IoTeach: 実世界と順序型コンテンツを連携したIoT学習支援システム

川谷 知寛 * 塚田 浩二 * 栗原 一貴 †

概要. 近年，IoT が一般化している中で，Arduino や M5Stack などの IoT デバイスの動作や仕組みを他者と共有する場面が増えてきている．さらにコロナ禍に伴い，各教育機関においても，チュートリアル動画等を用いたオンデマンド型や，ビデオ会議システム等でのスライド共有を中心とするライブ型のオンライン講義の導入が進められている．一方，こうした手法は実世界のデバイスやソースコードとのリンクに乏しい等の問題があった．そこで本研究は，Web 上で共有される動画やスライドといった順序型コンテンツにスクリプト言語を同期させ，実世界の IoT デバイスや編集可能なソースコードと連携させることで，IoT プログラミング学習を支援するシステム「IoTeach」を提案・開発する．本稿では，システムのコンセプトと実装を説明し，コンテンツ事例や課題と展望について議論する．

1 はじめに

IoT（Internet of Things）という言葉に代表されるように，身の回りのモノとインターネットを連動したシステムが一般化しつつある．さらに，IoT 機器のプロトタイピングを支える Arduino，Raspberry pi，M5Stack などの要素技術も充実してきており，その教育ニーズも日増しに高まっている．しかし，試作した IoT デバイスの振る舞いや仕組みを，空間的・時間的にその場にいない他者と共有することは難しく，実世界での振る舞いをわかりやすく示すために，チュートリアル動画がしばしば採用されている．例えば，M5Stack 社は自社 IoT デバイスのチュートリアル動画を多数公開している[1]．

一方，チュートリアル動画は従来のドキュメントに比べると，ソースコード等の実際にデバイスを動作させる要素とのリンクに乏しいという問題がある．ユーザは動画を見てデバイスの振る舞いを理解することはできても，手元の実機で動作を再現するのは容易ではない．ソースコードのどの部分がその挙動を起こしているかは動画では表現しにくいため，従来のドキュメントを改めて参照してソースコードを探し，専用の開発環境から手元のデバイスに転送する必要がある．すなわち，チュートリアル動画は特に初心者のユーザにデバイスの振る舞いを伝える「概要把握」に優れる反面，実機での「動作確認」や仕組みの「詳細理解」には，従来型ドキュメントやソースコードをユーザが自力で参照する必要があり，学習プロセスとしてみるとギャップがあった（図 1）．

さらに，コロナ禍に伴い各教育機関ではプログラミング演習のオンライン化が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が増加している．オンライン授業は主に，（チュートリアル動画の例に近い）動画等を学習者の好きなタイミングで視聴するオンデマンド型と，ビデオ会議システム等を活用したライブ型に分けられる．ライブ型の講義においても，講師は学習者の手元にある計算機や IoT デバイスの状態の把握や制御が難しい等の問題を抱えている．

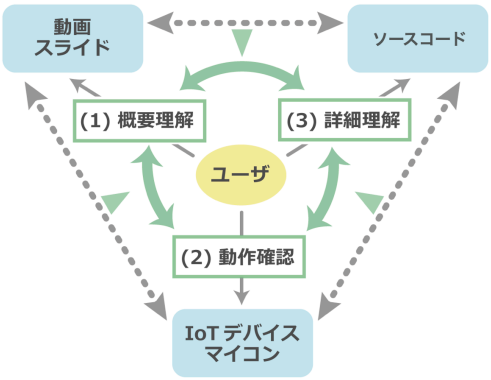

図 1. 本研究の狙い

そこで本研究では，Web 上で共有される動画・プレゼンテーション資料などの順序型コンテンツに対して，進行に対応して実行されるスクリプト言語を同期させることで，スクリプトと視聴者の手元の IoT デバイスとを連携可能なシステム「IoTeach」を構築する．例えば，図 2 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．IoTeach を用いることで，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を期待する．

