ファイルがアップロードされる仕組みになっている．アップロード後に閲覧システムの URL が提示され，すぐに動作を確認できる．

6. 議論

ここでは，今までに作成したチュートリアル動画のコンテンツ事例を挙げつつ，動画作成者と視聴者からみたメリット，デメリットについて述べる．また，ユーザが動画と連動している手元のデバイスの動作を見逃してしまう問題についても議論する．

図 8 srt 編集支援システムのスクリーンショット

6.1 コンテンツの撮影方法

これまでに，通常，ライブコーディング，動画と実世界のデバイスが連動している様子の 3 種類 (図 9) を作成した．

1 点目については，動画とソースコードをわかりやすく見ることができる反面，動画と手元のデバイスがいつ連動するのかや，ユーザの入力が必要な場面がわからないといった問題があった．

2 点目については，動画上でソースコードの表示や口頭での解説があり，特定のタイミングで手元のデバイスがコードに対応した挙動をするため，コードとデバイスの相互理解に繋がりやすいと考える．しかし，コードを書きながら解説をしていくため，動画時間が長くなりやすく手軽さが失われてしまう．また，デバイスの挙動を伝えることができないため，複雑な IoT 作品になった時に動画本来の良さが失われてしまう可能性があると考えられる．

3 点目については，手元のデバイスがいつ動くのかはわかりやすくなったが，肝心の動画が見づらいという問題点があった．

以上の考察結果から，本研究のコンセプトに一番あっている 1 点目の撮影方法を採用しつつ，手元のデバイスに注目させる方法を整理／実装していきたい．

6.2 手元のデバイスへの注目

前述した通り，いつ手元のデバイスが動くのかわからず見逃してしまったり，ユーザの入力がいる場面がわからずに動画が進んでしまうという問題がある．これらの問題を解決する手法として，現在は Web 上に obniz の状態を示したエージェントを点滅させる手法（図 3 の右下にあるデバイスの写真が点滅）を用いている．今後は，エージェントの表現の拡張や 5.1 で述べた機能の改善，動画上で何らかのアクションがあるタイミングを．時間軸として Web アプリ上に表示しておくという手法を検討していく．

7. まとめと今後の課題

本研究では，IoT デバイスのチュートリアル動画にスクリプト言語を埋め込むことで，実世界の IoT デバイスと Web 上でのソースコード提示を制御し，動画視聴者の学習を支援するシステムを提案・開発した．今後も，システム

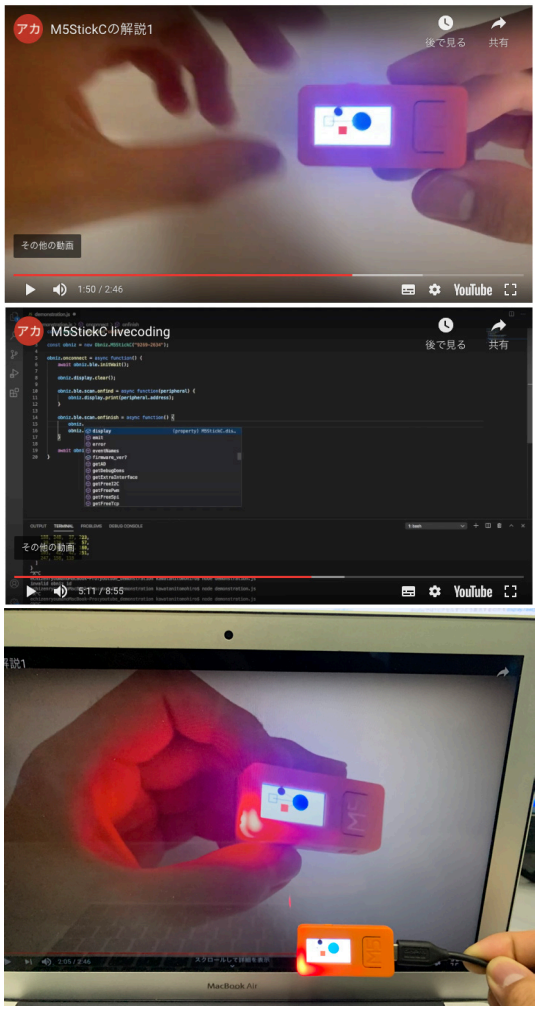

図 9 3 つの撮影方法の図．上から，通常，ライブコーディング，動画と実世界のデバイスが連動している様子

の UI や UX の改善を行い，システムに使用する動画コンテンツの拡張も行っていく予定である．これらのコンテンツは，Web プログラミングのためのチュートリアルに関する調査研究 [3] を参考に作成していく．また，現在の課題であるユーザの注目をどこに向けさせるかの問題や，コンテンツの利用者／製作者を支援する機能の拡張を引き続き検討していく．その後，実際にユーザを募ってシステムを使用してもらう評価実験を行っていく．

参考文献

[1] Kato, Jun and Nakano, Tomoyasu and Goto, Masataka.: TextAlive: Integrated Design Environment for Kinetic Typography, In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3403-3412, (2015).

[2] Kato, Jun and Ogata, Masa and Inoue, Takahiro and Goto, Masataka.: Songle Sync: A Large-Scale Web-based Platform for Controlling Various Devices in Synchronization with Music, In *Proceedings of the 26th ACM International Conference on Multimedia*, pp. 1697-1705, (2018).

[3] Kim, Ada. Suhkyung. and Ko, Andrew. Jensen.: A Pedagogical Analysis of Online Coding Tutorials, In Proceedings of the 2017 ACM SIGCSE Technical Symposium on

Computer Science Education, pp. 321–326 (2017).
[4] 栗原一貴, 橋本美香. srt.js:映像コンテンツへの IoT 指向拡張プログラム埋め込みフレームワーク, 日本ソフトウェア科学会 第 23 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS 2016) 論文集, pp.24-29, (2016）.
[5] 又吉 康綱, 中村 聡史. askTA : 消極的な受講生でも質問可能なオンライン演習講義支援システム, 第 28 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS 2020）, （2020）.
[6] 文部科学省（2020）：小学校プログラミング教育の手引（第三版）
[7] Yeh, Tom and Chang, Tsung Hsiang and Miller, Robert C.: Sikuli: Using GUI screenshots for search and automation, In *UIST 2009 - Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, pp. 183-192, (2009).