Vol.2021-CE-159 No.1
2021/3/13

教育向けマイコンボード micro:bit のボタン押下メソッドの検証とプログラミング教育に生起しうる問題の検討

矢田陽子[†1]　岡田繁[†1]　中西祥彦[†2]　江見圭司[†2]

概要：教育向けマイコンボード mico:bit のボタン押下メソッドの検証とプログラミング教育に生起しうる問題を組込み 技術から検討した。ボタン押下メソッドには was_pressed, is_pressed, get_presses があり,それの動作状況を詳細に検討した。特にじゃんけんプログラムで使用するときにボタンを押すタイミングによって起きる問題を議論する。

キーワード：教育向けマイコンボード，プログラミング，学習

Validation of the button pressing method of microcomputer board for education “micro:bit” and discussion about problems that may occur in programming education

YADA YOKO[†1] OKADA SHIGERU[†1] NAKANISHI YOSHIHIKO[†2]　EMI KEIJI[†2]

Abstract: We have validated the button pressing method of microcomputer board for education “micro:bit”. And we discuss about problems that may occur in programming education from the view point of embedded technology. The button pressing methods have “was_pressed”, “is_pressed”, “get_presses”. We discuss about the problems caused by the button pressing timing of the Rock-paper-scissors program.

Keywords: microcomputer board for education, programming, learning

1. はじめに

1.1 初心者向けのプログラミング支援

小学校の活動の中で，プログラミング的思考を指導することになり，数年前から本研究会を中心に様々な教育実践が報告されている．特に，教育向けマイコンボード BBC micro:bit を用いた報告は多数報告されている．教育向けマイコンボード micro:bit は加速度センサー，温度センサー，サウンド再生が可能なため，様々な教科で利用することができる上，教材が比較的安価であることが，教育実践で多い理由であろう．加えて，ブロック言語である MS MakeCode，ウェブ言語 JavaScript，そして Micro Python を利用することが可能であり，最近では Scratch や Python も利用可能となっている．また，限定的ながらも C/C++も利用可能なことから，プログラミング言語を選ばない教材と言える．

プログラミング言語を選ばないという特徴に注目して，高橋参吉，喜家村奨，稲川孝司は micro:bit で学ぶプログラミングの教材[1],[2]を開発した．小学生はブロック型言語 MakeCode，中学生はウェブ言語 JavaScript，そして高校生は MircoPython または Python を学習することを指向した教材である．小学生という入門者がブロック型言語でプログラミングを修得すれば，ブロック型言語による教育は通常のプログラミング言語への移行が難しくなるため，教育上望ましいかどうかは分からないという指摘は，従来から学会や研究会で行われてきた．そのような言説に対して，高橋らは，ブロック型言語 MakeCode[3]から JavaScript そして MicroPython[4],[5] / Python へスムーズに移行することを指向した言う点で秀逸であるとして筆者らは評価している．

筆者(岡田，中西，江見)は，小学校の活動のプログラミング的思考を指導する方法を開発するために，筆者(岡田)の経営する会社内で micro:bit を用いたプログラミング・ワークショップを数回実施した.それらの実践経験の一部は，情報コミュニケーション学会での教員対象ワークショップ[6]で公開した．教材を開発するにあたり，まず高橋らの教材を分析した．その結果，タイミングを取るときのウェイトをかける方法に困難があることを発見した.その本質が，ボタン押下メソッドである is_pressed()と was_pressed()にあることに起因することを突き止めたのである．

前項で述べたように，高橋らは小学生から高校生まで上がっていける教材を開発したが，筆者(岡田)のような組込みソフトウェア開発の専門家から見れば，高校生から専門家になるための橋渡しができていないことを指摘する．そのため，ボタン押下メソッドである is_pressed() と was_pressed()の違いを真剣に突き止めることが第一の目標

†1 電脳匠工房
Dennou Takumi Cobo
†2 京朋社
Keihousha

である．このような細かな違いは児童や生徒に伝える必要はないが，micro:bit は小さな組込みシステムであることを考慮すると，教員は組込み技術の専門家の指摘する事項をのべる．

