pysimavrgui Documentation

Release 0.0.1

ponty

November 18, 2011

CONTENTS

pysimavrgui

Date November 18, 2011

PDF pysimavrgui.pdf

Contents:

Simple GUI elements for AVR and arduino simulation. Programmed in python, based on pygame. Simavr is used for simulation.

Links:

- home: https://github.com/ponty/pysimavrgui
- documentation: http://ponty.github.com/pysimavrgui

Features:

- designed to use with pysimavr (simavr wrapper)
- arduino simulator included
- maximum speed can be real-time
- speed control
- audio backend: PyAudio
- graphic backend: PyGame (SDL wrapper)

Known problems:

- Python 3 is not supported
- tested only on linux
- real-time sleep() is used in simavr, so speed control is far from perfect
- occasional crash by firmware reload
- poor sound quality

CHAPTER

ONE

INSTALLATION

1.1 General

- install python
- install setuptools
- install PyGame
- install PyAudio (optional)
- install pysimavr
- install the program:

```
# as root
easy_install pysimavrgui
```

1.2 Ubuntu

```
sudo apt-get install python-setuptools
sudo apt-get install python-pygame
sudo apt-get install python-pyaudio

# pysimavr
sudo apt-get install swig
sudo apt-get install python-dev
sudo apt-get install gcc
sudo apt-get install libelf-dev
sudo easy_install pysimavr

sudo easy_install pysimavrgui
```

1.3 Uninstall

first install pip:

```
# as root
pip uninstall pysimavrgui
```

CHAPTER

TWO

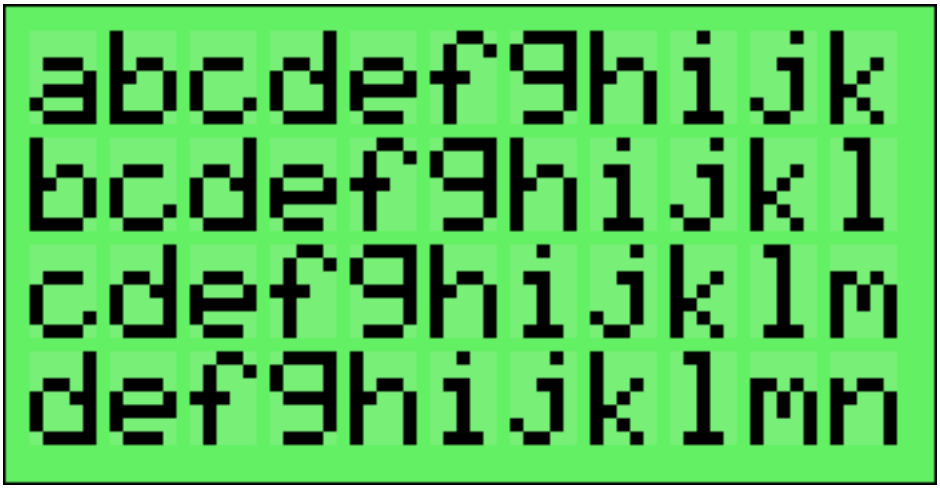
GUI EXAMPLES

These examples have no simulation, they test only GUI.

2.1 LCD

```
from entrypoint2 import entrypoint
from pysimavrgui.lcdgame import LcdGame
from pysimavrgui.maingame import MainGame

@entrypoint
def start():
    def char_func(x,y):
        return chr(ord('a')+x+y)
    lcd = LcdGame(char_func,(11,4))
    MainGame(lcd).run_game()
```

```
$ python -m pysimavrgui.examples.gui.lcdgame_ex
```



2.2 LED row

```
from entrypoint2 import entrypoint
from pysimavrgui.ledrowgame import LedRowGame
```

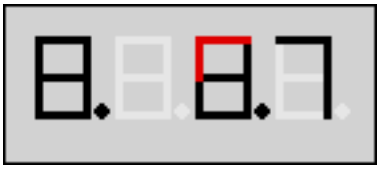
```
from pysimavrgui.maingame import MainGame

@entrypoint
def start():
    def func(i):
        return (i>1,i>2)
    dev = LedRowGame(func,disp_size=4,labels=['x','y','z'])
    MainGame(dev).run_game()
```

```
$ python -m pysimavrgui.examples.gui.ledrowgame_ex
```

2.3 7 segment display

```
from entrypoint2 import entrypoint
from pysimavrgui.maingame import MainGame
from pysimavrgui.sgm7game import Sgm7Game


@entrypoint
def start():
    def func(x):
        return [
                (255,0),
                (0,0),
                (255,33),
                (7,0),
                ][x]
    dev = Sgm7Game(func,4)
    MainGame(dev).run_game()
```

```
$ python -m pysimavrgui.examples.gui.sgm7game_ex
```

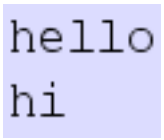


2.4 Text

```
from entrypoint2 import entrypoint
from pysimavrgui.compgame import CompositeGame
from pysimavrgui.maingame import MainGame
from pysimavrgui.textgame import TextGame

@entrypoint
def start():
    def func1():
        return 'hello'
```

```
    def func2():
        return 'hi'
    dev1 = TextGame(func1)
    dev2 = TextGame(func2)
    dev=CompositeGame([dev1,dev2],align=1)
    MainGame(dev).run_game()
```

```
$ python -m pysimavrgui.examples.gui.textgame_ex
```



CHAPTER

THREE

SIMULATION EXAMPLES

These examples have simulation.

