情報処理学会論文誌　教育とコンピュータ　Vol.3　No.1　53–63 (Feb. 2017)

論　文

タブレット端末で動作する，マイコン用ビジュアルプログラミング環境aiBlocksの開発

光永 法明[1,a)]　井芹 威晴[1,†1]　吉田 図夢[1,†2]

受付日 2015年11月4日，再受付日 2016年9月28日，
採録日 2016年11月3日

概要：タブレット端末が普及するにともない，パソコンではなくタブレット上でのプログラミング環境も増えつつある．最近ではマイクロコントローラあるいはマイクロコンピュータを省略してマイコンとよばれる１チップから数チップで計算機を構築できるコンピュータが，家電，車といった製品の中だけでなく，趣味の電子工作や学校でのプログラミングの学習にも使われている．しかし，そのプログラミングに使えるタブレット端末上のビジュアルプログラミング環境はみあたらない．もし，そのような環境があればマイコンや電子工作に親しむ層が増えると期待される．ところで，マイコンにインタプリタを載せればプログラミング環境にコンパイラは不要であり，デバッグ情報をインタプリタから提供すれば，タブレット端末のような資源の限られたコンピュータでもプログラムの動作状態やマイコンの入出力といった情報を表示するプログラミング環境を実現しやすい．そこで，マイコンにインタプリタを搭載し，そのプログラムを開発するタブレット端末用ビジュアルプログラミング環境 aiBlocks を実現したので，初めてプログラミングと電子工作に取り組んだ初心者による aiBlocks の評価とともに報告する．

キーワード：タブレット端末，ビジュアルプログラミング，マイコン，プログラミング環境

aiBlocks: A Visual Programming Environment on a Tablet PC to Write a Micro Controller's Program

Noriaki Mitsunaga[1,a)]　Takeharu Iseri[1,†1]　Tom Yoshida[1,†2]

Received: November 4, 2015, Revised: September 28, 2016,
Accepted: November 3, 2016

Abstract: In recent years, tablet PCs are becoming popular and number of visual programming environments on them are increasing. Meanwhile, micro controllers, small computers that consist of single or a few ICs, are becoming popular among hobyists and programming education. We beleive that a visual programming environment to build a micro controller program on a tablet PC makes more pepole to familar with micro controllers and electronical circuits. However, no such visual programming environment on tablet PC has been published as far as we know. Then, we have developed such an environment named "aiBlocks" on Android OS. In this paper, we report how we developed aiBlocks and how beginners used it.

Keywords: tablet PC, visual programming, micro controller, programming environment

1. はじめに

最近ではスマートフォンやタブレット端末とよばれる機器のほぼ全面がディスプレイとタッチパネルで構成される機器が，従来からあるパソコン以上に普及してきている．それらで動作するアプリケーションソフトは映像や文章などのコンテンツを表示するものが多かったが，普及するにつれ techBASIC [1]，Hopscotch [2]，Pyonkee [3]，ScratchJr [4] といったプログラミング環境も増えている．こういった端末でプログラミング環境を提供することで，あらたにプログラミング用のハードウェア（たとえばパソ

[1] 大阪教育大学
Osaka Kyoiku University, Kashiwara, Osaka 582–8582, Japan
[†1] 現在，岸和田市立桜台中学校
Presently with Kishiwada Sakura-dai Junior High School
[†2] 現在，Fubic Singapore PTE. LTD.
Presently with Fubic Singapore PTE. LTD.
[a)] mitunaga@cc.osaka-kyoiku.ac.jp

コン）を用意せずに入門できるという利点がある．また，ビジュアルプログラミング環境を使うとキーボード操作に不慣れな児童・生徒もプログラミングを楽しむことができる [5] といわれており，ディスプレイと一体型のタッチパネルによる操作も直感的で相性がいいと考えられる．

ところでマイコン（マイクロコントローラ，マイクロコンピュータの略称，1 チップから数チップで構成され家電や車などに組み込まれることが多いコンピュータ）が広く普及し，産業機器や電化製品だけでなく趣味の電子工作や学校での学習にも広く部品として使われている．マイコンを使った電子回路の場合には，ハードウェア，ソフトウェアの両方についての知識が必要になる．そのため，ハードウェアのみの電子工作，ソフトウェアのみで完結するプログラミングと比べて学習のハードルが高くなりやすい．そこで，ある程度ハードウェアを完成させたマイコンボードやモジュールを用意したり，ライブラリを充実させたプログラミング環境を用意したりするといった工夫がされている．そのような環境はマイコンの機能と速度を最大限に生かす方向のものと，多少の制限があっても開発のステップを少なくする方向のものがある．初心者にとっては後者が有効であり，BASIC 言語処理系 [6], [7], [8], [9] や Arduino [10] などが開発・市販されている．とくに利用するマイコンを限定し周辺回路の準備のステップを少なくした Arduino は利用者が多い．

そのような環境を用いても多くの場合に，回路やプログラムが意図どおりに動作していないときにはデバッガや回路計（テスタ），オシロスコープといった従来からの動作状態の可視化ツールが必要とされている．しかし，初心者はそれらのツールの利用方法に不慣れなためツールが準備されていても操作の習得に時間がかかり，また慣れない接続でミスをすると余計に混乱するといった問題がある．それに対し，マイコン上にプログラミング言語を解釈実行するインタプリタを載せ，インタプリタから動作の情報を開発ツール上に送り，意識しなくてもプログラムの実行場所を表示し追加の配線なしにピンの様子が観察できる（可視化される）開発環境 iArduinoTerminal が提案されている [11] が，テキスト形式でのプログラミング環境である．

そこで，初心者にも多く使われている Arduino マイコンボード用の，Android を載せたタブレット端末で動作するビジュアルプログラミング環境 aiBlocks [*1] を開発し，開発の経過を報告してきた [12], [13], [14]．aiBlocks の実現には Arduino に iArduino インタプリタを載せ，タブレット端末ではビジュアルプログラミング言語から iArduino インタプリタが解釈・実行できる言語に変換し，実行中の入出力の様子を特別な操作なしに把握できる環境とする．本論文では，これまでの aiBlocks の開発 [12], [13]，aiBlocks を用いたプログラムの作例集試作 [14] の報告を 1 つの報告にまとめるとともに，他のビジュアルプログラミング環境との比較，開発にあたって想定している初心者のプログラミング過程の説明を加え，aiBlocks 開発環境全体として報告する．以下，2 章ではタブレット端末上でのプログラミング環境を概観し，aiBlocks の開発の基とした iArduinoTerminal for Android を紹介する．そして，想定する初心者のマイコンプログラムの作成過程を述べる．3 章では aiBlocks の全体構成と操作方法，実装，作例集について説明する．そして，4 章で aiBlocks の使用感と初心者による評価を報告し，5 章にまとめと今後の課題を述べる．