1.2 ボタン押下メソッドで問題を起こす場面

ボタン押下メソッドと sleep()の関係を明らかにする必要性を突き止めたのである．たとえば，ジャンケンのプログラムでは，ボタン A とボタン B を押すわけであるが，タイミングによっては，反応しない場合がある．そこでボタン押下メソッドと sleep()関数との関係について調べた．MicroPython の関数は，以下の 3 つ[4], [7],[8],[9],[10]である．

(a) button_a.is_pressed()

指定のボタンが押されていれば True を返し，押されていなければ False を返す

(b) button_a.was_pressed()

デバイスが起動されてから，もしくは前回このメソッドが呼ばれてからボタンが(上から下へ)押されたかによって True または False を返す。このメソットを呼び出すとボタンが押されたかの状態がクリアされるので，再度このメソッドが True を返すようにするには，このメソッドを呼び出す前にボタンが押されなければならない。

(c) button_a.get_presses()

ボタンを押した回数の合計を返し，返す前に回数をゼロにリセットする．

これらの MakeCode では以下の図のようになっているが，MakeCode における同様な実験は行わなかった．

図 1(a) MakeCode の button_a.is_pressed()相等

図 1(b) MakeCode の button_a.was_pressed()相等

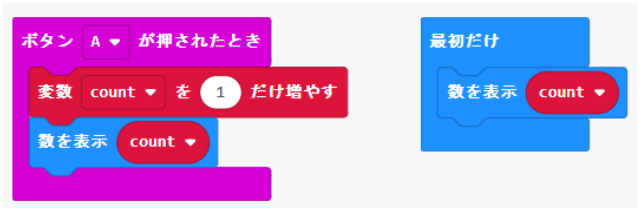


図 1(c) MakeCode の button_a.get_presses()相等

図 1(d) MakeCode の sleep()相等

教材[1](a)はウェブサイト[1](b)にあるので，そこを参照してもらいたい．図 2 に例題 2-1(保存ファイル名：microbit-py-rei2-1)の 2 人用ジャンケンのプログラムに考えることにする．

この問題設定[1]は以下の通りである．

==========

じゃんけんゲームで，ボタン A を押すとと A さんが，ボタン B を押すと B さんが，出した「グー」「チョキ」「パー」のアイコンを表示するプログラムを作成してみよう．なお，「グー」「チョキ」「パー」のアイコンは，--（中略）--を選択する．そして，表示は 2 回使うので関数にし，関数名を「hyouji」にする．

==========

次に，このプログラムのアルゴリズムをフローチャートで図 2(a)に示す．本研究での改良案を図 2(b)に掲載する．またソースコードを図 3 に掲載する．ここでは，button_a.is_pressed()などは button_a.was_pressed()と変更した．また，MicroPython は Mu Editor で編集した．

手続き型のイベント駆動型プログラミング[11]においては，最初に各イベントに対応する処理を記述した手続き（関数やメソッド）を行い，システムあるいはアプリケーションに登録する。この手続きはイベントハンドラーと呼ばれる。イベントハンドラーは，イベントが発生したときにシステムあるいはアプリケーションによってコールバックされる。イベントの待機中の処理はシステムに任せるのである。そして，組込み技術開発[12],[13]ではこの課程を無限ループで繰り返しても動作する必要がある．その観点から改善し，その妥当性を示す実験を以下におこなった．