3.1 ledramp

Program:

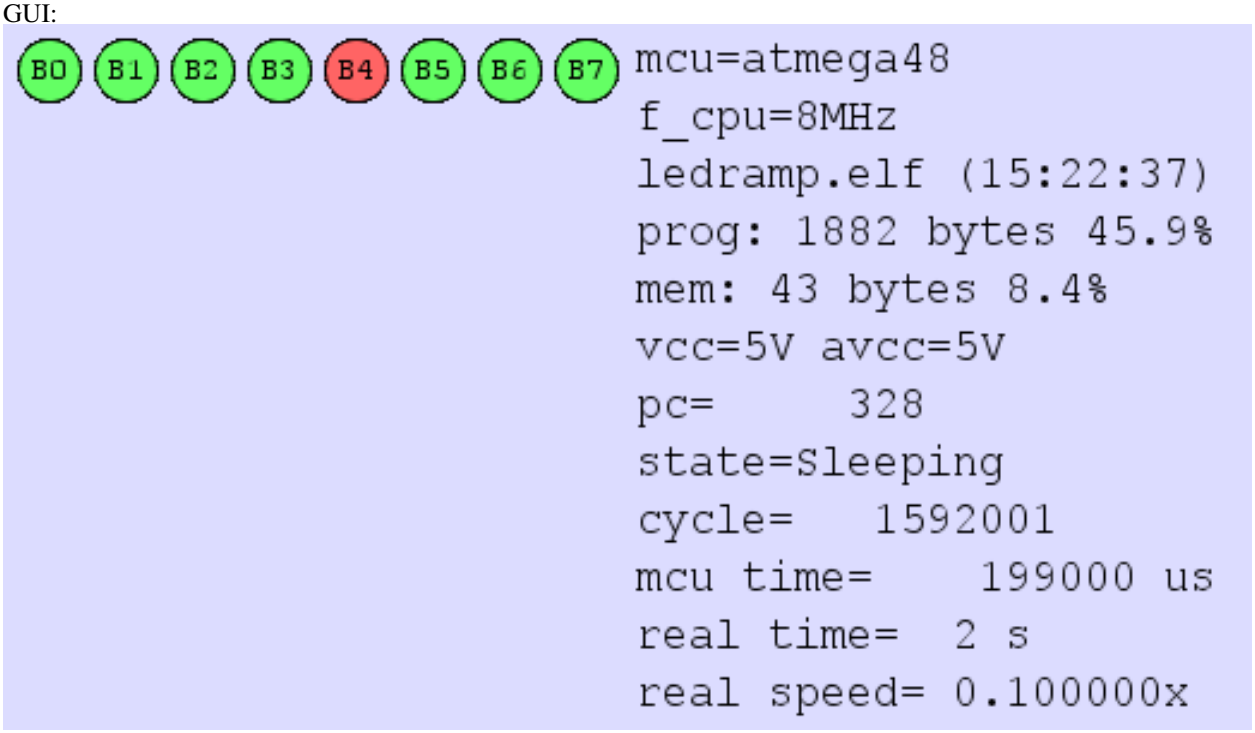
```
from entrypoint2 import entrypoint
from path import path
from pysimavr.avr import Avr
from pysimavr.connect import connect_pins_by_rule
from pysimavrgui.examples.sim.avrsimmain import AvrSimMain
from pysimavr.firmware import Firmware
from pysimavrgui.compgame import CompositeGame
from pysimavrgui.infogame import InfoGame
from pysimavrgui.ledrowgame import LedRowGame
from pysimavr.ledrow import LedRow
from pysimavr.vcdfile import VcdFile

@entrypoint
def run_sim(vcdfile='ledramp.vcd', speed=0.1, fps=20, timeout=0.0, visible=1, image_file=''):
    firmware = Firmware(path(__file__).dirname() / 'ledramp.elf')
    avr = Avr(firmware, f_cpu=8000000, mcu='atmega48')

    vcd = VcdFile(avr, period=1000, filename=vcdfile)
    ledrow = LedRow(avr)
    connect_pins_by_rule('''
                         avr.B0 ==> led.0 -> vcd
                         avr.B1 ==> led.1 -> vcd
                         avr.B2 ==> led.2 -> vcd
                         avr.B3 ==> led.3 -> vcd
                         avr.B4 ==> led.4 -> vcd
                         avr.B5 ==> led.5 -> vcd
                         avr.B6 ==> led.6 -> vcd
                         avr.B7 ==> led.7 -> vcd
                         ''',
                          dict(
                              avr=avr,
                              led=ledrow,
                              ),
                          vcd=vcd,
    )
```

```
    def state_func(i):
        return (ledrow.pinstate(i), ledrow.reset_dirty(i))
    led_game = LedRowGame(state_func=state_func,
                          labels=['B' + str(x) for x in range(8)])

    dev = CompositeGame([
                        led_game,
                        InfoGame(avr),
                        ])

    scrshot_by_exit = [(dev, image_file)] if image_file else None

    AvrSimMain(avr, dev, vcd, speed=speed, fps=fps, visible=visible, timeout=timeout,
               scrshot_by_exit=scrshot_by_exit).run_game()
```

Starting program:

```
>>> from pysimavrgui.examples.sim.ledramp import run_sim
>>> run_sim(vcdfile='docs/ledramp.vcd', speed=0.1, timeout=0.2, fps=50, visible=0, image_file='docs/l
Loaded 1850 .text
Loaded 32 .data
Starting atmega48 - flashend 0fff ramend 02ff e2end 00ff
atmega48 init
avr_timer_reconfigure-2 clock turned off
avr_timer_configure-2 TOP 4096.00Hz = 1953 cycles
avr_timer_write_ocr-2 mode 2 UNSUPPORTED
avr_timer_configure-2 TOP 64.00Hz = 125000 cycles
avr_timer_configure-2 A 64.00Hz = 125000 cycles
```