[*1] http://www.osaka-kyoiku.ac.jp/m̃itunaga/aiBlocks/ で公開している．

2. タブレット端末向けプログラミング環境

2.1 タブレット端末向けプログラミング環境の現状

現在入手しやすいタブレット端末の OS には，Android，iOS，Windows がある．Windows はキーボードのあるパソコンだけでなくタブレット型のパソコンでも利用される OS である．そのため，Windows 上のソフトウェアはタブレット形式のパソコンでも動作するがマウス操作を中心に考えられているものが多く，必ずしも指で画面を触っての操作はやりやすくない．またタブレット端末での Windows の OS の占有率は 3.7%と低い（2013 年時）[21] ため，以下では Windows をパソコン（PC）用 OS として扱う．

表 1 に著者らが調べた 1) 初心者にも扱いやすいと考えられるマイコン用ビジュアルプログラミング環境，2) タブレット端末上でのビジュアルプログラミング環境，3) タブレット端末上でのマイコン用プログラミング環境の一覧を示す．1) 初心者に扱いやすいと考えられるマイコン用プログラミング環境には ArduBlock [15]，PICAXE [16]（ターゲットの PICAXE は Microchip Technology Inc. の PIC マイコンに専用ファームウェアを書き込んだマイコンである），12Blocks [17]（Parallax Propeller は Parallax Inc. のマイコンの名称）などがあげられるが，これらは PC 用の OS（Windows，Mac OS X，Linux のすべて，もしくはいずれか）でのみ動作する．LEGO mindstorms EV3 のアプリケーションソフトウェア [18] はタブレット端末の OS である iOS 上で動作するが専用ハードウェアを必要とする．BlocklyDuino [19] はブロックを並べてプログラムを作成できるが，プログラムを Arduino 言語に変換する機能のみを持ち，コンパイラもしくは開発環境（Arduino IDE（Arduino マイコンボード用の標準的な開発環境），ArduinoDroid [20] など）が別途必要である．日本国内の中学校向けの学習教材などのマイコン用ビジュアルプログラミング環境は PC 上の OS でのみ動作し，自社のハードウェア向けとしてのみ供給されているのであげていない．

2) タブレット端末上でのビジュアルプログラミング環境には Hopscotch [2]，Pyonkee [3]，ScratchJr [4] などがあるが，マイコンのプログラムを作成する環境ではない．3)

表 1　初心者に扱いやすいと考えられるビジュアルプログラミング環境と，タブレット上で実現されているマイコン用プログラミング環境
Table 1　Visual programming environments for beginners including environments which run on tablet PC and targetting for micro controllers.

名称	動作 OS	言語形式	ターゲット	動作の可視化	備考
ArduBlock [15]	PC	ビジュアル	Arduino	printf デバッグ	
PICAXE [16]	PC	ビジュアル	PICAXE	シミュレータ上で可	
12Blocks [17]	PC	ビジュアル	Parallax Propeller, PICAXE, Arduino ほか	あり	ターゲットにより可視化に制限あり
LEGO mindstorms EV3 [18]	iOS	ビジュアル	LEGO mindstorms EV3	あり	専用ハードウェア
BlocklyDuino [19]	Android	ビジュアル	Arduino	-	ブロックから Arduino 言語への変換のみ
Hopscotch [2]	iOS	ビジュアル	タブレット上	あり	
Pyonkee [3]	iOS	ビジュアル	タブレット上	あり	
ScratchJr [4]	iOS	ビジュアル	タブレット上	あり	
ArduinoDroid [20]	Android	テキスト	Arduino	printf デバッグ	
iArduinoTerminal for Android [11]	Android	テキスト	Arduino	あり	

タブレット端末上でのマイコン用プログラミング環境である ArduinoDroid [20] と iArduinoTerminal for Android [11] はテキストベースのプログラミング言語用の環境である．ArduinoDroid はプログラムやハードウェアの動作を確認するために，いわゆる printf デバッグが必要であるのに対し，iArduinoTerminal for Android は変数の値表示や入出力のグラフ表示，プログラムなしでの出力の操作といった機能を持つ．

2.2　インタプリタ iArduino と開発環境

iArduino は Arduino 上で動作するインタプリタである [22]．Arduino の標準の Arduino 言語に文法などが似たプログラミング言語 iArduino 言語を解釈して実行する．iArduino は非同期シリアルインタフェースを通じて対話的な実行と実行中のプログラムの制御を受け付け，パソコン上のシリアルターミナルなどを通して操作する．通常のプログラムの実行に加え，ステップ実行，自動ステップ実行（文の解釈と解釈の間に一定時間実行停止）ができる．そして，iArduino 言語のプログラムを不揮発メモリ（マイコン内蔵の EEPROM）に保存し，リセット時に自動実行ができる．したがって開発が終われば Arduino 単体で動作する．また可読形式に加えバイナリ形式のデバッグ用インタフェースを持っており，入出力ピンの読み書きや変数の値の読み出しがプログラムの実行中でも可能である．加えて実行中のプログラムの位置（アドレスに相当）を知ることができる．

iArduinoTerminal は Windows 用, iArduinoTerminal for Android は Android 用の iArduino 用シリアルターミナルソフトである [11]．シリアルターミナルに加えて，プログラムエディタ，変数と入出力波形のモニタ，入出力ピンの操作インタフェースを備えているため開発環境と考えられる．プログラムエディタはプログラムの実行中の部分をハイライト表示する．これにより，プログラムの実行の様子，入出力ピンの様子を難しい操作なしに観察できる．またデバッグ用インタフェース部分のみを Arduino 言語のライブラリとして利用できるので，iArduino 言語で開発できない速度や規模のプログラムについて Arduino 言語に移行しても入出力波形のモニタとピンの操作ができる．そこで aiBlocks は iArduinoTerminal for Android を拡張してビジュアルプログラミング環境を実現する．

aiBlocks は Android 用アプリ開発の標準プログラミング言語である Java で記述する．aiBlocks を JavaScript で記述しブラウザ上のウェブアプリケーションとして実現できれば実行環境がより多くなるメリットがあるが，ウェブアプリケーションではシリアルポートを操作できないためである．著者らが調べた限り，シリアルポートを操作できる JavaScript の実行環境は Chrome アプリと node.js である．Chrome アプリは Chrome OS 上または PC 用 OS の Chrome ブラウザ上で実行できるが，Andoroid の Chrome ブラウザでは実行できない．また node.js はサーバサイドの JavaScript 実行環境である．PC の場合にはシリアルポートサーバとよばれるシリアルポートの通信をネットワークの通信に変換するプログラムを動作させ，ブラウザ上の JavaScript がそのプログラムと通信することも考えられる．しかし，Android OS の場合にはそのようなプログラムを用意することが難しい．