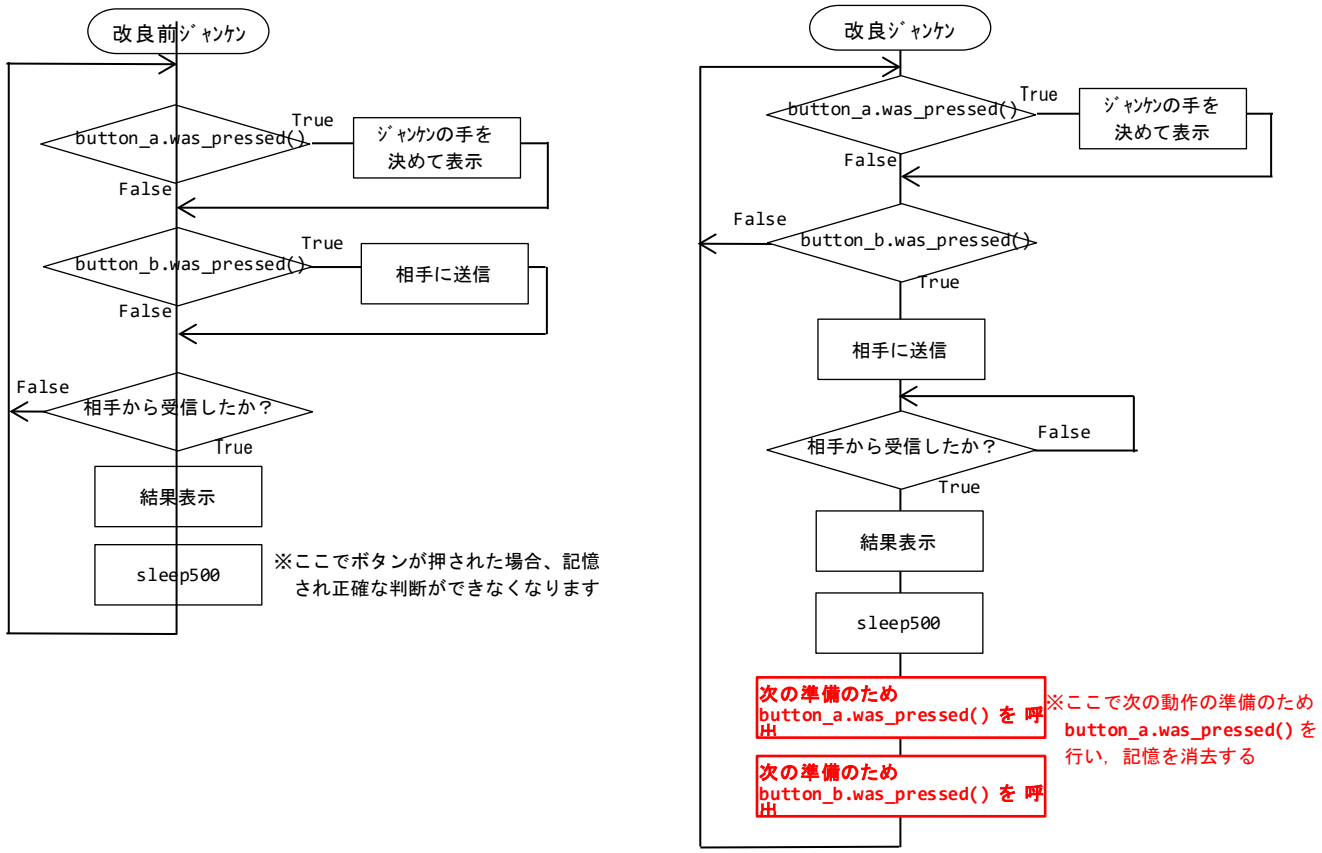


図 2 例題 2-1(保存ファイル名：microbit-py-rei2-1)の２人用ジャンケンの MicroPython プログラムのアルゴリズムを示す．(a) 左側は文献[1]のもの．(b) 右側は本研究での改良案．図 3 のソースコードの 11 行目のあとに sleep(500)と 22 行目のあとに sleep(1000)を追加する