GUI:



Signals:

3.2 LCD

Program:

```
from entrypoint2 import entrypoint
from path import path
from pysimavr.ac import Ac
from pysimavr.avr import Avr
from pysimavr.connect import connect_pins_by_rule
from pysimavrgui.examples.sim.avrsimmain import AvrSimMain
from pysimavr.firmware import Firmware
from pysimavrgui.compgame import CompositeGame
from pysimavrgui.infogame import InfoGame
from pysimavrgui.lcdgame import LcdGame
from pysimavrgui.ledrowgame import LedRowGame
from pysimavr.lcd import Lcd
from pysimavr.ledrow import LedRow
from pysimavr.vcdfile import VcdFile

@entrypoint
def run_sim(vcdfile='lcd.vcd', speed=0.1, fps=20, timeout=0.0, visible=1, image_file=''):
    firmware = Firmware(path(__file__).dirname() / 'lcd.elf')
    avr = Avr(firmware,f_cpu=16000000)
    lcd = Lcd(avr)
    ledrow = LedRow(avr, size=7)
    # period=1000 -> vcd error
    vcd = VcdFile(avr, period=10, filename=vcdfile)

    def state_func(i):
        return (ledrow.pinstate(i), ledrow.reset_dirty(i))
    led_game = LedRowGame(state_func=state_func,
                          labels='D4 D5 D6 D7 RS E RW'.split()
                          )
    ac = Ac(avr)
    connect_pins_by_rule('''
    avr.B0 <=> lcd.D4 -> vcd
    avr.B1 <=> lcd.D5 -> vcd
    avr.B2 <=> lcd.D6 -> vcd
    avr.B3 <=> lcd.D7 -> vcd

    avr.B4 ==> lcd.RS -> vcd
    avr.B5 ==> lcd.E  -> vcd
    avr.B6 ==> lcd.RW -> vcd
    vcd <- ac.OUT -> avr.D2

    lcd.D4 -> led.0
    lcd.D5 -> led.1
```

```
    lcd.D6 -> led.2
    lcd.D7 -> led.3

    lcd.RS -> led.4
    lcd.E  -> led.5
    lcd.RW -> led.6
                        ''',
                         dict(
                             avr=avr,
                             led=ledrow,
                             lcd=lcd,
                             ac=ac
                             ),
                         vcd=vcd,
                         )

    dev = CompositeGame([
                        CompositeGame(
                                [LcdGame(lambda x, y:lcd.get_char(x, y), (20, 2)),
                                 led_game,
                                 ],
                                  align=1),
                        InfoGame(avr),
                        ])

    scrshot_by_exit = [(dev, image_file)] if image_file else None

    AvrSimMain(avr, dev, vcd, speed=speed, fps=fps, visible=visible, timeout=timeout,
               scrshot_by_exit=scrshot_by_exit).run_game()
```

Starting program:

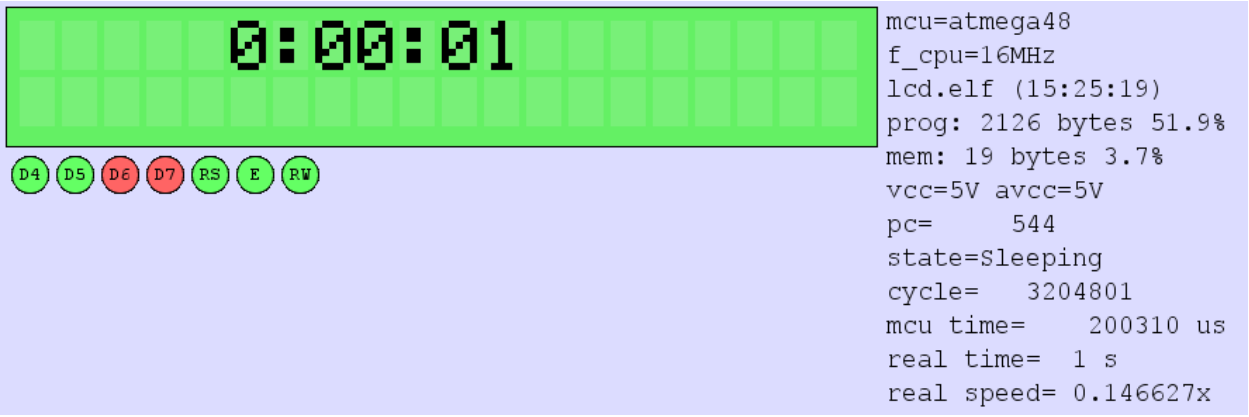
```
>>> from pysimavrgui.examples.sim.lcd import run_sim
>>> run_sim(vcdfile='docs/lcd.vcd', speed=1, timeout=0.2, fps=50, visible=0, image_file='docs/lcd.png
Loaded 2112 .text
Loaded 14 .data
Starting atmega48 - flashend 0fff ramend 02ff e2end 00ff
atmega48 init
LCD: 37uS is 592 cycles for your AVR
LCD: 1uS is 16 cycles for your AVR
ac_input_init period 2000uS or 32000 cycles
hd44780_write_command 33
hd44780_write_command 30
hd44780_process_write command 20 write when still BUSY
hd44780_write_command 20
hd44780_write_command activating 4 bits mode
hd44780_write_command 28
hd44780_write_command 08
hd44780_write_command 01
hd44780_write_command 06
hd44780_write_command 0e
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
```

```
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 31
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 32
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 33
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 34
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 35
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 36
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
```