2.3　ビジュアルプログラミング環境の設計方針と想定する初心者のマイコンプログラムに親しむ過程

ここではブロックを並べて手続き型のプログラムを作成するビジュアルプログラミング言語を想定する．そのような言語の場合にはあらかじめ用意するプログラムのブロッ

クの種類が多くなることが想定される．そのため，利用可能なブロックをすべて画面に表示するのではなく，いくつかのカテゴリに分類しメニューを用意することになると考えられる．

Scratch [23] は子ども向けのプログラミング環境として有名であり，多くの子どもたちが利用している（図 1）．Scratch はプログラミング言語を教えプログラムを作らせることを目的としておらず，子ども自身が興味を持って何かを達成しようとすることを実現するためのツールを目標としている．したがって Scratch では子どものやりたいことでブロックが分類，メニューが作成されている．図 1 にあるように，たとえば，キャラクタを動かしたいときには「動き」を，キャラクタの見た目を変えたい（画像を切り替えたい）ときには「見た目」を，音を出したいときには「音」を，マウスによるクリックなどのイベントに反応させたいときには「イベント」を，繰返しや条件分岐などプログラムの流れを制御したいときには「制御」をクリックすれば，その分類のブロックが表示される．またブロックの色はメニューの色と統一されていて同じ分類のブロックのメニューを探しやすくなっている．

一方，マイコン用のプログラミング環境である ArduBlock を見るとブロックは「制御」「ピン」「比べる」「計算する」「変数/定数」「ユーティリティー」「通信」などに分類されている．この分類はプログラミング言語の構造を知っていると分かりやすい．たとえば「制御」には for 文，while 文，if 文と時間待ちの，「ピン」には入出力ピン操作の，「比べる」には比較演算子と論理演算子に相当するブロックが分類されている．LED を点灯するには「ピン」に，音を出すには「ユーティリティー」にあるブロックを使う．どちらかというと，マイコンにできることを理解してから必要なブロックを探す必要がある．

ところで，マイコンと簡単な部品を利用した作品で何をしたいかという問いを子どもにすると，最初の答えは光らせたい，音を出したい，動かしたいといった答えになると考える．それができてから具体的に，点滅させたい，音を変化させたい，動きを変えたいとなり，さらには外界に反応させたいとなるのではないだろうか．LED の点滅，音の出力，モータの動きといったものがあると回路・マイコンの動作が分かりやすい．また作りたい作品のイメージを具体化するにも役立つと考える．自身の経験も含め，初心者のマイコンプログラム（作品）の作成と理解の過程は次のようになると考える．

(1) 作品づくりの最初の段階では，出力に関係するブロックの使い方を覚える．
(2) 次に，そういったブロックと時間待ちのブロックを並べ，オープンループで動作するプログラムを作る．
(3) 繰返しや，条件分岐，あるいは演算を使って動作を複雑にする．
(4) 入力に応じて動作を分岐するプログラムにする．
(5) 入力や記憶（変数）を使って計算（演算）をし，動作を変えるプログラムにする．
(6) デバッグ方法も覚え，さらに発展させていく．

そうであれば ArduBlock のようなプログラミング言語の構造に基づくブロックの分類よりも，Scratch のような分類がよいと考えられる．以下では，この想定に基づいて操作などが煩雑にならないようにプログラミング環境を設計，実現していく．

図 1　Scratch 2.0 の表示（一部切り抜き）

Fig. 1　Outlook of visual programming environment Scratch 2.0.

3. ビジュアルプログラミング環境 aiBlocks

3.1 全体構成

aiBlocks の全体構成を図 2 に示す．プログラムの開発対象である Arduino 上に iArduino インタプリタを載せ

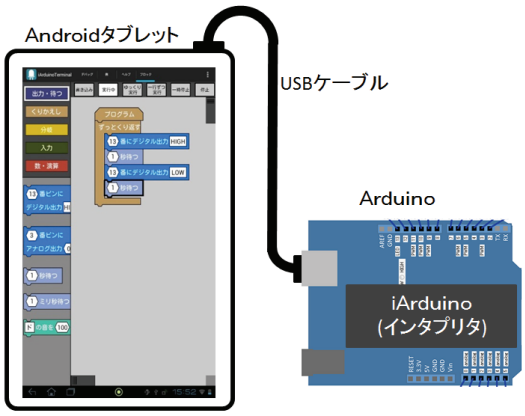


図 2　aiBlocks の動作するタブレットと Arduino は USB ケーブルでつなぐ．Android 上の aiBlocks アプリケーションで作成したプログラムを，Arduino マイコンボード上の iArduino インタプリタが実行する

Fig. 2　The aiBlocks runs on a tablet PC. The target Arduino, micro controller, which runs iArduino interpreter is connected with the tablet PC via USB cable. The program written in aiBlocks runs on the Arduino being interpreted by iArduino.

る．aiBlocks は Android 上で動作し，ブロックを並べて作成したプログラムを iArduino 言語に変換し，iArduino にシリアルインタフェースを通して送る．iArduino は受け取ったプログラムを RAM に記憶し，逐次解釈しながら実行する．プログラムの開発が終わり RAM 上のプログラムを EEPROM に書き込めば Arduino だけで動作する．Arduino 上には USB シリアルインタフェースが載っており，簡単な回路であればバスパワーで動作するのでタブレット端末と USB ケーブルでつなぐだけでよい．

3.2　画面構成と基本操作

aiBlocks の画面表示を**図 3** に示す．画面左上にブロックのカテゴリを選択する①『カテゴリ選択ボタン』が並び，その下には②『ブロックパレット』がある．画面中央には，ブロックを並べてプログラムを作る③『プログラムエリア』があり，その上部には実行に関するボタンが並ぶ④『実行ツールバー』がある．

プログラムを作成するときはまず，使いたいブロックの①カテゴリをタップする．すると選択したカテゴリの②ブロックパレットが表示される．その中から使いたいブロックを選び，③プログラムエリアにドラッグし並べる．図 3 では「出力・待つ」のカテゴリの「デジタル出力」ブロックと「待つ」ブロックを並べて，1 秒間 Arduino 上の LED を点灯するプログラムを作っている．プログラムが完成したら④実行ツールバーの『書き込み』ボタンをタップする．するとブロックのプログラムが iArduino 言語に変換され iArduino に送られる．『実行』ボタンをタップすると，そのプログラムが実行される．

3.3　プログラミングに使うブロック

aiBlocks で用意しているブロックをカテゴリごとに**表 2** に示す．ブロックのカテゴリは 2.3 節での想定に基づき

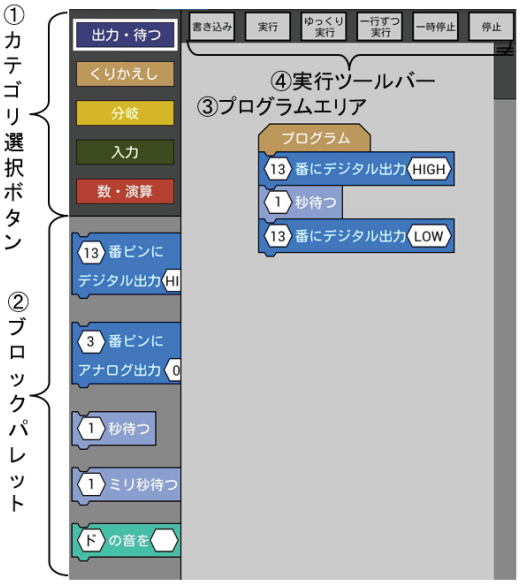


図 3　aiBlocks のプログラミングタブの表示．上部のタブバーを省略している

Fig. 3　Toolbars and buttons of aiBlocks.