* 公立はこだて未来大学
† 津田塾大学
[1] https://www.youtube.com/c/M5Stack/

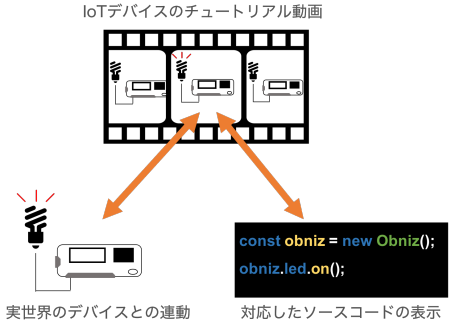

図 2. IoTeach の概要．Web 上の動画やスライド等の順序型コンテンツの進行に対応したスクリプトを同期させることで，実世界の IoT デバイスの操作や関連するソースコードの提示を行う．

2 関連研究

2.1 コンテンツにスクリプトを組み込む事例

順序型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[5] 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．

Text Alive[4] 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を行う Web サービスである．作成された動画はプログラミング要素を含んでおり，字幕の表示設定や図形の描画などをプログラムを書いてユーザ自身が実装できる．

ぴぴつい [6] は PowerPoint のスライドショーに連動して，事前に発表者ノートに記述した内容をツイートする PowerPoint のアドインである．

2.2 コンテンツと実世界デバイスを連動する事例

Songle Sync[3] は，音楽を元に作成された動画を実世界の様々なデバイスで制御して，音楽と物理空間が融合した体験を生成する．

また，IoT デバイスと Web を連携する仕組みとしては，obniz Board[2]といったプラットフォームがある．これ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．

2.3 オンラインプログラミング教育支援の事例

Progate[1] やドットインストール [2] は，オンラインで様々なプログラミング言語を学ぶことができる Web サービスである．スライドや動画での学習をした後に，サイト上でコードを書いて実践的に学ぶことができる．しかし，これらのサイトで IoT デバイスについて学習することはできない．

[2] https://obniz.com/

また，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．Typing.run[9] は，プログラミング学習のタイピング練習と基本命令や関数の理解と定着の支援を行う Web システムである．このシステムによって，利用者のタイピングスコアの上昇やプログラムの理解や定着に良好なことが確認された．Amadeus[8] は，学習者側だけではなく講師側の支援も行ったプログラミング初心者のためのオンラインコンパイラシステムである．学生側の環境構築の手間を省き，同時に講師は学生が困っている時にすぐに気づくことができるようになっている．

2.4 本研究の特徴

このように，順序型コンテンツへのスクリプトの同期や，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，両者を統合した学習支援の試みはほとんど行われてこなかった．また，オンラインプログラミング学習支援システムは多数存在するが，IoT デバイスに焦点を当てた支援はあまり行われてこなかった．本研究では，動画やスライド等の順序型コンテンツにスクリプトを同期させ，コンテンツ内のデバイスの動作を実世界のデバイスやソースコードと連携させることで，IoT プログラミング学習を支援する点で独自性がある．

3 提案

3.1 コンセプト

図 1 をもとに，IoTeach のコンセプトを説明する．大きく分けて，Web 上の要素である順序型コンテンツ（動画／スライド）とソースコード，実世界の要素である IoT デバイス（マイコン）の 3 つの要素がある．ここで，コンテンツの順序（例: 動画の時間軸）に基づいて，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，動画内のデバイスの動作と連動したソースコードを見ることができるため，制御の詳細を理解しやすくなる．また，動画を見るだけで手元のデバイスが動作する体験は，特に初学者ユーザの学習意欲の向上等に繋がると考える．

システムの利用例を図 3 に示す．ここでは，ユーザは PC 等で IoT マイコンのチュートリアル動画を視聴している．動画内のマイコンの LED が光ると，ユーザの持つマイコンの LED も連動して光る．さらに，対応するソースコードが画面右側のエディタ上に提示される．