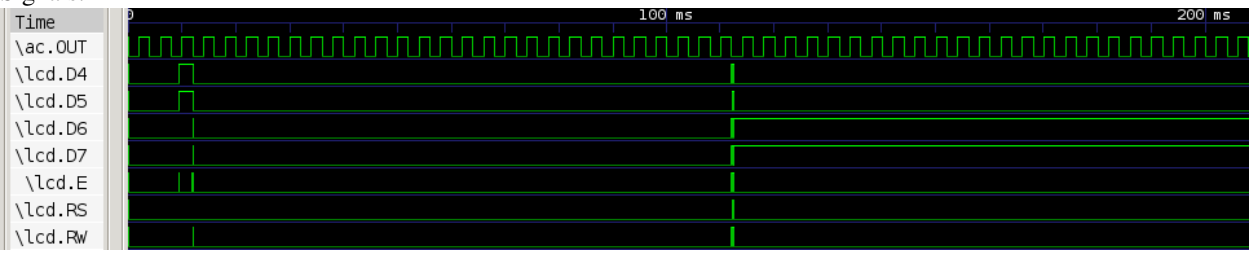
```
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 37
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 38
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 39
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 31
hd44780_write_data 30
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 31
hd44780_write_data 31
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 31
hd44780_write_data 32
hd44780_write_command 01
hd44780_write_command 84
hd44780_write_data 20
```

```
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 30
hd44780_write_data 30
hd44780_write_data 3a
hd44780_write_data 31
hd44780_write_data 33
```

GUI:



Signals:



3.3 seven segment display

Program:

```
from entrypoint2 import entrypoint
from path import path
from pysimavr.avr import Avr
from pysimavr.connect import connect_pins_by_rule
from pysimavrgui.examples.sim.avrsimmain import AvrSimMain
from pysimavr.firmware import Firmware
from pysimavrgui.compgame import CompositeGame
from pysimavrgui.infogame import InfoGame
from pysimavrgui.ledrowgame import LedRowGame
from pysimavrgui.sgm7game import Sgm7Game
from pysimavr.inverter import Inverter
from pysimavr.ledrow import LedRow
from pysimavr.sgm7 import Sgm7
from pysimavr.vcdfile import VcdFile

@entrypoint
def run_sim(vcdfile='sgm7.vcd', speed=0.001, fps=20, timeout=0.0, visible=1, image_file=''):
    firmware = Firmware(path(__file__).dirname() / 'sgm7.elf')
```