表 2　aiBlocks に用意しているブロック一覧

Table 2　List of programming blocks on aiBlocks.

カテゴリ	ブロック名称（C 言語の相当する名称）	動作・処理など
出力・待つ	デジタル出力	指定したピンのデジタル出力（L/H）を変える
	アナログ出力	指定したピンにアナログ出力（256 段階の PWM 出力）を変える
	待つ（秒）	指定した時間（秒単位）だけ待つ
	待つ（ミリ秒）	指定した時間（ミリ秒単位）だけ待つ
	音	指定した周波数（Hz）の音を指定した時間（ミリ秒単位）鳴らす
くりかえし	無限ループ	ずっとブロック（群）を繰り返す
	for 文	指定した回数ブロック（群）を繰り返す
	while 文	条件式が成り立つ（真：値が 0 でない）間ブロック（群）を繰り返す
分岐	if 文	条件式が成り立つ（真：値が 0 でない）ときブロック（群）を実行する
	if-else 文	条件式が成り立つとき 1 つ目のブロック（群）を，成り立たないとき 2 つ目のブロック（群）をを実行する
	比較演算子	左右の値を比較し，等式（不等式）が成り立つとき 1 となる演算子
	AND	左右がともに真である場合 1 となる演算子
	OR	左右のいずれかが真である場合 1 となる演算子
	NOT	式が 0（偽）である場合 1 となる演算子
入力	デジタル入力	指定したピンのデジタル入力の値（0(L) か 1(H) を返す）
	アナログ入力	指定したピンのアナログ入力の値（0～1023）を返す
数・演算	定数・変数	定数（数値）または変数名（アルファベット a～m）を表す
	代入文	変数に式の値を代入する
	四則演算	四則演算の二項演算子

(a) 数値モード　(b) 変数モード

(a) numeric input mode　(b) variable input mode

図 4　数値/変数名をタップすると入力パッドを表示する．入力パッドは (a) 数値モードと (b) 変数名モードをタブで切り替えられる

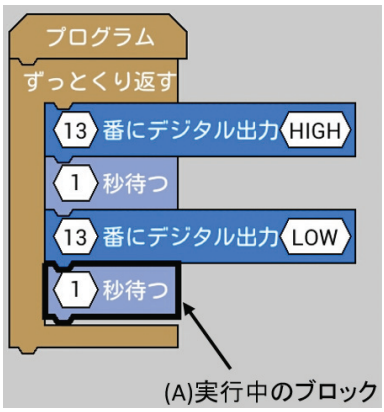
Fig. 4　An input pad is shown when you tap a number of a variable. The pad has two modes, (a) numeric input mode and (b) variable selection mode.

『出力・待つ』『くりかえし』『分岐』『入力』『数・演算』の 5 種類に分けている．ブロックは全部で 19 種類である．『出力・待つ』には，デジタル/アナログ出力と待つ（busy loop），音を出すブロックがある．『くりかえし』には，C 言語の for 文と while 文に相当するブロックが，『分岐』には，if 文に相当するブロックと条件式でよく利用する演算子に相当するブロックがある．『入力』には，デジタル/アナログ入力のブロックが，『数・演算』には，数値や変数，代入，四則演算のブロックがある．

ブロックの形状は，単独で C 言語の文（式文）に相当するブロック（デジタル出力や変数への代入など），式の一部になるブロック（数値，変数名，演算子，デジタル/アナログ入力など），制御文に相当するブロック（for，while，if 文など）で変え，形状が異なるブロックが結合できないようにしている．これにより文法的な誤りを防ぐ．

また，数値や変数名などの入力には専用の入力パッドを用意している．数値・変数名の入力パッドを図 4 に示す．上部のタブで (a) 数値と (b) 変数名の入力モードを切り替える．(a) 数値モードでは 8 桁までの数値を，(b) 変数名モードでは a から m の範囲のアルファベット 1 文字を入力できる．他にピン番号，アナログ入力の値，等号・不等号，音階，四則演算記号の入力用パッドがある．これにより正しい iArduino 言語として変換できない，あるいは実行してもおかしくなるような文字入力をできるだけ防ぐ．

aiBlocks で作ったプログラムの例を図 5 に示す．『無限ループ』のブロックの中に『デジタル出力』と『待つ（秒）』のブロックがある．このプログラムを iArduino 言語のプログラムに変換すると図 5 下のテキストになる．このプログラムはユーザに見せない．「書き込み」ボタンをタップするとこの変換をして iArduino に送る．これを実行すると Arduino 上の LED（13 番ピンに接続）が 1 秒ごとに点灯/消灯を繰り返す．




```
pinMode(13,OUTPUT);
for(;;){
 digitalWrite(13,HIGH);
 delay(1*1000);
 digitalWrite(13,LOW);
 delay(1*1000);
}
```

図 5　aiBlocks で作成したプログラム（上）を，iArduino 言語のプログラム（下）に変換し，インタプリタへ渡す．プログラムの実行中は，（上）の (A) が指す「1 秒待つ」のように実行中のブロックの枠を太く表示する

Fig. 5　A program written with blocks (upper) is converted to iArduino language (lower) and sent to iArduino interpreter. Running block is highligted as shown (A) when the micro controller runs a program.

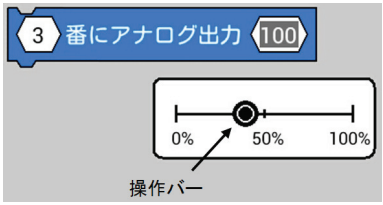


図 6　アナログ出力のブロックとその入力パッド．入力パッドの操作バーをスライドさせると，マイコンボードの出力がリアルタイムに変化する

Fig. 6　An analog output block and its input bar to determine output value. The output of micro controller changes in real time when you slide the bar.