3.2 システム要件

本研究では，順序型コンテンツ（動画／スライド）の視聴時において，コンテンツとデバイスの連動，

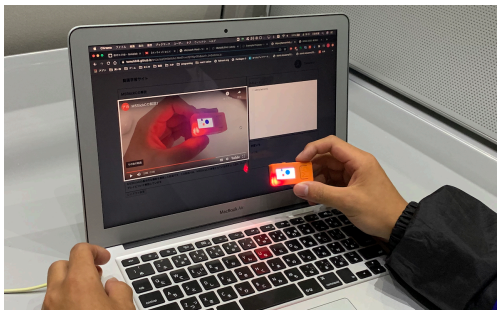

図 3. IoTeach の利用例

及び，コンテンツ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 3 点のシステム要件を設定した．

ユーザの手元でデバイスが動作
動画／スライド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で，理解を深める．例えば，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，動画を見るだけで手元のデバイスが動くという体験は，特に初学者の学習意欲の向上につながると考える．

デバイスの挙動に合わせたソースコードの表示
前述した通り，動画／スライド上のデバイスの動作がソースコードのどの部分で実現されているか確認するのは難しい．そこで，動画上等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等の編集操作が可能な状態で表示することにより，動画とソースコードのスムースな連携を行う．

注目箇所の誘導
IoT プログラミング学習では，コンテンツ（動画／スライド），ソースコード，手元のデバイス等，複数の要素を意識して学習する必要があり，どこを見ればよいのか，混乱が生じやすい．そこで，システムで注目すべき要素を強調表示することで，ユーザの注意を誘導する仕組みを構築する．

3.3 動画版システムとスライド版システム

本研究では，オンデマンド教育／ライブ教育を見据えて，動画／スライドの 2 種類の順序型コンテンツに対応したシステムを構築する．なお，学習者は主に，マイコンや IoT 開発の経験がない初学者を想定する．

動画版システムは，YouTube 等のオンライン動画を Web アプリ上に埋め込み，動画の時間軸に沿って処理を実行するスクリプトを連動させる．学習者は原則一人でシステムを利用するため，一つのアプリ（インスタンス）に対して，一つの IoT デバイスが対応する．オンデマンド教材以外にも，個人やハッカソン等で制作した IoT 作品の公開／共有にも活用できると考える．

スライド版システムは，PDF 等のページ単位を持つスライドをオンラインに配置し，ページ遷移に沿って処理を実行するスクリプトを連動させる．主にライブ型オンライン授業でビデオ会議システムと併用し，一人の講師が複数の学習者に対して，スライドを画面共有等で見せながら進行する形態を想定している．よって，一つのアプリ（インスタンス）に対して，各学習者の持つ複数の IoT デバイスが対応する必要がある．

4 実装

3.3 節で述べたとおり，本研究では動画版システムとスライド版システムの 2 種類を開発する．本システムのシステム構成図を図 4 に示す．両システムとも，Web アプリとして実装した．大まかな実装技術としては，HTML，CSS，JavaScript 等を用いてフロントエンドを構築し，Firebase[3]を用いて，ホスティングとデータベースの管理を行っている．以下，利用する IoT デバイスとスクリプトについて紹介した後に，動画版／スライド版システムについて述べる．

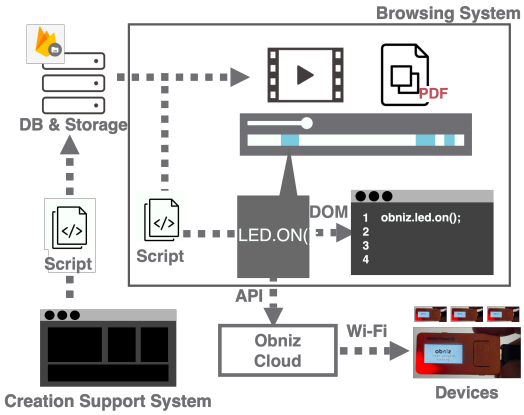

図 4. システム構成図

4.1 IoT デバイス

システムで利用する IoT デバイスの要件としては，

1. 外部のセンサやアクチュエータを接続する拡張性を持つ．
2. Wi-Fi 等でインターネットに接続できる．
3. インターネット経由でプログラムを実行／書き換えできる

