Digi-Sign Player

画面コントローラーを持つ、リモート・デジサイン・プレイヤー

kobayashikorio@gmail.com
2018/5/15

■ **Digi-Sign Player**

The project “Digi-player” is intended to manipulate a slide show on a notebook which created and written by the program, and in this case, an examle slide show is “Mathematica program philosophy.” Because the slides are shown on the notebook, an input is written by the program and the result will be created by the Wolfram language evaluation. This Digi-player is a stand-alone system working on the RaspberryPi Zero connected to a HDMI display. System includes hardware electro-static switches to manage slide show, stop-restart-forward-backward functions.

This program is composed of following specific routines using;

- Dataset of associations for the contents
- Note book writing from the kernel for the running script
- GPIO connected electro-static switch for the controller
- Task scheduling I/O driver for un-synchronous control
- Object oriented program method for the switch driver

■ hardware

1. RaspberryPi Zero W
2. 4-ch touch switch TTP224

■ pin connection

RaspberryPi Zero W <-> 4-ch touch switch TTP224
V3.3V -> Vcc
GND -> GND
BCM19 -> OUT1
BCM16 -> OUT2
BCM37 -> OUT3
BCM38 -> OUT4
Shutdown switch
BCM05 -> 1
GND -> 2

■ Environment setup

1. Display resolution 800x600 56Hz is suitable
2. program cells must be set to Initialization-cell
3. Options for the cell initialization must be set as,
   Format -> Options for Global Preferences -> Global Preferences -> Notebook Options -> Evaluation Options -> InitializationCellWarning -> False
   Format -> Options for Global Preferences -> Global Preferences -> Notebook Options -> Evaluation Options -> InitializationCellEvaluation -> True
4. To autostart the program, RaspberryPi PIXEL Desktop autostart set-up
   add a next line to the file, /home/pi/.config/lxsession/LXDE-pi/autostart
   @/opt/Wolfram/WolframEngine/11.0/Executables/mathematica /home/pi/test.nb