また出力に関するブロックをタップするとすぐにピンの出力が変わる．たとえば図 5 のデジタル出力のブロックをタップすると 13 番ピンの出力がすぐに HIGH または LOW になる．またアナログ出力の値の入力パッドは操作バーをスライドさせて値（0〜255）を変える（図 6）．指を離さずバーにのせたままでも出力が変化するようにしている．入力のブロックは入力の値をリアルタイムに表示す

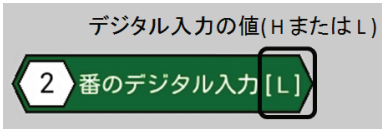


図 7　デジタル入力のブロック．現在の値（L または H）を表示する
Fig. 7　An digital input block. It shows current value L or H.

図 8　アナログ入力のブロック．現在の値（0〜1023）を表示する
Fig. 8　An analog input block. It shows current value (from 0 to 1023).

る．デジタル入力のブロックは電圧が高い場合は H，低い場合は L をブロックに表示する（図 7）．Arduino のアナログ入力は 0 [V]〜5 [V] の電圧を 1,024 段階で測ることができる．アナログ入力のブロックは現在の入力電圧を 0〜1023 の数値で表示する（図 8）．

3.4　プログラムの実行

プログラムの実行中は実行しているブロックをハイライト表示する．図 5 では (A) の 1 秒待つブロック（delay(1*1000) に変換）を実行中である．プログラムの動作を観察しやすいよう，通常の実行（図 3 の「実行」ボタン）に加え，1 命令実行すると停止するステップ実行（「1 行ずつ実行」ボタン）と，文の解釈と解釈の間に 0.5 秒間実行を停止する自動ステップ実行（「ゆっくり実行」ボタン）ができる．

3.5　ブロックの描画と操作の実現

aiBlocks の開発言語は Android 標準の Java である．ブロックの描画に適した標準のライブラリがないため，画面の図 3 の部分を Canvas とし，そこに多角形や円弧の描画によりボタンやブロックを実現している．そのため，プログラムエリアのスクロールも aiBlocks アプリ内に実装して実現している．Canvas 内をタップするとタップした座標によりブロック上，プログラムエリア内（ブロック上ではない），ボタン上，それ以外（何もしない）と判断する．ブロック上をタップしたと判断した後に指を動かすとドラッグと判断し，指の動きに合わせてブロックの座標を更新し，Canvas を再描画する．指が離れたら（ドラッグ終了），離れた場所の座標とすでにプログラムエリアにある各ブロックの座標を比較し，隣接ブロックを探す．ドラッグしたブロックが既存のブロックの下より（文に相当するブロックの場合）で座標のずれがブロックの大きさの 1/3 以内であれば隣接ブロックとし，既存のブロックとつなぐ．

ブロック上ではないプログラムエリア内をタップした場合にはスクロールをしたいと判断し，ブロック全体の描画原点を移動してスクロールの効果を出す．それ以外のカテゴリ選択ボタンや実行ツールバー上のボタンのタップであれば，それぞれの機能を実現する関数を呼び出す．

実行中のブロックのハイライト表示については次のようにして実現している．『書き込み』ボタンがタップされブロックによるプログラムを iArduino 言語に変換する際に，各ブロックの変換後のプログラムリスト上での位置（先頭からのバイト数と長さ）を記憶しておく．プログラムの実行中には iArduino から実行中のプログラムの位置が送られてくるので，対応するブロックを探しそれをハイライト表示（枠線を太く）する．

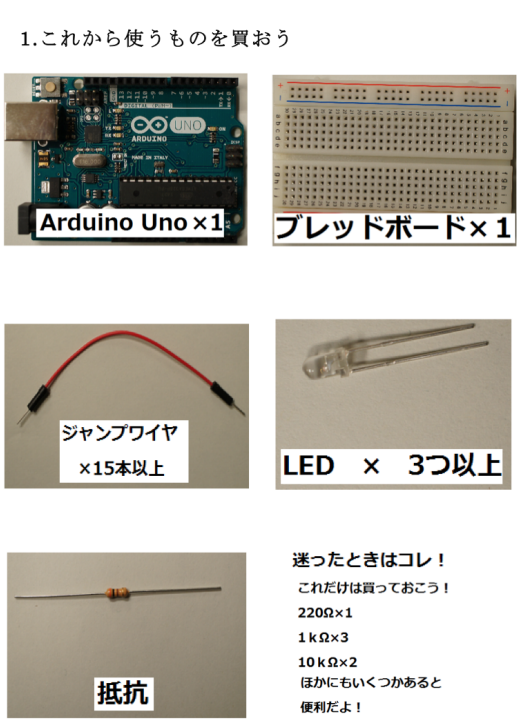


図 9　作例を真似るのに必要な部品を紹介するページ（部分）
Fig. 9　This page introduces electric parts which are used for examples.

3.6　aiBlocks を利用したマイコン電子工作の作例集

作例集ではまず，作例を真似るのに必要な部品について簡単な紹介（図 9）をし，ブレッドボードの使い方，カラーコードの読み方，aiBlocks の操作などを説明する（図 10）．次に Arduino 上の LED を使うプログラムを紹介した後で，ブレッドボード上での配線例と aiBlocks のプログラムリストを使って作例を紹介する（図 11）．見開き 2 ページを使い準備する部品，配線例，プログラムを紹介し，トラブルシューティングを載せる．そして部品やプログラムの使い方を理解するために試すとよい内容を「やってみよう」として載せる．