[3] https://firebase.google.com/

といった点が上げられる．これらの要件は，ESP32 マイコンや Arduino+Wi-Fi シールド等でも達成可能であるが，3 点目の実現にはファームウェア開発等の手間が掛かる．

そこで，本研究では obnizOS が導入されたマイコン[4]を利用することにした．obnizOS は，IoT マイコン上で動作するファームウェアであり，デバイス毎にユニークな ID を備えている．クラウドの API から ID を指定することで，特定のデバイスに対して，内蔵センサ／アクチュエータの制御や，ポートの入出力制御を行うことができる．また，obniz 系マイコンの開発言語は，Web アプリで広く利用される JavaScript であり，次節で紹介する動画と同期して実行されるスクリプト言語「srt.js[7]」との相性が良い点も配慮した．

学習者は対応する obniz 系マイコンさえ持っていれば，複雑な準備を必要とせずに，本システムを利用できる．

4.2 コンテンツと同期するスクリプト

ここでは，動画版／スライド版システムと同期して動作するスクリプトについて説明する．どちらのスクリプトも，基本的には標準的な JavaScript と obniz ライブラリを用いるが，動画版では「動画上の経過時間」，スライド版では「スライド上のページ遷移」に対する記述方式を備える点が特徴である．

具体的には，動画版では「srt.js」という JavaScript のフレームワークを使用する．これは，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たものであり，動画上の経過時間に対応して特定のコードを実行できる．例えば，図 5 上部のように動画上の経過時間（例: 10 秒→ 15 秒）を記述して，次にその時間に実行するコードを記述する．ここで，経過時間の直前の通し番号から空行までが一ブロックとして認識される．

スライド版では，図 5 下部のように，関数の配列を用意し，スライドのページ番号をインデックスとして指定して，関数内にコードを記述する．関数は，各ページに遷移した（画面に表示された）タイミングで実行される．

これらのスクリプトは，コンテンツと紐付けて Firebase 上のデータベースに保存する．

4.3 動画版システム

動画版システムは，大きく Web アプリ，動画共有サービス（例: YouTube），IoT マイコン（obniz）から構成される．Web アプリとしては，閲覧アプリと作成支援アプリの二つを開発した．以下，詳細について述べる．

[4] https://obniz.com/ja/products/obnizos

```
0
00:00:00,000 -> 00:00:01,000
const obniz = new Obniz("input id here");
displayCaption("Hello World");

1
00:00:10,000 -> 00:00:15,000
obniz.display.clear();
obniz.diplay.print("Hello World");
```

```
const obniz = new Obniz("input id here");

slideFunction[2] = () => {
  obniz.display.clear();
  obniz.diplay.print("Hello World");
}
```

図 5. スクリプトの記述例（上：動画版，下：スライド版）

図 6. 閲覧アプリのスクリーンショット

4.3.1 閲覧アプリ

閲覧アプリのスクリーンショットを図 6 に示す．大きく，(1)YouTube から埋め込まれた動画表示部分，(2) ソースコード表示部分，(3) 字幕表示部分，(4) 出力コンソールの 4 つの要素から構成される．

ソースコード表示部分は，関数／変数等の色付けに対応したエディタライブラリである，CodeMirror を用いて実装した．コード表示部分は,「リアルタイム」「全体」という二つのタブを備えている.「リアルタイム」では，動画の時間軸（現在のデバイスの挙動）に対応したコードのみを表示する.「全体」では，一連の処理を行うコードをまとめて表示する．