■ Usage

1. abort & debug -> Press reset button before opening the window

2. shutdown the system -> Press reset button anytime after the slide show begun

```
(* digi-sign contents *)
slides = Dataset[{
     Association[type → page, title -> "Mathematica\n\nPhilosophy of Programming"],
     Association[type → page, title -> "\n\nSymbol and Expression"],
     Association[type → exec, title -> "Blank Symbol", contents → "_"],
     Association[type → exec, title -> "Expression", contents → "_[]"],
     Association[type → exec, title -> "Expression with Blank", contents → "_[_]"],
     Association[type → exec, title -> "Expression Extended", contents → "_[_,_]"],
     Association[type → exec, title -> "Expression Nested", contents → "_[_[]]"],
     Association[type → exec, title -> "Expression Serialized", contents → "_[_][_]"],
     Association[type → exec, title -> "Expression Extended, Nested and Serialized",
      contents → "_[_, _[][_]]"],
     Association[type → exec, title -> "Pattern named \"a\"", contents → "a_"],
     Association[type → exec, title -> "Expression named \"x\"", contents → "x[_]"],
     Association[type → exec,
      title -> "Sequences of Evaluation", contents → "x[_];x[_,_]"],
     Association[type → page, title -> "\n\nSet something to Symbol"],
     Association[type → exec,
      title -> "Predefined Expression \"Set\"", contents → "Set[x,1]"],
     Association[type → exec, title -> "Evaluate Symbol \"x\"", contents → "x"],
     Association[type → exec,
      title -> "Predefined Expression \"Plus\"", contents → "Plus[x,a]"],
     Association[type → page, title -> "\n\nSet something to Expression"],
     Association[type → exec, title -> "Evaluate Definition", contents → "f[b]"],
     Association[type → exec2,
      title -> "Syntax Sugar for Expression Definition with Pattern",
      contents → {"f[a_]=1+a", "f[2]"}],
     Association[type → exec, title -> "Expression Predefined, \"List\"",
      contents → "List[1,2,a]"],
     Association[type → exec, title -> "Syntax Sugar for \"List\"",
      contents → "{1,☺,a,x[2], a=2,{a, 0}}"],
     Association[type → page, title -> "\n\nSet and SetDelayed"],
     Association[type → exec, title -> "Set", contents → {"a=2;\nf[a_]=1+a", "f[y]"}],
     Association[type → exec2,
      title -> "SetDelayed", contents → {"a=2;\nf[a_]:=1+a", "f[y]"}],
     Association[type → page, title -> "\n\nListable"],
     Association[type → exec, title -> "Plus for List", contents → "Plus[{1,2,b},c]"],
     Association[type → exec, title -> "Plus is Listable", contents → "{1,2,b} +c"],
     Association[type → exec,
      title -> "Times for List", contents → "Times[{1,2,b}, z]"],
     Association[type → exec, title -> "Apply Times and Plus to List",
      contents → "{1,2,b}z+ c"],
     Association[type → exec, title -> "Apply Expression to List",
      contents → "Map[g,{2,3,4,b}]"],
     Association[type → exec, title -> "Apply User Defined Expression to List",
      contents → "f[a_]:=1+a;\nMap[f,{1, 2, 3, b}]"],
     Association[type → exec, title -> "Make List", contents → "Table[i, {i, 5}]"],
     Association[type → page, title -> "\n\nBeyond the Symbol"],
     Association[type → exec2, title -> "Symbol is member of the Ring",
      contents → {"{_*1, _*0, _^-1*_}", "{s+s,s*s,s^-1,s*s^-1}"}],
     Association[type → exec, title -> "So, Any Symbol can name the Expression",
      contents → "{0[], ☺[], {1}[], Sin[_]}"],
     Association[type → page, title -> "\n\nPure Function"],
     Association[type → exec2, title -> "Function and Pure Function",
      contents → {"f[a_]:=a^2;\nMap[f,{1, 2, 3, b}]", "Map[#^2&,{1,2,3,b}]"}],
     Association[type → exec, title -> "Pure Function is also Listable",
      contents → "{#,#^2,#^3}&[{1,3}]"],
     Association[type → exec, title -> "List Element Selection with Pure Function",
      contents → "Select[{2,3,4,5},#>3&]"],
     Association[type → exec2, title -> "Pure Function returned from Evaluation",
      contents → {"h[x_]:=x^3+x^2; h'", "h'[2]"}],
     Association[type → page, title -> "\n\nList Oriented Programming"],
     Association[type → exec2, title -> "Replacement by Rules",
      contents → {"Unset[x];\nReplaceAll[{x,1,z},x->5]", "{x,y,z}/.x->5"}],
     Association[type → exec2, title -> "Replace Expression Applying Rules",
      contents → {"g=p*z+q/.z->y", "g/.{q->2, p->3}"}],
```

```
    Association[type → exec2, title -> "Replace List Components",
     contents → {"{1,2,3,4,5,6,7}/.(2|4|7)->w", "{i,j,k,l}/.{j->k,k->j}"}],
    Association[type → exec2, title ->
      "Applying Answer of Solve to the Original Equation",
     contents → {"x =.;\nans=Solve[y^2==x, y]", "y^2==x/. ans"}],
    Association[type → exec2, title -> "Replace List Components with Conditioned Rule",
     contents → {"x= 9;\n{2, 3, 4}/.x_/;x>3 ->0",
       "x=.;\n{{1,2},{a,1}}/.{x_,y_}->x^2+y^2"}],
    Association[type → exec2, title -> "Replace List Components with Pattern Test",