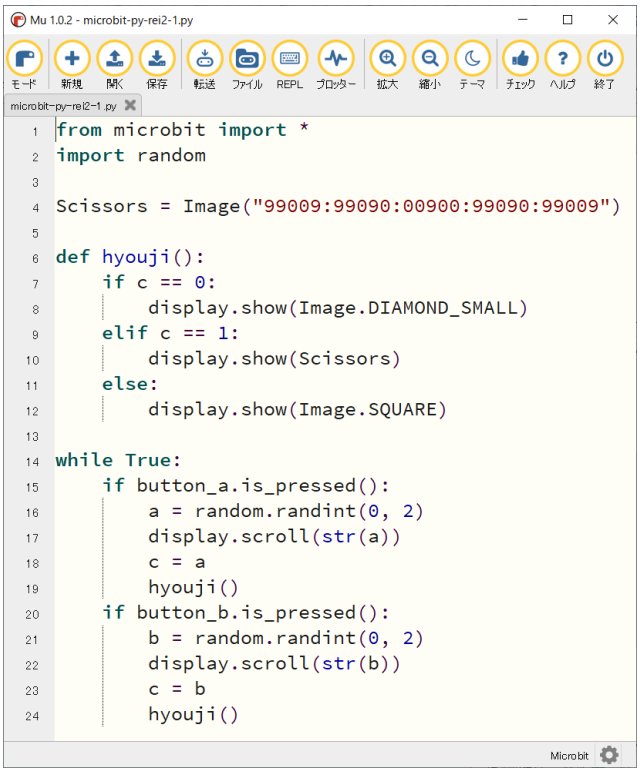


図 3 例題 2-1(保存ファイル名：microbit-py-rei2-1)の２人用ジャンケンの MicroPython プログラムにおいて is を was に変更した．

2. ボタンの状態および有効性を調査する

2.1 実験の概要

組込み技術開発業界で行われているような方法でボタン状態と有効性を確認するために以下の様な試験を実施することにした．これを実験と呼ぶことにする．組込み技術では基本的にソフトウェアは電源を切るまで再起動することなく無限ループで動作させることが求められる．そのため，ある処理の前に前処理を行うことが多いが，本稿ではすべて「準備」と表現する．

＝＝＝＝＝試験＝＝＝＝＝＝＝＝＝

●基本的な確認試験

1.ボタンの状態確認

押しているときと押した後と押していないときの状態をそれぞれ確認する

1.1 ボタンの 1 回押し

(1)is_pressed についての状態確認

(2)was_pressed についての状態確認

(3)get_presses についての状態確認

1.2 ボタンの 2 回押し

1.1 と同じ

1.3 ボタンの長押し

1.1 と同じ

●応用試験

2.ボタンの時間経過確認 (sleep 時に入力後確認)

入力していない状態の時(sleep)にボタンを押したあとボタンの ON を認識しているかを確認する

2.1 ボタンの 1 回押し

(1)is_pressed についての ON を認識しているかの確認

(2)was_pressed についての ON を認識しているかの確認

(3)get_presses についての回数を認識しているかの確認

2.2 ボタンの 2 回押し

2.1 と同じ

2.3 ボタンの 3 回押し

2.1 と同じ

2.4 ボタンの長押し

2.1 と同じ

3.ボタンの入力関数を複数種類呼びボタンの状態に影響するかを確認する

入力している状態の時(sleep でない)にボタンを押したあと複数の入力関数を呼び ON を認識するのに影響するかを確認する．

3.1 ボタンの 1 回押し

3 種類のうちから 2 種類を取り出して実行して影響を調べる

(1)is_pressed と was_pressed を交互に確認

(2)was_pressed と is_pressed を交互に確認

(3)is_pressed と get_presses を交互に確認

(4)get_presses と is_pressed を交互に確認

(5)was_pressed と get_presses を交互に確認

(6)get_presses と was_pressed を交互に確認

3.2 ボタンの 2 回押し

3.1 と同じ．

4.ボタンの入力関数を複数呼び時間経過後の状態に影響するかを確認 (sleep 時に入力後確認)