字幕表示部分は，デバイスの動作の補足説明やユーザに注目してほしい箇所等を表示する．作成者は，srt.js の記法に沿って，コードと交えて字幕を記述することができる．出力コンソールは，手元のデバイスの動作（例: ボタンや加速度センサ）に対応したセンサデータ等を，リアルタイムに出力する．

また，これら 4 つの要素（動画／ソースコード／字幕／コンソール）について，注目させたい箇所をハイライト表示させることができる．ハイライト表示の例と，対応するコードを図 7 に示す．

4.3.2 作成支援アプリ

動画版システムでは，スクリプトに動画の経過時間を入力する必要がある．これを手作業でやる場合，動画とエディタを並べて開いておいて，動画上でデバイスの動作場面を探し，経過時間をエディタに記

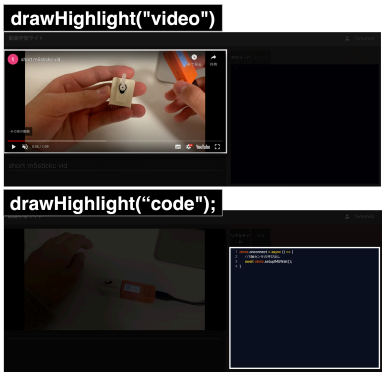

図 7. ハイライト表示の例（上：動画のみ，下：ソースコードのみ）

述する必要があり，手間が掛かる．

そのため，動画版システムのコンテンツ作成を支援するシステムを開発した．作成支援アプリのスクリーンショットを図 8 に示す．大きく，(1) 動画表示部分，(2) タイムライン，(3) コードエディタ（オリジナル），(4) コードエディタ（タイムライン）の 4 つの要素から構成される．

タイムラインは動画の時間軸と対応しており，タイムライン上の任意の箇所をダブルクリックすることで，動画の経過時間[5]を自動的に挿入したコードブロックを作成できる．さらに，コードブロックをダブルクリックすることで，右上のコードディタ（タイムライン）で開いて編集する．コードエディタ（オリジナル）には事前に作成したコードを読み込み，コピー＆ペースト等を用いることで，srt.js 形式のコードを効率的に編集できるように配慮している．

最後に,「アップロード」ボタンを押すことで，Firebase 上のデータベースに srt.js 形式のファイルがアップロードされる．アップロード後には閲覧アプリの URL が提示され，即座に動作を確認できる．

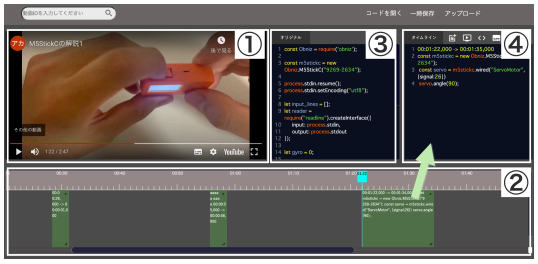

図 8. 作成支援アプリのスクリーンショット

4.4 スライド版システム

スライド版システムの作成にあたり，ページ遷移を API 経由で検出できるサービスが見つからなかったため，reveal.js[6] というライブラリを利用して，独自のスライドショーシステムを構築することにした．

図 9 にシステムのスクリーンショットと利用の流れを示す．まず，作成者（主に講師）は初期画面で，スライド（PDF），スクリプト（JavaScript），講師／学習者のデバイス ID の一覧（CSV）を Web アプリに読み込む．前述したように，スライド版システムでは講師の PC と全ての学習者の obniz 系マイコンを接続するため，CSV ファイルでまとめて読み込めるように配慮した．

3 つのファイルを読み込んだら，プレビュー画面に遷移する．プレビュー画面では，各スライドへの遷移時に実行されるコードを確認することができる．問題がある場合,「戻る」ボタンを押すことで，特定のファイルのみを差し替えることができる．