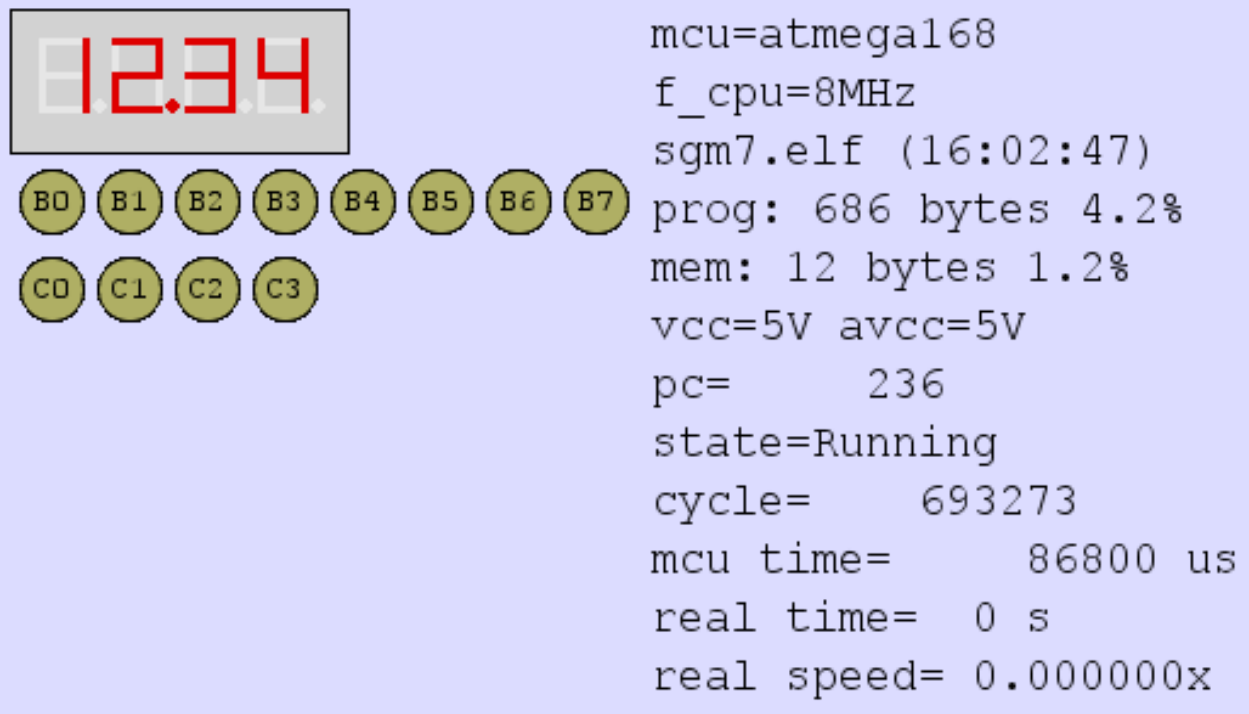
```
firmware.f_cpu = 8000000
firmware.mcu = "atmega168"
avr = Avr(firmware)
vcd = VcdFile(avr, period=1000, filename=vcdfile)

#####################################################
# ledrow
ledrow = LedRow(avr, size=12)

#####################################################
# ledrow game
def state_func_seg(i):
    return (ledrow.pinstate(i), ledrow.reset_dirty(i))
led_game_seg = LedRowGame(state_func=state_func_seg,
                      disp_size=8,
                      labels=['B' + str(x) for x in range(8)]
                      )
def state_func_dig(i):
    return (ledrow.pinstate(i + 8), ledrow.reset_dirty(i + 8))
led_game_dig = LedRowGame(state_func=state_func_dig,
                      disp_size=4,
                      labels=['C' + str(x) for x in range(4)])
#####################################################
# sgm7
sgm7 = Sgm7(avr, size=4)

inv = [Inverter(avr) for x in range(4)]

connect_pins_by_rule('''
                     ledrow.0 <== avr.B0 ==> sgm7.A -> vcd
                     ledrow.1 <== avr.B1 ==> sgm7.B -> vcd
                     ledrow.2 <== avr.B2 ==> sgm7.C -> vcd
                     ledrow.3 <== avr.B3 ==> sgm7.D -> vcd
                     ledrow.4 <== avr.B4 ==> sgm7.E -> vcd
                     ledrow.5 <== avr.B5 ==> sgm7.F -> vcd
                     ledrow.6 <== avr.B6 ==> sgm7.G -> vcd
                     ledrow.7 <== avr.B7 ==> sgm7.P -> vcd

                     ledrow.8 <== avr.C0 ==> inv0.IN | inv0.OUT -> sgm7.D0 -> vcd
                     ledrow.9 <== avr.C1 ==> inv1.IN | inv1.OUT -> sgm7.D1 -> vcd
                     ledrow.10<== avr.C2 ==> inv2.IN | inv2.OUT -> sgm7.D2 -> vcd
                     ledrow.11<== avr.C3 ==> inv3.IN | inv3.OUT -> sgm7.D3 -> vcd
                     ''',
                      dict(
                          avr=avr,
                          sgm7=sgm7,
                          ledrow=ledrow,
                          inv0=inv[0],
                          inv1=inv[1],
                          inv2=inv[2],
                          inv3=inv[3],
                          ),
                      vcd=vcd,
                      )

#####################################################
# sgm7 game
def segments_func(digit_index):
```

```
    return (sgm7.digit_segments(digit_index), sgm7.reset_dirty(digit_index))
sgm7_game = Sgm7Game(segments_func=segments_func, disp_size=4)

##################################################
# compose game
dev = CompositeGame([
                  CompositeGame(
                          [
                           sgm7_game,
                           led_game_seg,
                           led_game_dig,
                           ],
                                align=1),
                  InfoGame(avr),
                  ])

scrshot_by_exit = [(dev, image_file)] if image_file else None

AvrSimMain(avr, dev, vcd, speed=speed, fps=fps, visible=visible, timeout=timeout,
           scrshot_by_exit=scrshot_by_exit).run_game()
```

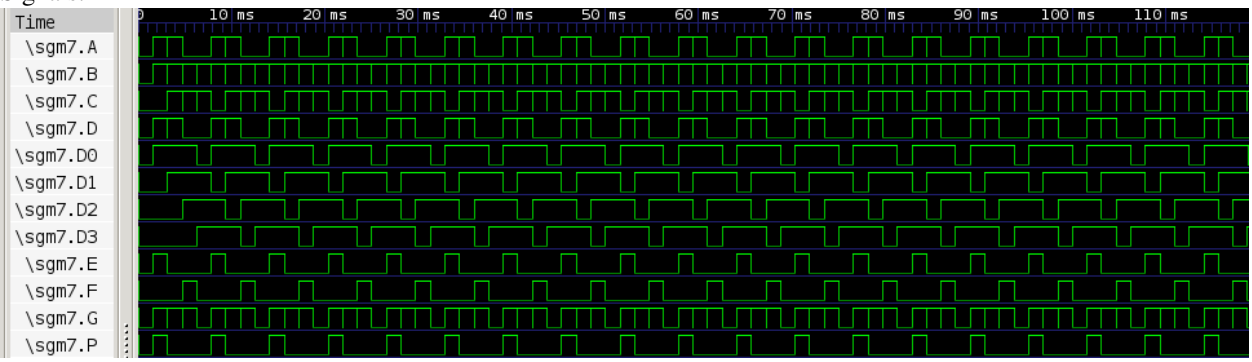
Starting program:

```
>>> from pysimavrgui.examples.sim.sgm7 import run_sim
>>> run_sim(vcdfile='docs/sgm7.vcd', speed=1, timeout=0.1, fps=50, visible=0, image_file='docs/sgm7.p
Loaded 680 .text
Loaded 6 .data
Starting atmega168 - flashend 3fff ramend 04ff e2end 01ff
atmega168 init
avr_timer_reconfigure-1 clock turned off
avr_timer_write_ocr-1 mode 0 UNSUPPORTED
avr_timer_reconfigure-1 clock turned off
avr_timer_configure-1 TOP 639.80Hz = 12503 cycles
avr_timer_configure-1 A 639.80Hz = 12503 cycles
```

GUI:


12.34
B0 B1 B2 B3 B4 B5 B6 B7
C0 C1 C2 C3
mcu=atmega168
f_cpu=8MHz
sgm7.elf (16:02:47)
prog: 686 bytes 4.2%
mem: 12 bytes 1.2%
vcc=5V avcc=5V
pc= 236
state=Running
cycle= 693273
mcu time= 86800 us
real time= 0 s
real speed= 0.000000x

Signals:


Time
\sgm7.A
\sgm7.B
\sgm7.C
\sgm7.D
\sgm7.D0
\sgm7.D1
\sgm7.D2
\sgm7.D3
\sgm7.E
\sgm7.F
\sgm7.G
\sgm7.P
10 ms
20 ms
30 ms
40 ms
50 ms
60 ms
70 ms
80 ms
90 ms
100 ms
110 ms

CHAPTER

FOUR

ARDUINO SIMULATOR

How to use it:

- start arduino software
- compile a sketch, the firmware will be saved in temporary directory
- start the arduino simulator example: `'python -m pysimavrgui.examples.sim.arduino'` The name of the sketch is displayed on the GUI.
- after recompiling in arduino select 'reload' on simulator GUI

4.1 LCD sketch

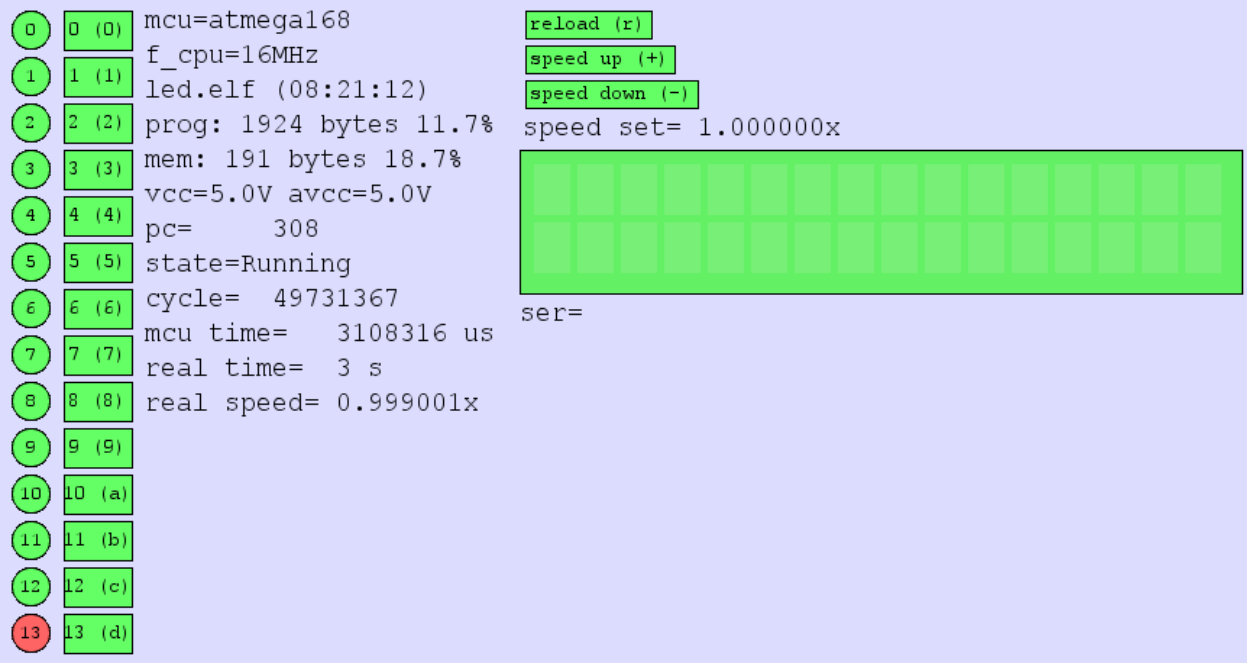
```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
}
```

```
$ python -m pysimavrgui.examples.sim.arduino -c pysimavrgui/examples/arduino/lcd.pde
```

4.2 LED sketch

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);