入力していない状態の時(sleep)にボタンを押したあと複数の入力関数を呼び ON を認識するのに影響するかを確認する

4.1 ボタンの 1 回押し

3 種類のうちから 2 種類を取り出して実行して影響を調べる

(1)is_pressed と was_pressed を交互に確認

(2)was_pressed と is_pressed を交互に確認

(3)is_pressed と get_presses を交互に確認

(4)get_presses と is_pressed を交互に確認

(5)was_pressed と get_presses を交互に確認

(6)get_presses と was_pressed を交互に確認

4.2 ボタンの 2 回押し

4.1 と同じ

4.3 ボタンの長押し

4.1 と同じ

＝＝＝＝＝試験内容は以上＝＝＝＝＝＝＝＝＝

2.2 試験プログラムの実装の考え方

「2.ボタンの時間経過確認の 2.1 ボタンを 1 回押す」という場合を例にとって，以下にソースコードの考え方の例を示す．ディスプレイ表示は以下のように定めた．

O → is_pressed が ON あるいあ was_pressed が ON

X → is_pressed が OFF あるいは was_pressed が OFF

S → sleep を開始する時

E → sleep が終了する時

0，1, , , → get_presses に入っているボタンのカウント数

プログラムを作成するときは以下のようにした．

1 行目に from microbit import*を一番上に書く．

1 行空けて，ON＝True と OFF＝False と書く．

ボタンは A のボタンを使用して動作を確認する．

以下に，試験プログラムを MicroPython のソースコードではなく，日本語を用いたアルゴリズムの説明を用いることにする．組込み技術開発ではよく用いる方法である．この方法であればコーディングも容易になり，コーデレビューの敷居もさがるのである．

