Карпов В.Э.

Система выбора программы при загрузке Raspberry

07.11.2024

Назначение: определение режима запуска прикладных программ (приложений) при включении Raspberry.

Для этого автозапуск настраивается так, чтобы был запущен некоторый скрипт, результат работы которого определяет режим работы.

Основой стартового скрипта является программа выбора режима. Для работы программа использует аппаратные средства – специальный терминал.

1. Терминал

В состав терминала входят:

1. Экран (LCD или OLED SSD1306), подключенный по интерфейсу i2c.
2. Четырехкнопочная клавиатура.
3. Активная пищалка.
4. Световая индикация.

При использовании в качестве экране LCD, надо иметь клавиатуру из 4 нормально разомкнутых кнопок, подтянутых к «1». Кнопки клавиатуры OLED уже подтянуты.

2. Программа выбора режима psel

Назначение – определение режима работы (выбор номера программы). Выбор программы осуществляется четырьмя кнопками.

Для отображения используется либо LCD, либо OLED SSD1306, подключенных по интерфейсу i2c. Кроме того, программа отправляет управляющие сигналы на внешнюю пищалку и световой индикатор.

Программа помещает в стандартный выходной поток номер выбранной программы. По умолчанию результат равен 1.

Выбор устройства определяется парой переменных: USE_LCD и USE_OLED.

```
# Выбор устройства
USE_LCD = False
USE_OLED = True
```

Кнопки управления:

'^' – увеличить номер программы

'v' – уменьшить номер программы

'*' – режим (не используется)

'#' – Старт (выбор)

Если прошло 10 с. и ничего не нажато, то определяется выбор по умолчанию – программа №1.

Если клавиатура с подтянутыми кнопками не подключена (программа проверяет это), то будет выбор по умолчанию.

Пищалка и индикатор работаю синхронно.

Звуковая (световая) индикация:

- запуск программы – 1 сигнал
- ошибка инициализации LCD/OLDE – 3 сигнала
- нормальное завершение программы – 1 сигнал

3. Требуемые компоненты и пакеты

Требуются компоненты для работы с портами а также синтезатор речи (говорилка).

Порты GPIO

sudo apt-get update

sudo dpkg --configure -a

sudo apt-get install python3-rpi.gpio

sudo apt-get install python-rpi.gpio

Говорилка

RHVoice https://kiberblog.clan.su/blog/2015-08-13-282 https://windwheel.ru/?module=articles&c=news&b=1&a=51 sudo snap install rhvoice

Установка драйвера:

sudo apt-get install alsa-utils –y

sudo modprobe snd-bcm2835

Добавление драйвера в автозагрузку

Открываем файл автозагрузки:

sudo nano /etc/modules

Добавляем в конец файла следующую строку:

snd-bcm2835

4. Настройка автозагрузки

Стартовый скрипт выполняется в режиме суперпользователя. Отсюда много проблем, особенно со звуком.

1. Редактирование /etc/rc.local

Добавить строку

/home/ubuntu/ros/src/yy_ctl/startup/startup.sh

2. Установка громкости звука

Использовать утилиту alsamixer

5. Определение состояния сети

Windows:

ipconfig

ping

netstat

```
for /l %i in (1,1,255) do @ping 192.168.0.%i -w 1 -n 1|find /i "ttl="
```

Linux:

ifconfig

nmap

ping

6. Файлы

6.1. Файл rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

/home/ubuntu/ros/src/yy_ctl/startup/startup.sh

exit 0
```

6.2. Файл startup.sh

```
#!/bin/bash
#
# Автозапуск
# Задача - что-то сказать и определить номер программы.

HOMEDIR=/home/ubuntu/ros/src/yy_ctl/startup
LOGFILE=/home/ubuntu/log

# Добавляем запись в log-файл
echo "Statup at `date`" >> $LOGFILE

# Озвучиваем запуск скрипта
runuser - ubuntu -c "$HOMEDIR/say.sh Система загружена"
runuser - ubuntu -c "$HOMEDIR/say.sh Выбор текущей программы"

# Запускаем скрипт выбора программы.
# Реальная работа может быть произведена в нем, а может использоваться
результат ее работы.
# Программа помещает в стандартный выходной поток номер выбранной программы
(номер программы - 1, 2, 3, 4).
# Здесь результат помещается в NPROG.
NPROG=`python3 $HOMEDIR/psel.py`

# Протоколируем результат в log-файле
echo "prog=$NPROG">> $LOGFILE

# Озвучиваем номер программы
runuser - ubuntu -c "$HOMEDIR/say.sh Выбрана программа $NPROG"

#
# Далее делаем все, что необходимо, исходя из значения переменной NPROG
# ...
#

runuser - ubuntu -c "$HOMEDIR/say.sh Система готова к работе"
```

6.3. Файл say.sh