  digitalWrite(13, HIGH);   // set the LED on
}

void loop() {
}
```

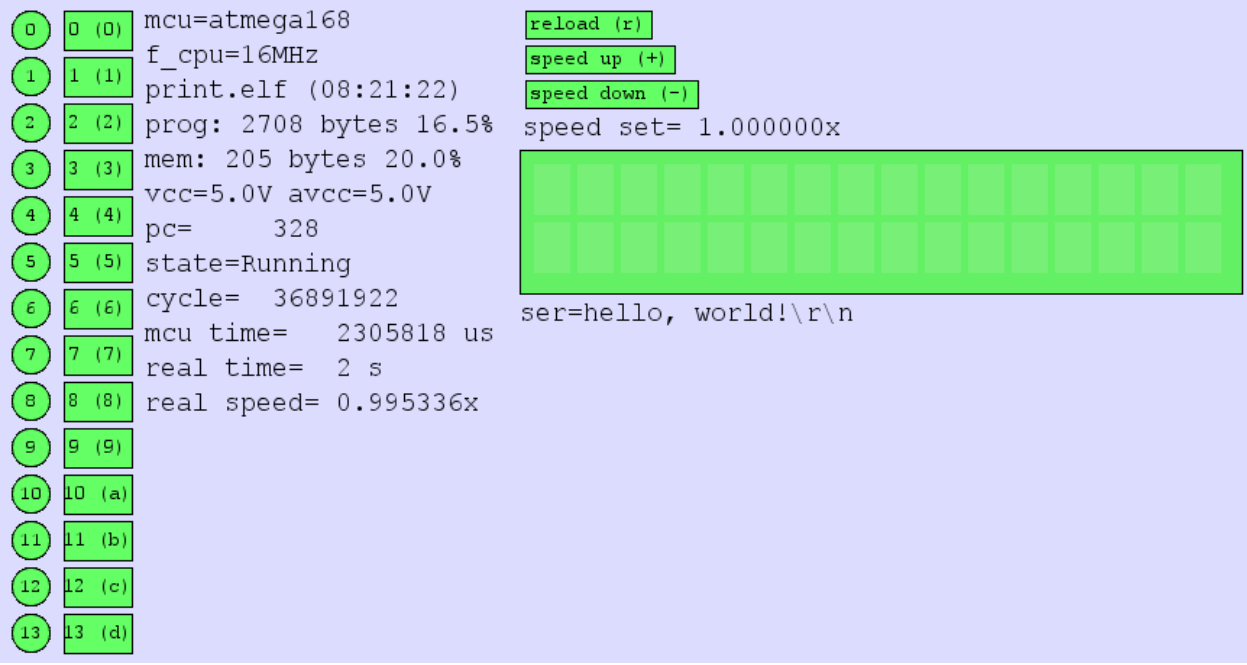
```
$ python -m pysimavrgui.examples.sim.arduino -c pysimavrgui/examples/arduino/led.pde
```



4.3 print sketch

```
void setup() {
    Serial.begin(9600);
    Serial.println("hello, world!");
}

void loop() {
}
```

```
$ python -m pysimavrgui.examples.sim.arduino -c pysimavrgui/examples/arduino/print.pde
```


0
0 (0)
1
1 (1)
2
2 (2)
3
3 (3)
4
4 (4)
5
5 (5)
6
6 (6)
7
7 (7)
8
8 (8)
9
9 (9)
10
10 (a)
11
11 (b)
12
12 (c)
13
13 (d)
mcu=atmega168
f_cpu=16MHz
print.elf (08:21:22)
prog: 2708 bytes 16.5%
mem: 205 bytes 20.0%
vcc=5.0V avcc=5.0V
pc= 328
state=Running
cycle= 36891922
mcu time= 2305818 us
real time= 2 s
real speed= 0.995336x
reload (r)
speed up (+)
speed down (-)
speed set= 1.000000x
ser=hello, world!\r\n

CHAPTER

FIVE

CLI HELP FOR EXAMPLES

5.1 Pygame GUI examples

```
$ python -m pysimavrgui.examples.gui.lcdgame_ex --help
usage: lcdgame_ex.py [-h] [--debug]

optional arguments:
  -h, --help  show this help message and exit
  --debug     set logging level to DEBUG

$ python -m pysimavrgui.examples.gui.ledrowgame_ex --help
usage: ledrowgame_ex.py [-h] [--debug]

optional arguments:
  -h, --help  show this help message and exit
  --debug     set logging level to DEBUG

$ python -m pysimavrgui.examples.gui.sgm7game_ex --help
usage: sgm7game_ex.py [-h] [--debug]

optional arguments:
  -h, --help  show this help message and exit
  --debug     set logging level to DEBUG

$ python -m pysimavrgui.examples.gui.textgame_ex --help
usage: textgame_ex.py [-h] [--debug]

optional arguments:
  -h, --help  show this help message and exit
  --debug     set logging level to DEBUG
```

5.2 Simulation examples

```
$ python -m pysimavrgui.examples.sim.ledramp --help
usage: ledramp.py [-h] [-v VCDFILE] [-s SPEED] [-f FPS] [-t TIMEOUT]
                  [--visible VISIBLE] [-i IMAGE_FILE] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  -v VCDFILE, --vcdfile VCDFILE
```

```
  -s SPEED, --speed SPEED
  -f FPS, --fps FPS
  -t TIMEOUT, --timeout TIMEOUT
  --visible VISIBLE
  -i IMAGE_FILE, --image-file IMAGE_FILE
  --debug               set logging level to DEBUG

$ python -m pysimavrgui.examples.sim.lcd --help
usage: lcd.py [-h] [-v VCDFILE] [-s SPEED] [-f FPS] [-t TIMEOUT]
              [--visible VISIBLE] [-i IMAGE_FILE] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  -v VCDFILE, --vcdfile VCDFILE
  -s SPEED, --speed SPEED
  -f FPS, --fps FPS
  -t TIMEOUT, --timeout TIMEOUT
  --visible VISIBLE
  -i IMAGE_FILE, --image-file IMAGE_FILE
  --debug               set logging level to DEBUG

$ python -m pysimavrgui.examples.sim.sgm7 --help
usage: sgm7.py [-h] [-v VCDFILE] [-s SPEED] [-f FPS] [-t TIMEOUT]
               [--visible VISIBLE] [-i IMAGE_FILE] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  -v VCDFILE, --vcdfile VCDFILE
  -s SPEED, --speed SPEED
  -f FPS, --fps FPS
  -t TIMEOUT, --timeout TIMEOUT
  --visible VISIBLE
  -i IMAGE_FILE, --image-file IMAGE_FILE
  --debug               set logging level to DEBUG