●確認方法

sleep が始まったことを知らせるために，ディスプレイに S を表示させる．

sleep(10000)を使って 10 秒プログラムを止めている間にボタンを押す．

sleep が終わったことを知らせるために，ディスプレイに E を表示させる．

ボタンを離す時間を考慮するために sleep(1000)を使って 1 秒プログラムを止める．

ans = (is_pressed / was_pressed / get_presses)でボタンの状態を代入する．

●(1) is_pressed あるいは (2)was_pressed の場合

if ans == ON: → ディスプレイに O を表示．

else: → ディスプレイに x を表示．

表示を確認するために sleep(500)を用意する．

ans = (is_pressed / was_pressed)でボタンの状態を代入する．

if ans == ON: → ディスプレイに O を表示

else: → ディスプレイに x を表示

表示を確認するために sleep(500)を用意する．

ans = (is_pressed / was_pressed)でボタンの状態を入れる．

if ans == ON: → ディスプレイに O を表示．

else: → ディスプレイに x を表示．

表示を確認するために sleep(500)を用意する．

ans = (is_pressed/was_pressed)でボタンの状態を代入する．

if ans == ON: → ディスプレイに O を表示．

else: → ディスプレイに x を表示．

表示を確認するために sleep(500)を用意する．

結果表示を確認するために sleep(500)を用意する．

●(3)get_presses の場合

ディスプレイに ans の数を表示．

表示を確認するために sleep(500)を用意する．

ans = get_presses でボタンの状態を代入する．

ディスプレイに ans の数を表示する．

表示を確認するために sleep(500)を用意する．

ans = get_presses でボタンの状態を入れる．

ディスプレイに ans の数を表示する．

表示を確認するために sleep(500)を用意する．

ans = get_presses でボタンの状態を代入する．

ディスプレイに ans の数を表示する．

3. 実験結果

3.1 基本的な確認試験

以下の様なプログラムで確認した．

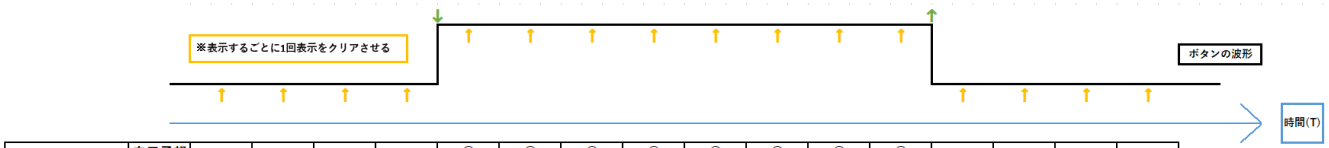
無限ループ(while True)の中で

ans = (is_pressed / was_pressed / get_presses)でボタンの状態を入力する．

一連の処理が終わった後は初期化する．

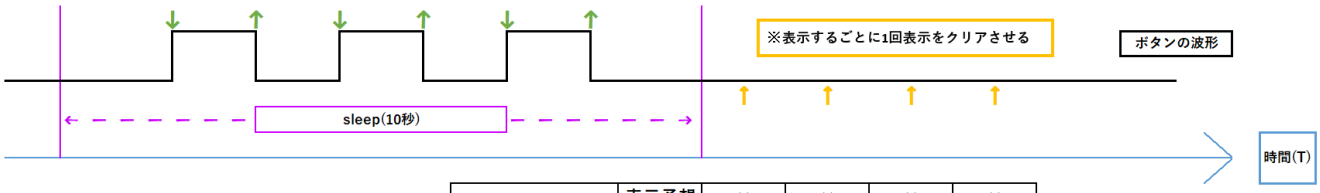
図 4 にその結果の例を示す．

メソッド is_pressed は 押下している間はずっと，TRUE を返す．一方，メソッド was_pressed は押下した直後だけ TRUE を返す．メソッド get_presses は押下した直後だけ 1 と表示する．



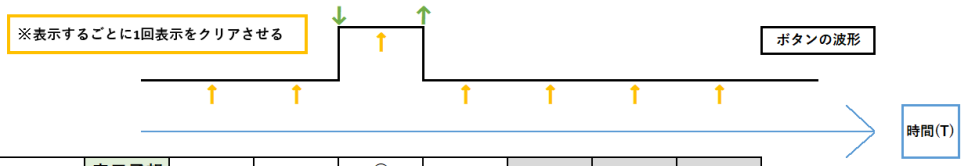
(1)is_pressed	表示予想	×	×	×	×	○	○	○	○	○	○	○	○	×	×	×	×
	表示結果	×	×	×	×	○	○	○	○	○	○	○	○	×	×	×	×
(2)was_pressed	表示予想	×	×	×	×	○	×	×	×	×	×	×	×	×	×	×	×
	表示結果	×	×	×	×	○	×	×	×	×	×	×	×	×	×	×	×
(3)get_presses	表示予想	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
	表示結果	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

図 4 基本的な確認試験「1.ボタンの状態確認」の例として「1.3 ボタンの長押し」の試験概要と結果



(1)is_pressed	表示予想	×	×	×	×
	表示結果	×	×	×	×
(2)was_pressed	表示予想	○	×	×	×
	表示結果	○	×	×	×
(3)get_presses	表示予想	3	0	0	0
	表示結果	3	0	0	0

図 5 応用試験「2.ボタンの時間経過確認　(sleep 時に入力後確認)」の例として「2.3 ボタンの 3 回押し」の試験概要と結果



is_pressed	表示予想	×	×	○	×	—	—	—
	表示結果	×	×	○	×	—	—	—
was_pressed	表示予想	—	—	—	—	○	×	×
	表示結果	—	—	—	—	○	×	×

図 6　応用試験「3.ボタンの入力関数を複数種類呼びボタンの状態に影響するかを確認する」の例として「3.1 ボタンの 1 回押し」の試験概要と結果

is_pressed	表示予想	×	—	×	—	×	—
	表示結果	×	—	×	—	×	—
get_presses	表示予想	—	1	—	0	—	0
	表示結果	—	1	—	0	—	0

図 7(a)　応用試験「4.ボタンの入力関数を複数呼び時間経過後の状態に影響するかを確認　(sleep 時に入力後確認) 」の例として「4.1 ボタンの 1 回押し(3)is_pressed と get_presses を交互に確認」の試験概要と結果

was_pressed	表示予想	○	—	×	—	×	—
	表示結果	○	—	×	—	×	—
get_presses	表示予想	—	1	—	0	—	0
	表示結果	—	1	—	0	—	0

図 7(b)　応用試験「4.ボタンの入力関数を複数呼び時間経過後の状態に影響するかを確認　(sleep 時に入力後確認) 」の例として「4.1 ボタンの 1 回押し(5)was_pressed と get_presses を交互に確認」の試験概要と結果

3.2 ボタンの時間経過確認 (sleep 時に入力後確認)

図 5 にその結果の一例を示す.