     contents → {"x=1;\n{1,2,3,4}/.x_?(#>2 &):>x+1",
       "{{1,2},{2,4}}/.\n{x_/;EvenQ[x],y_}->{y,x}"}],
    Association[type → exec2, title -> "Pattern Priority",
     contents → {"q[_]= -1;\nq[1]=0;\nq[x_Integer]:=1;\nq[x_List]:=Total[x];",
       "{q[2],q[1],q[{1,2,3}],\nq[a],q[a=1],q[♫]}"}],
    Association[type → exec, title -> "Direct Replacement for List Component",
     contents → "n={1,2,3,4,5,6,7};\nn[[3;;5]]= ☺;\nn"],
    Association[type → page, title -> "\n\nProgramming Paradigm"],
    Association[type → exec2, title -> "Procedural programming",
     contents → {"sum= 0;\nDo[sum =sum+i, {i, 10}]", "sum"}],
    Association[type → exec, title -> "List Oriented programming",
     contents → "Apply[Plus, Range[10]]"],
    Association[type → exec2, title -> "Functional Programming",
     contents → {"sumF[0]= 0;\nsumF[n_/;n>0]:=n+sumF[n-1]", "sumF[10]"}],
    Association[type → exec, title ->
      "Object Oriented Programming - Class definition", contents →
      "class[nam_]:= \nModule[{a}, \nput[nam[x_]]^:=a= x;\nget[nam]^:= a;];"],
    Association[type → exec, title ->
      "Object Oriented Programming - Instance Construction",
     contents → "class[alpha];\nclass[beta];"],
    Association[type → exec2, title ->
      "Object Oriented Programming - Instance Execution",
     contents → {"{put[alpha[2]],put[beta[1]]}", "{get[alpha],get[beta]}"}],
    Association[type → page, title -> "end of slides"]
   }
  ];

(*display font size definition*)
inputSize = 38; inputSmallSize = 30;
outputSize = 38; outputSmallSize = 30;
titleSize = 40;
textSize = 50;
innerInterval = 2;
innerIntervalInit = 2;
pageInterval = 2;
pageNumber = 1;
```

```
(*clear note book cells*)
clearWindow := (SelectionMove[nb, All, Notebook];
  NotebookDelete[nb];
  NotebookWrite[nb, Cell[ToString[pageNumber], "Text"]];
 )

(*text page write on the note*)
showText[text_] := (clearWindow;
  NotebookWrite[nb, Cell[text, "Text", textSize, Bold, Background → LightBlue]]
 )

(*slide page increment and decrement*)

pageControlPlus := pageNumber = Mod[pageNumber + 1, Length[slides], 1];
pageControlMinus := pageNumber = Max[pageNumber - 1, 1];

(*show input and output*)
showExec[title_, contents_] := (clearWindow;
  NotebookWrite[nb, Cell[title, "Subsection", titleSize]];
  SelectionMove[nb, Next, Cell];
  evalDo[contents])

(*show input and output including internal wait*)
showExec2[title_, contents_] := (clearWindow;
   NotebookWrite[nb, Cell[title, "Subsection", titleSize]];
   evalDo[contents[[1]]];
   sTime = SessionTime[] + innerInterval;
   While[And[sTime > SessionTime[], status[stop] == "0"], None];
   evalDo[contents[[2]]]
  );

(*evalDo*)
evalDo[contents_] := (SelectionMove[nb, Next, Cell];
  NotebookWrite[nb, Cell[contents, "Input", GrayLevel[0.1], inputSmallSize]];
  sTime = SessionTime[] + innerInterval;
  While[And[sTime > SessionTime[], status[stop] == "0"], None];
  SelectionMove[nb, Next, Cell];
  output = ToString[Evaluate[ToExpression[contents], StandardForm]];
  NotebookWrite[nb,
   Cell[TextData[{"evaluated.. ", Cell[BoxData[output], Bold, Blue, 32]}]]]
 )

(*slide format changer*)
showPage := (this = Normal[slides[[pageNumber]]];
   Switch[this[type], page, showText[this[title]], exec, showExec[this[title],
     this[contents]], exec2, showExec2[this[title], this[contents]], None, {}];);

(*switch Class*)
gpio[nam_] := Module[{process, comstr, schTask},
   set[nam[pin_]] ^:= (process = StartProcess[$SystemShell];
     comstr = "gpio -g read " <> ToString[pin];
     schTask = CreateScheduledTask[WriteLine[process, comstr];
       status[nam] = ReadLine[process], 0.1];);
   start[nam] ^:= StartScheduledTask[schTask];];

 (*setup switch for shutdown process*)
swProcess = StartProcess[$SystemShell];
WriteLine[swProcess, "gpio -g mode 5 in"];
WriteLine[swProcess, "gpio -g read 5"];
If[ReadLine[swProcess] == "0", Exit[]];
```

```
(*control switch-pin definition*)
gpioPin = Association[stop → 19, start → 16, forward → 26, backward → 20];

(*construction of instances,initialize and start*)Map[gpio, Keys[gpioPin]];
Map[set, KeyValueMap[#1[#2] &, gpioPin]];
Map[start, Keys[gpioPin]];

(*prepare note book window*)

nb = CreateWindow[WindowMargins → {{0, 0}, {0, 0}},
   WindowFrame → "ModalDialog", WindowTitle → None, WindowElements → None];

(*initialize slide show task*)
task = CreateScheduledTask[pageControlPlus;
   showPage, pageInterval];
(*page show task*)
StartScheduledTask[task];
 (*mail loop*)
While[True,
 WriteLine[swProcess, "gpio -g read 5"];
If[ReadLine[swProcess] == "0",
  NotebookClose[nb]; $Epilog := WriteLine[swProcess, "sudo shutdown -h now"];
Exit[]];
 If[status[stop] == "1", StopScheduledTask[task]; innerInterval = 0];
 If[status[start] == "1",
  StartScheduledTask[task]; innerInterval = innerIntervalInit];
 If[status[forward] == "1", pageControlPlus; showPage];
 If[status[backward] == "1", pageControlMinus; showPage];]
```