作例の一覧を図 12 に示す．2.3 節の想定を基に自身が

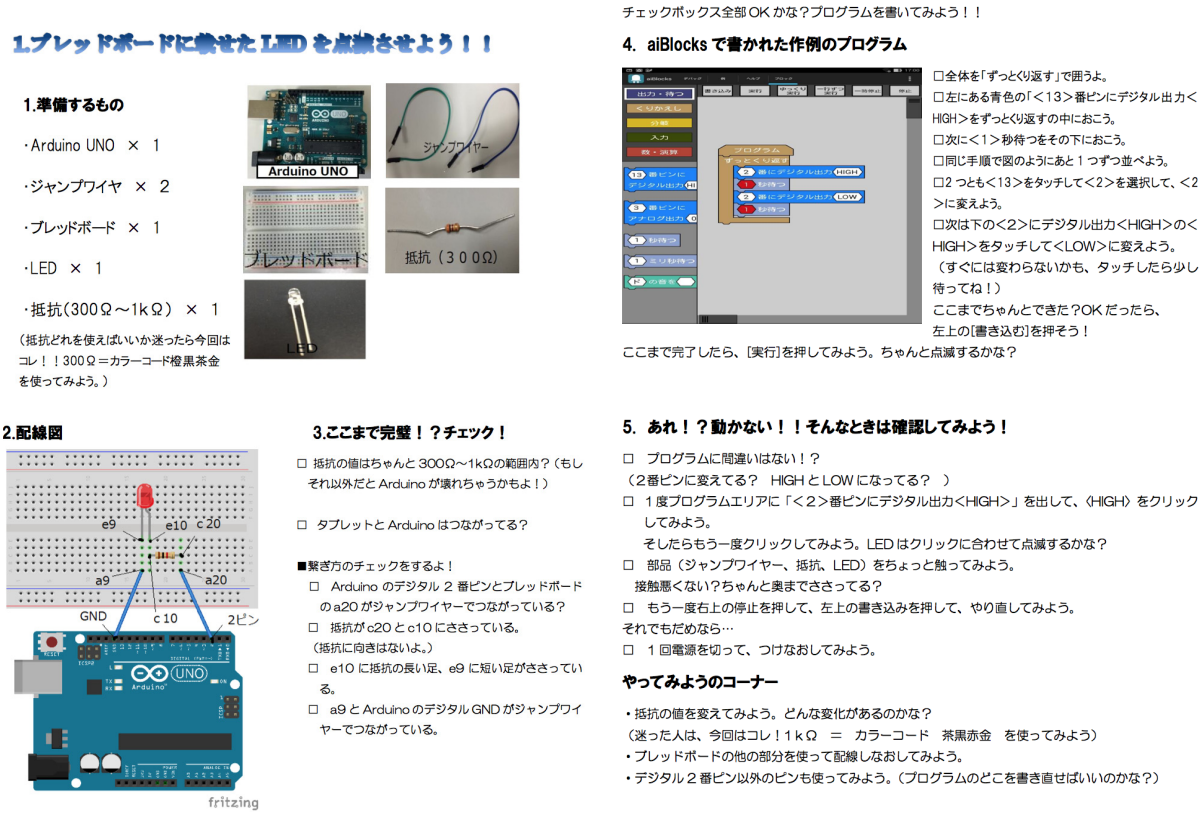


図 11　ブレッドボードに載せた LED を点滅させる作例のページ．見開き 2 ページで構成している

Fig. 11　These pages explain how to blink an LED on a solderless breadboard.

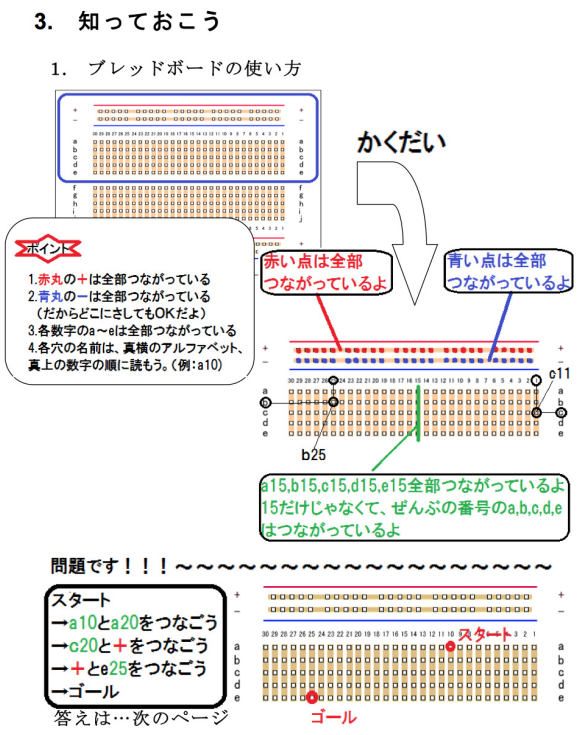


図 10　ブレッドボードの使い方を説明するページ

Fig. 10　This page explains how to use a solderless breadboard.

電子工作とプログラミングの初心者である著者の 1 人が構成を決めた．1 から 5 は出力と時間待ち，くりかえしのブロックを主に使った作例である．まず「1 LED を点滅させる」でブレッドボードと aiBlocks の操作に慣れる．「2 LED を 2 つ点滅させる」「3 圧電スピーカを使って音を出す」「4 模型用モータを回してみる」でデジタル出力，音，待ちブロックを使った，光，音，動きの作り方を知る．そして「5 LED の明るさを変えてみる」でアナログ出力ブロックを扱う．

次に入力に応じて動作を分岐させる作例が 6 と 7 である．「6 スイッチで LED を点滅させる」「7 スイッチを押し

1 LED を点滅させる
2 LED を 2 つ点滅させる
3 圧電スピーカーを使って音を出す
4 模型用モータを回してみる
5 LED の明るさを変えてみる
6 スイッチで LED を点滅させる
7 スイッチを押したらカッコウがなく
8 ボリュームの角度で LED の点滅の速さを変える
9 暗くなったら LED を点滅させる
10 温度センサで温度が高くなると LED を点灯させる
11 熱くなったら扇風機を回す
12 重さを感じたら音を出す
13 人感センサで人が近づいたら LED を点滅させる
14 距離センサで何かが近づいたら LED を点滅させる
15 LED をだんだん明るく，だんだん暗くする
16 LED を順番に点灯する
17 明るさの変化に反応させる

図 12　作例集で取り扱う作例

Fig. 12　List of examples.

たらカッコウがなく」ではデジタル入力と if ブロックの使い方を扱う．そして入力を演算する作例「8 ボリュームの角度で LED の点滅の速さを変える」があり，アナログ入力の使い方を紹介する．そして「9 暗くなったら LED を点滅させる」から「14 距離センサで何かが近づいたら LED を点滅させる」はセンサの紹介を兼ねて条件分岐を復習している．さいごに「15 LED をだんだん明るく，だんだん暗くする」「16 LED を順番に点灯する」「17 明るさの変化に反応させる」で少し抽象度の高い話題を取り扱う．

これらの作例の紹介の後で，作例の応用を 3 つ紹介している．1 つは「3 圧電スピーカを使って音を出す」「9 暗くなったら LED を点滅させる」を組み合わせた，照明を消すとハッピーバースデーのメロディが流れる作品である．次に「3 圧電スピーカを使って音を出す」「10 温度センサで温度が高くなると LED を点灯させる」「17 明るさの変化に反応させる」を組み合わせた，冷蔵庫のドアが開放されているとメロディで知らせる作品である．最後に「4 模型用モータを回してみる」「7 スイッチを押したらカッコウが鳴く」を組み合わせた，スイッチで動く模型の作品を紹介している．ほかにトラブルシューティングの方法を紹介するページを設けている．