3 つの前項からメソッド is_pressed は 押下している間はずっと，TRUE を返すので，今回の試験では当然，FALSE を返す．一方，メソッド was_pressed は sleep 中に押下しても TRUE を返す．メソッド get_presses は sleep 中に押下しても押下した回数に応じて表示する.

3.3 ボタンの入力関数を複数種類呼びボタンの状態に影響するかを確認する

図 6 にその結果の一例を示す.

入力している状態の時(sleep でない)にボタンを押したあと複数の入力関数を呼び ON を認識するのに影響するかを確認したが，3 種類のうちから 2 種類を取り出して実行して影響を調べたがそれぞれで干渉することはなかった.

3.4 ボタンの入力関数を複数呼び時間経過後の状態に影響するかを確認 (sleep 時に入力後確認)

図 7 にその結果の一例を示す.

入力していない状態の時(sleep)にボタンを押したあと複数の入力関数を呼び ON を認識するのに影響するかを確認したが，3 種類のうちから 2 種類を取り出して実行して影響を調べたがそれぞれで干渉することはなかった.

メソッド is_pressed は 押下している間はずっと，TRUE を返すので，今回の試験では当然，FALSE を返す．したがって，その後にメソッド get_presses で取得しても回数は 0 になる.

一方，メソッド was_pressed は sleep 中に押下しても TRUE を返す．メソッド get_presses はしたがって，その後にメソッド get_presses で取得しても回数は 0 になる.

3.5 結果のまとめ

上記の結果より

(1)is_pressed は状態を通知

(2)was_pressed はボタンの押下の変化を検知し通知

(3)get_presses はボタンの押下回数を計数し通知

と確証できる .

4. まとめ

4.1 ボタン押下メソッド

基本的にボタンは was_pressed()を使用したほうが良いと考える。メソッド is_pressed()はボタンの状態が通知される。状態なので押下時は実行回数分となる. Trueが返送される。そのため押下される度に True の処理を実行する。

一方, メソッド was_pressed()は変化が通知される。変化であるので押下時は 1 回だけ True が返送され，それ以後は False が返送される。メソッド was_pressed()で注意すべき点は処理に関係なく変化を記憶しているということである。例えば sleep()時にボタンが押下されると，変化として記憶される。

どのように防ぐか(対処するか)はボタンを有効にする前に「準備」として「was_pressed()を 1 回空読み」することで記憶が消去され誤認識を防ぐことができる。

なおプログラミングで get_presses()は何度使用しても問題がないことを確認した.

4.2 じゃんけんゲーム 2 人対戦に更なる改良版

高橋参吉ほか[1]の (ソースコードファイル名 : microbit-py-hat7-2)に更に改良を加えた．特にボタン A とボタン B の同時押しの部分の改良を加えた. 図 8 に掲載する.