プレビュー画面から,「プレゼンテーション開始」ボタンを押すとスライドが全画面表示され，デバイスとの連動が開始される．ここで，特定のスライドが表示される（ページ遷移する）と，ページ番号に対応した関数が実行され，事前に読み込んだデバイス ID に対応する全ての obniz 系マイコンが連動して動作する．講師の負担を軽減するために，コード上では複数のマイコンに対する繰り返し処理は隠蔽され，自動的に補完されるように配慮した．

なお，アップロードされた PDF ／ JavaScript ／ CSV は Firebase 上のデータベースで管理され，PDF ファイル名とタイムスタンプをキーとして後から呼び出すことができる．

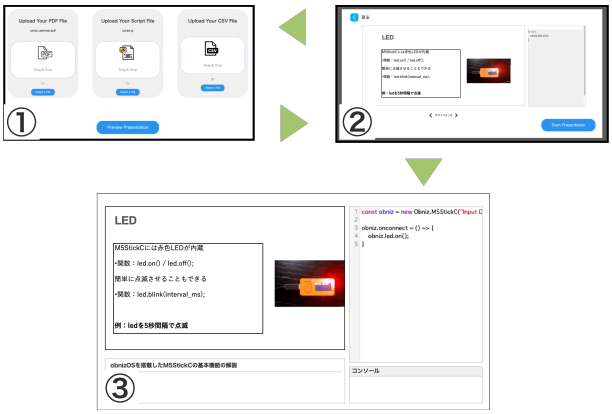

図 9. スライド版システム（1. 初期画面，2. プレビュー画面， 3. プレゼンテーション画面）

5 議論

5.1 コンテンツ事例

ここでは，主に先行して実装した動画版システムを中心としたコンテンツ事例について述べる．まず，

[5] 開始時間を動画の現在再生時刻，終了時間を現在再生時刻から 3 秒後としている

[6] https://revealjs.com/

コンテンツを作成するにあたり，基本的な撮影方法を検討した．通常／システム利用中の様子の 2 種類の撮影方法でコンテンツを作成した（図 10）．通常の撮影方法については，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りにくいといった問題があった．システム利用の様子を撮影した場合については，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．また，動画を二重に撮影する必要があるため，作成者の手間も増えてしまう．このように，それぞれの撮影方法は一長一短であるが，手間の少なさと動画の見やすさを考慮して，1 点目の撮影方法を基本的に利用し，手元のデバイスに注目させる方法を整理／実装していく方針とした．

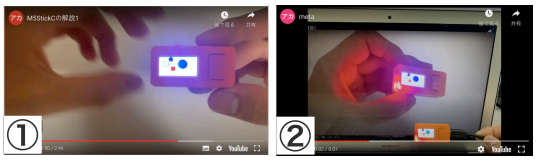

図 10. 2 つの撮影方法 (1．通常，2．システム利用中)

また，m5stickC のような加速度センサ／ジャイロセンサが搭載されているマイコンであれば，センサでデバイスの状態を検出して，システムの挙動に反映できる．一例として，ユーザがデバイスを手に取っている時に動画を停止し，手から離すと動画を再生するような機能を実装した．一方，この機能はセンサ搭載デバイスでしか利用できず，意図せぬ動作を引き起こす可能性もあるため，汎用的な手法も併せて検討する必要がある．

さらに,「LED を点滅させる」,「LCD で図形を描画する」,「加速度センサの値をコンソールに出力する」等といったマイコンの基本的な機能を紹介するコンテンツや,「ジャイロセンサの値に応じてサーボモーターを制御する」,「加速度センサやマイコンのボタンを用いてリモコンのように YouTube 動画を操作する」といった簡単なアプリケーションを提案システムを用いて作成した．内容の一部については，デモビデオでも紹介する．