4.　aiBlocks の使用感と初心者による評価

4.1　aiBlocks を使った製作例と使用感

aiBlocks を使って製作したライントレースカーの外観を図 **13** に，回路図を図 **14** に示す．マイコンボードは ArduinoUno，電源には単三形乾電池 4 本を使用している．モータの駆動にはトランジスタを使用し，ベース端子をアナログ出力ができる 10 番と 11 番ピンに接続する．ラインの判別には LED と CdS セルを使用する．LED と CdS を，ライン（黒色）の幅以上のスペースを空けて 2 組固定し，LED で床面を照らし，CdS でその反射光を受ける．固定抵抗と CdS の抵抗分圧により，CdS の明るさの変化による抵抗値の変化を電圧に変える．CdS で受ける光が弱い（ライン上）とアナログ入力の値が大きくなり，逆に受ける光が強い（ライン外）とアナログ入力の値が小さくなる．

車体の中心がライン上にあるときは左右の CdS の間にラインがあるので，アナログ入力の値は左右ともに小さい．このときは左右のモータの回転数を同一にして直進すればよい．ラインが左に曲がっていたり車体の中心がラインの右寄りになり，左の CdS がラインの黒色の影響を受けるとアナログ入力の値が大きくなる．このとき左のモータの回転数を下げれば，車体の中心がライン上に戻る．右の CdS がラインの黒色の影響を受けた場合も同様である．

ライントレースカーのプログラムを図 **15** に示す．車体の左にある CdS を A0 番ピンに，モータを 10 番ピンにつないでいる．1 つ目の if-else 文で CdS が黒色の影響を受け A0 番ピンの値が 250 以上であれば左のモータの回転速度

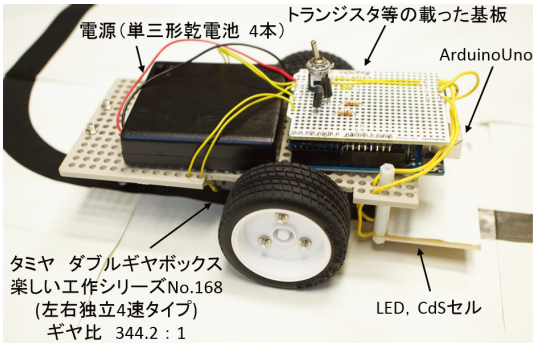


図 **13**　作成したライントレースカーの外観
Fig. 13　Outlook of a simple line follower robot.

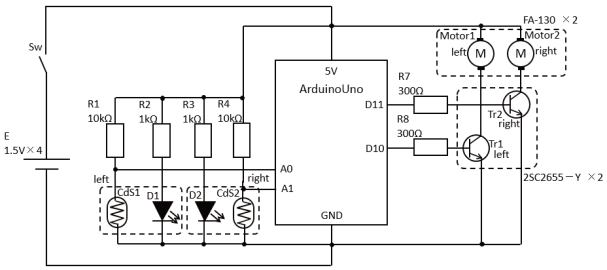


図 **14**　作成したライントレースカーの回路図
Fig. 14　Circuit diagram of a line follower robot.

図 **15**　aiBlocks で書いたライントレースカーのプログラム
Fig. 15　Program written in aiBlocks for a line follower robot.

を決める 10 番ピンのアナログ出力を 15 とし，そうでなければ 50 としている．2 つ目の if-else 文は車体の右にある CdS とモータ用である．この 2 つの if-else 文を繰り返すことでライン上を左右に振れながら車体が進む．

ライントレースカーは著者の 1 人が製作したが，その過程で以下の使用感を得た．使用した Android 端末は Acer 製の ICONIA TAB A200（Andriod OS 4.0.3，画面サイズ 10.1 インチ，解像度 1,280×800，静電容量式タッチパネル）である．こういった小さな模型を初心者が製作する場合には，if-else 文の条件式の閾値をラインとライン以外の反射率などから計算によって決めるのではなく，実際にラインとライン以外でのアナログ入力の値を測って決めることが

図 16　条件式の入っている if-else 文ブロック
Fig. 16　An if-else block which has a conditional expression block.

想定される．そのような方法で閾値を決めるにあたり，アナログ入力の値をブロックに表示していることが有効と感じた．また，モータの回転数も試行錯誤的に決めると想定される．それについてもアナログ出力のブロックをタップして 2 つの速度の切り替え，入力パッドの操作バー（図 6）による実際のモータの回転数の調整が有効と感じた．また Android 端末は軽くバッテリの残量を気にせず，床に置いたコース上のライントレースカーのそばで利用できた．

一方で，ブロックの移動操作が直感的でないと感じる場面があった．具体的には，if 文などのブロックを移動させようとタップしてドラッグしたつもりが，条件式のブロックをタップしてドラッグしてしまうことが何度もあった．**図 16** の if-else 文のブロックを移動させようとしたときには，条件式の部分ではなく「もし」「なら」といった文字の部分をタップしてドラッグする必要がある．しかし，条件式を含んだブロック全体の中央，つまり条件式の中央をタップする傾向が強かった．同じ著者が Scratch や ArduBlock をマウスで操作した場合には，そのような傾向がなかったことから，指によるタッチパネル操作固有の問題の可能性がある．

4.2　初心者による評価

aiBlocks と作例集の評価のため，プログラミング未経験の教育学部所属の大学生 5 名の利用の様子を観察しインタビューした．使用した Android 端末は ASUS 製の ASUS Pad TF103C（Android 4.4.2，画面サイズ 10.1 インチ，解像度 1,280×800，静電容量式タッチパネル）であり，作例集のほかに制作に必要な部品やブレッドボードなどを用意し，インタビュー場所へ運んだ．観察にあたっては次のように利用者に指示した．まず 1〜4 の作例を作成する．これによりブレッドボードを使った配線に慣れ，出力，時間待ち，繰返しのブロックの意味を理解することを期待する．次に 12，15 以外の好きなプログラムを選んでもらいテキストのページだけを見て作成する．そして，配線の最も複雑な「12 重さを感じたら音を出す」とプログラムの最も複雑な「15 LED の明るさをだんだん明るく，だんだん暗くする」を作成する．それ以上の細かな指示はせず，プログラムの改変などは制限しなかった．利用は 1 人ずつとし，1 人の利用時間はおおよそ 1 時間 30 分である．

利用にあたって aiBlocks と作例集に大きな問題は見られなかった．Android 端末は同程度の価格の PC よりも軽く動作時間が長い（TF103C の場合，キーボードなしで約 550 g，カタログ上の使用時間は約 9.5 時間）ので，作例集，部品一式とともに持ち運び，電源がないインタビュー場所で利用するのに不便はなかった．また大学生は PC に不慣れであってもスマートフォンの操作には慣れているため，操作上の問題もなかった．その一方で，Arduino ボード上の LED に気づきにくい，ブレッドボードの穴の位置の間違いに気づきにくいという問題がみられた．しかし，作例集を利用し配線図の読み方に慣れてくると，配線ミス（ブレッドボードの穴の位置の間違い）には自然と気づくような様子が観察された．