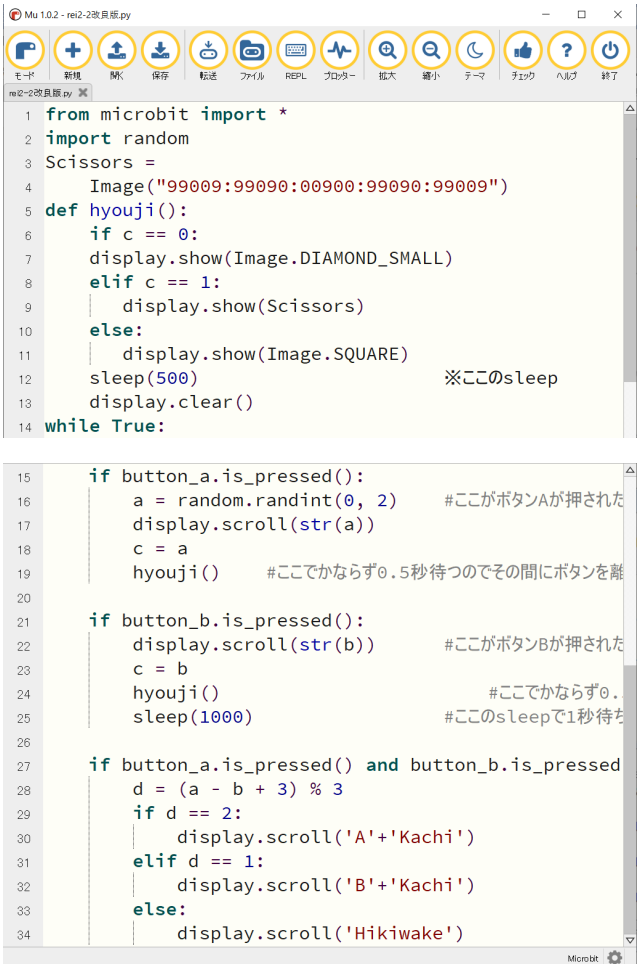


図 8 例題 2-1(保存ファイル名 : microbit-py-rei2-1)の 2 人用ジャンケンの MicroPython プログラムを更に改良した実際の Python のコード.

4.3 じゃんけんゲーム 3 人対戦

3 人対戦のジャンケンともなれば，個々のマイクロビットの通信方式まで詳細に検討する必要がある．今後はこの点にもついて検討した結果を発表したいと考えている．パ

ソコン環境では，通信は TCP/IP 方式を用いるので，比較的楽に実装することは可能であるが，MicroPython では無線通信の根本から考える必要があるため，教材としては難しくなる．

4.4 教員研修用教材

今回，実験で用いた手法は産業界で用いられている手法である．これらの知見を教員研修用の教材にすれば，一段高い部分からプログラミングを捉える力がつくことを期待することが可能である．機会があれば，ワークショップを実施すること[14]を願っている．

謝辞 本研究に関して，IEC 情報教育学研究会の会員の皆様にご協力頂いたことに感謝の意を述べる．

参考文献

[1] (a) 高橋　参吉, 喜家村　奨, 稲川　孝司, micro:bit で学ぶプログラミング- ブロック型から JavaScript そして Python へ - , 2019, コロナ社，pp.1-128.; (b) micro:bit によるプログラミング, https://www.u-manabi.net/microbit/　(2021 年 1 月 16 日閲覧)
[2] https://microbit-micropython.readthedocs.io/ja/latest/button.html /　(2021 年 1 月 16 日閲覧)
[3] Microsoft MakeCode for micro:bit https://makecode.microbit.org/　(2021 年 1 月 16 日閲覧)
[4] BBC micro:bit MicroPython ドキュメンテーション　BBC micro:bit MicroPython 1.0.1 ドキュメント https://microbit-micropython.readthedocs.io/ja/latest/ /　(2021 年 1 月 16 日閲覧)
[5] Code With Mu https://codewith.mu//　(2021 年 1 月 16 日閲覧)
[6] 岡田　繁，中西　祥彦，江見　圭司，アルゴリズムよりも大切なプログラミングの心得，情報コミュニケーション学会第 15 回情報教育合同研究会ワークショップ，2019
[7] Simon Monk, Programming the BBC Micro:bit: Getting Started With MicroPython, Tab Books, 2017, pp.1-165
[8] Nicholas H. Tollervey, Programming with MicroPython: Embedded Programming with Microcontrollers and Python, O'Reilly Media, 2017, p.1-214
[9] Charles Bell, MicroPython for the Internet of Things: A Beginner's Guide to Programming with Python on Microcontrollers, 2017, pp.1-472
[10] Marwan Alsabbagh, MicroPython Cookbook: Over 110 practical recipes for programming embedded systems and microcontrollers with Python, 2019, Packt Publishing, pp.1-452
[11] 組込みシステム技術協会エンベデッド技術者育成委員会，エンベデッド技術 改訂，電波新聞社，2009，pp.1-370
[12] 酒井由夫，リアルタイム OS から出発して 組込みソフトエンジニアを極める[改装版] ，星雲社，2016，pp.1-278
[13] 鷹合　大輔, 田村　修，組込み開発のための実践的プログラミング，近代科学社 2018，pp.1-240
[14] WCCE2022 http://wcce2022.org/　(2021 年 1 月 16 日閲覧)