5.2 システムの使用感と課題

本格的な評価実験はまだ行っていないが，これまでに著者らにより複数のコンテンツを作成し，所属研究室内で，動画版システムの予備実験とスライド版システムを用いた簡易ワークショップを実施した．詳細は紙面の都合で割愛するが，これらの経験に基づくシステムの印象や課題について議論する．

まず，コンテンツ閲覧時の使用感として，動画版システムでは，予備実験のアンケートからコンテンツに合わせてデバイスが連動する点や，動作に関連するソースコードを提示する点については好印象を得られた．スライド版システムを用いてライブ型の体験をしてもらった際には，スライドの切替に応じて各ユーザの手元のデバイスが連動する際に,「おおっ」という感嘆の声が複数上がっていた．

課題としては，動画版システムでは，現状では時系列通りに動画を閲覧しないと，変数の初期化等が正しく実施されず，予期せぬ動作を起こす可能性がある．よって，体験者にはスキップ等の動画の操作をなるべく利用しないように依頼したが，ストレスであるという声も上がった．対策として，全てのスクリプトの実行前に自動的に実行される初期化用コードを用意したり，スキップした間のコードを自動的に実行するような仕組みを検討している．また，スクリプトの埋め込んである時刻をタイムライン上で可視化するような機能の実装も予定している．

次に，著者らがコンテンツ作成を通じて感じた印象や課題について述べる．動画版では，当初手動で通し番号や動画の経過時間を記述していたため，作成支援システムを開発したことで，こうした労力は大分減ったと感じた．一方，ユーザにコードを提示する場合は実行用／表示用で 2 度同じコードを書く必要があるため冗長であると感じた．また，動作確認の際に閲覧アプリと作成支援アプリを何度も切り替える必要があり，少々面倒であった．今後は，コードを手軽に表示できる記法や，作成支援システム上での文法チェックや簡易的なデバッグを行えるように改良していきたい．

スライド版では，一人の講師が複数の学習者に対して利用するが，学習者側のデバイスの動作状況が分かりにくい問題があった．今回は，音声やチャット等でコミュニケーションをとったが，今後は学習者側のデバイスの動作状況（例: 入出力ポートの状態）等を手軽に把握できるような視覚化機能の実装を検討していく．また，現時点では講師のスライドと全ての学習者のマイコンが連動するため，個別のトラブルシューティング等を行う際には不便である．今後は，スライドとマイコンの接続関係を手軽に変更できるような機能と UI の実装を進めていきたい．

6 まとめ

本研究は，動画やスライド等の順序型コンテンツにスクリプト言語を同期させ，実世界の IoT デバイスや編集可能なソースコードと連携させることで，IoT プログラミング学習を支援するシステム「IoTeach」を提案・試作した．さらに，動画版を中心としたコンテンツ事例や，動画版／スライド版システムの使用感と課題を議論した．今後は議論で述べたシステムの改善を行った上で，システムの有無での学習効果の変化やモチベーション向上について検証を進めていく．

参考文献

[1] Progate. https://prog-8.com.

[2] ドットインストール-3 分動画でマスターできるプログラミング学習サービス. https://dotinstall.com/.

[3] K. Jun, O. Masa, I. Takahiro, and G. 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.

[4] K. Jun, N. Tomoyasu, and G. 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.

[5] Y. Tom, C. T. Hsiang, and M. R. 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.

[6] 栗原一貴. 学会イベント支援 2) 放送化の時代のプレゼンテーション支援システム. 56(5):465–471, 2015.

[7] 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 WISS2016 論文集, 2016.

[8] 森田浩平, 松本慎平. 大学でのプログラミング基礎講義における初学者を支援するためのオンラインコンパイラの開発. 人工知能学会 第 82 回 先進的学習科学と工学研究会 (SIG-ALST), Vol. B5, No. 03, pp. 63–66, 2017.

[9] 又吉康綱, 中村聡史. typing.run: 初学者のプログラミング学習を支援するプログラムタイピングシステムの提案と実践. 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol. 2020-HCI-189, pp. 1–8, 2020.