またプログラムの作成が楽しくなり aiBlocks で作成できる以上の長さのプログラムに挑戦した学生がいた．インタビューでは「慣れてきたら楽しかった」，「仕組みは分からないけど，動いたのはうれしかった」という感想が得られた．したがって aiBlocks と作例集は初めてマイコンのプログラムの作成する初心者が楽しめるものであると考える．

5.　おわりに

本論文では，インタプリタを載せたマイコンのプログラムを作成する，タブレット端末用ビジュアルプログラミング環境 aiBlocks とその作例集の開発について報告した．開発にあたっては初心者がマイコンのプログラムに親しむ過程をあらかじめ想定し，その過程をスムーズに経るように配慮した．そして aiBlocks を利用した製作例としてライントレースカーを紹介し著者による使用感を述べた．また初心者に aiBlocks をインストールしたタブレット，作例集と必要な部品を渡し評価を求めた．その結果，aiBlocks と作例集を利用することでプログラミングを楽しめることが分かった．今後はサンプルプログラムを充実させるとともに，多くの利用者を得たい．

謝辞　本研究は JSPS 科研費 25870418 の助成を受けたものである．

参考文献

[1] Byte works Inc.: techBASIC, available from ⟨http://www.byteworks.us/Byte_Works/techBASIC.html⟩ (accessed 2016-09-27).
[2] Hopscotch Technologies: HOPSCOTCH Make your own games, available from ⟨https://www.gethopscotch.com/⟩ (accessed 2016-09-27).
[3] 合同会社ソフトウメヤ：Pyonkee, available from ⟨http://www.softumeya.com/pyonkee/ja/⟩ (accessed 2016-09-27).
[4] ScratchJr project: ScratchJr, available from ⟨http://www.scratchjr.org/⟩ (accessed 2016-09-27).
[5] 兼宗　進，阿部和広，原田康徳：プログラミングが好きになる言語環境，情報処理，Vol.50, No.10, pp.986–995 (2009).

[6] Parallax Inc.: BASIC Stamp Windows Editor, available from ⟨https://www.parallax.com/downloads/basic-stamp-editor-software-windows⟩ (accessed 2016-09-27).
[7] MCS Electronics: BACOM-AVR, available from ⟨http://mcselec.com/index.php?option=com_content&task=view&id=14&Itemid=41⟩ (accessed 2016-09-27).
[8] mikroElektronika: mikroBASIC Pro for PIC, available from ⟨http://www.mikroe.com/mikrobasic/pic/⟩ (accessed 2016-09-27).
[9] MicroEngineering Labs.: PICBASIC PRO, available from ⟨http://www.picbasic.co.uk/forum/⟩ (accessed 2016-09-27).
[10] Arduino project: Arduino, available from ⟨http://arduino.cc/⟩ (accessed 2016-09-27).
[11] 光永法明：タブレット端末で動作する，インタプリタ型言語搭載マイコンのプログラミング環境の開発，情報処理学会研究報告，Vol.2013-CE-119(8), pp.1–4 (2013).
[12] 井芹威晴，光永法明：タブレット端末で動く，マイコン用ビジュアルプログラミング環境 aiBlocks の開発，日本産業技術教育学会近畿支部第 31 回研究発表会講演論文集，pp.29–30 (2014).
[13] 井芹威晴，光永法明：aiBlocks：マイコンにインタプリタを載せて利用するタブレット端末用ビジュアルプログラミング環境，情報処理学会研究報告，Vol.2015-CE-128(8) (2015).
[14] 光永法明, 吉田図夢, 井芹威晴：タブレット端末で動作する Arduino 用プログラミング環境 aiBlocks の初心者向け作例集の試作と評価，第 42 回人工知能学会 AI チャレンジ研究会人工知能学会研究会資料，Vol.SIG-Challenge-042-04, pp.16–20 (2015).
[15] ArduBlock project: ArduBlock, available from ⟨http://blog.ardublock.com/⟩ (accessed 2016-09-27).
[16] Revolution Education Ltd.: PICAXE Editor 6, available from ⟨http://www.picaxe.com/Software/PICAXE/PICAXE-Editor-6/⟩ (accessed 2016-09-27).
[17] OneRobot: 12Blocks, available from ⟨http://onerobot.org/products/12blocks/⟩ (accessed 2016-09-27).
[18] The LEGO Group: EV3 プログラミングアプリ，入手先 ⟨http://education.lego.com/ja-jp/learn/middle-school/mindstorms-ev3/teaching-resources/software/tablet-app⟩ （参照 2016-09-27）.
[19] Lin, F.: BlocklyDuino is a web-based visual programming editor for arduino, available from ⟨https://github.com/BlocklyDuino/BlocklyDuino⟩ (accessed 2016-09-27).
[20] Smirnov, A.: ArduinoDroid - Arduino IDE for Android, available from ⟨http://www.arduinodroid.info/⟩ (accessed 2016-09-27).
[21] 総務省：平成 25 年版情報通信白書第 1 部特集「スマート ICT」の戦略的活用でいかに日本に元気と成長をもたらすか，入手先 ⟨http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h25/pdf/⟩ （参照 2016-09-27）.
[22] Mitsunaga, N.: An interpreted language with debugging interface for a micro controller, *IEEE 1st Global Conference on Consumer Electronics*, pp.115–119 (2012).
[23] Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmod, E.: The Scratch programming language and environment, *Trans. Comput. Educ.*, Vol.10, No.4, pp.16:1–16:15 (2010).

光永 法明　（正会員）

1997 年大阪大学大学院工学研究科電子制御機械工学専攻博士前期課程修了．同年同研究科知能・機能創成工学専攻博士後期課程進学．2002 年同単位取得済み退学．同年同大学科学技術振興特任教員．2003 年博士後期課程修了．同年大阪大学大学院工学研究科助手．2004 年より ATR 知能ロボティクス研究所研究員．2008 年より金沢工業大学機械系ロボティクス学科講師．2011 年より大阪教育大学教員養成課程技術教育講座准教授．日本ロボット学会，人工知能学会会，日本産業技術教育学会各会員．

井芹 威晴　（正会員）

2013 年大阪教育大学学校教育教員養成課程技術教育専攻卒業．同年同大学大学院教育学研究科技術教育専攻修士課程進学．2015 年同修了．同年大阪府岸和田市立桜台中学校教諭．現在に至る．

吉田 図夢

2015 年大阪教育大学学校教育教員養成課程技術教育専攻卒業．同年株式会社フュービック入社．フィジカルフィットネス Div Dr. ストレッチイオンモール京都店配属．2016 年 Dr. ストレッチ西宮ガーデンズ店へ異動，店長へ昇格．同年所属を Fubic Singapore PTE. LTD. へ異動，Suntec city mall shop 配属．現在に至る．