$ python -m pysimavrgui.examples.sim.arduino --help
usage: arduino.py [-h] [-e ELF] [-m MCU] [-f F_CPU] [-v VCDFILE] [-s SPEED]
                  [--fps FPS] [-t TIMEOUT] [--visible VISIBLE] [-i IMAGE_FILE]
                  [-r RATE] [-b BUTTONS_ENABLE] [--vcd-enable VCD_ENABLE]
                  [--spk-enable SPK_ENABLE] [-u UDP_ENABLE] [-a AVCC]
                  [--vcc VCC] [-c CODE] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  -e ELF, --elf ELF
  -m MCU, --mcu MCU
  -f F_CPU, --f-cpu F_CPU
  -v VCDFILE, --vcdfile VCDFILE
  -s SPEED, --speed SPEED
  --fps FPS
  -t TIMEOUT, --timeout TIMEOUT
  --visible VISIBLE
  -i IMAGE_FILE, --image-file IMAGE_FILE
  -r RATE, --rate RATE
  -b BUTTONS_ENABLE, --buttons-enable BUTTONS_ENABLE
  --vcd-enable VCD_ENABLE
  --spk-enable SPK_ENABLE
  -u UDP_ENABLE, --udp-enable UDP_ENABLE
```

```
-a AVCC, --avcc AVCC  AVcc in mV
--vcc VCC             Vcc in mV
-c CODE, --code CODE
--debug               set logging level to DEBUG
```

CHAPTER

SIX

API

class pysimavrgui.buttongame.**ButtonGame**(*hook=None*, *shortcut=None*, *label=''*, *size='auto'*, *display_shortcut=True*, *font_size=14*)

colors = {'text': (0, 0, 0), 'border': (0, 0, 0), 'transparent': (7, 7, 7), 'off': (100, 255, 100), 'on': (255, 100, 100)}

handleEvents(*event*)

size

surface

update()

class pysimavrgui.compgame.**CompositeGame**(*devs*, *align=0*, *size='auto'*, *gap=2*)

BG_COLOR = (220, 220, 255)

exit()

handleEvents(*event*)

size

surface

update()

class pysimavrgui.infogame.**InfoGame**(*avr*)

exit()

reload()

pysimavrgui.infogame.**format_freq**(*f*)

class pysimavrgui.lcdgame.**LcdGame**(*char_func*, *disp_size=(10, 2)*, *label=''*)

colors = {'text': (0, 0, 0), 'bgr': (100, 240, 100), 'font_bgr': (120, 240, 120), 'border': (0, 0, 0)}

load_fonts()

size

surface

update()

class pysimavrgui.ledgame.**LedGame**(*state_func*, *label=''*, *size=(30, 30)*)

colors = {'on': (255, 100, 100), 'off': (100, 255, 100), 'text': (0, 0, 0), 'border': (0, 0, 0), 'transparent': (7, 7, 7), 'pulse': (

on

pulse

size

state
(on/off,pulse)

surface

update()

class pysimavrgui.ledrowgame.**LedRowGame**(*state_func*, *disp_size=None*, *labels=None*, *align=0*, *size='auto'*)

class pysimavrgui.maingame.**MainGame**(*dev*, *pos=(0, 0)*, *fps=50*, *size='auto'*, *title='python-simavr'*, *visible=True*, *scrshot_by_exit=None*)

BG_COLOR = (100, 100, 180)

cb_exit()

cb_loop()

handleEvents()

run_game()

screenshot(*dev=None*, *img_file='screenshot.png'*)

terminate()

class pysimavrgui.sgm7game.**Sgm7Game**(*segments_func*, *disp_size=4*, *label=''*)

colors = {'on': (0, 0, 0), 'bgr': [210, 210, 210], 'off': [230, 230, 230], 'text': (0, 0, 0), 'border': (0, 0, 0), 'pulse': (222, 0, 0

draw_digit(*i*, *segments*, *color*)

size

surface

update()

class pysimavrgui.textgame.**TextGame**(*text_func*, *size=(30, 30)*, *font_size=19*)
multi line is not supported!

colors = {'text': (0, 0, 0)}

font

size

surface

text

update()

CHAPTER

SEVEN

DEVELOPMENT

7.1 Tools

1. setuptools
2. Paver
3. nose
4. ghp-import
5. pyflakes
6. pychecker
7. paved fork
8. Sphinx
9. sphinxcontrib-programscreenshot
10. sphinxcontrib-paverutils
11. `autorun` from sphinx-contrib (there is no simple method, you have to download/unpack/setup)

7.2 Install on ubuntu

```
sudo apt-get install python-setuptools
sudo apt-get install python-paver
sudo apt-get install python-nose
sudo easy_install ghp-import
sudo apt-get install pyflakes
sudo apt-get install pychecker
sudo easy_install https://github.com/ponty/paved/zipball/master
sudo apt-get install scrot
sudo apt-get install xvfb
sudo apt-get install xserver-xephyr
sudo apt-get install python-imaging
sudo apt-get install python-sphinx
sudo easy_install sphinxcontrib-programscreenshot
sudo easy_install sphinxcontrib-programoutput
sudo easy_install sphinxcontrib-paverutils
```

7.3 Tasks

Paver is used for task management, settings are saved in pavement.py. Sphinx is used to generate documentation.

print paver settings:

```
paver printoptions
```

clean generated files:

```
paver clean
```

generate documentation under *docs/_build/html*:

```
paver cog pdf html
```

upload documentation to github:

```
paver ghpages
```

run unit tests:

```
paver nose
#or
nosetests --verbose
```

check python code:

```
paver pyflakes
paver pychecker
```

generate python distribution:

```
paver sdist
```

upload python distribution to PyPI:

```
paver upload
```

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p

INDEX

T

U