```
#!/bin/bash

if [ $# -lt 1 ];
then
        echo "Usage is: $0 text"
        exit
fi

#echo $1 | spd-say -o rhvoice -l ru  -e -t female1

echo $1 $2 $3 $4 $5 $6 $7 $8 $9 | RHVoice-client -s irina+CLB | aplay
```

6.4. Файл psel.py

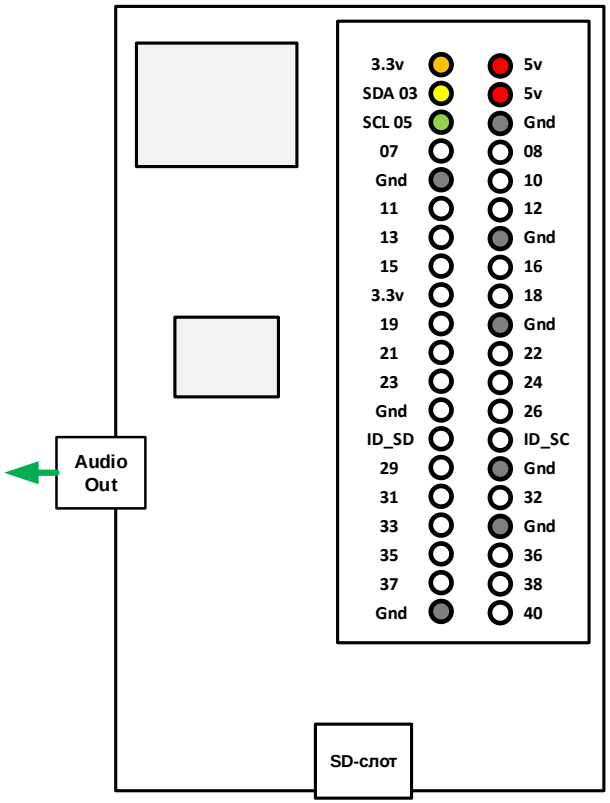
```
#!/usr/bin/env python3
# coding: utf-8
'''
  Определение режима работы (выбор номера программы)
  Используется либо LCD, либо OLED SSD1306
  Все кнопки должны быть подтянуты к 1 (кнопки OLED уже подтянуты)
  -- 2 кнопки выбора номера программы (для OLED: '^', 'v')
  -- 1 кнопка "Старт" (для OLED: '#')

  Если прошло 10 с. и ничего не нажато, то определяется выбор по умолчанию - №1
  Если клавиатура с подтянутыми кнопками не подключена (программа проверяет это), то будет выбор по умолчанию

  Пищалка и индикатор работаю синхронно

  Звуковая индикация:
  -- запуск программы -- 1 сигнал
  -- ошибка инициализации LCD/OLDE -- 3 сигнала
  -- нормальное завершение программы -- 1 сигнал
  V 1.02
  04.08.2024
  LP 07.11.2024
'''

import RPi.GPIO as gpio
import os, sys, time

sys.path.insert(0, os.path.dirname(os.path.realpath(__file__))+"/lib")

import lcddriver
import oledssd1306

# Выбор устройства
USE_LCD = False
USE_OLED = True

gpio.setwarnings(False)
gpio.cleanup()

'''
  GPIO.BOARD — нумерация портов на плате по порядку
  GPIO.BCM — это более низкий уровень и обращается напрямую к номерам каналов на процессоре
```

```
    gpio.setmode(gpio.BCM)
    gpio.setmode(gpio.BOARD)
'''

# Было так: pin_mode = gpio.BOARD
pin_mode = gpio.BCM

gpio.setmode(pin_mode)

pin_mode = gpio.getmode() # -> GPIO.BOARD, GPIO.BCM

'''
  PIN_INP[0] -- Up   ('^' номер программы++)
  PIN_INP[1] -- Down ('v' номер программы--)
  PIN_INP[2] -- Mode ('*' не используется)
  PIN_INP[3] -- Start ('#')
'''

if pin_mode==gpio.BOARD:
    PIN_LED = 11  # Индикатор
    PIN_BEEP = 13 # Пищалка
    PIN_INP = [15, 16, 18, 19]
else:
    PIN_LED = 17  # Индикатор
    PIN_BEEP = 27 # Пищалка
    PIN_INP = [22, 23, 24, 10]

def Beep(T):
    gpio.output(PIN_LED, gpio.HIGH)
    gpio.output(PIN_BEEP, gpio.HIGH)
    time.sleep(T)
    gpio.output(PIN_LED, gpio.LOW)
    gpio.output(PIN_BEEP, gpio.LOW)
    time.sleep(T)

def ReadButton():
    for i in range(len(PIN_INP)):
        if gpio.input(PIN_INP[i]) == gpio.LOW:
            while gpio.input(PIN_INP[i]) == gpio.LOW:
                time.sleep(0.1)
            return i
    return -1

if __name__ == '__main__':
    gpio.setup([PIN_LED, PIN_BEEP], gpio.OUT, initial=gpio.LOW)
    gpio.setup(PIN_INP, gpio.IN, pull_up_down=gpio.PUD_UP)

    T = 0.5
    Beep(T)

    PROGR_NUM = 4 # Максимальное количество программ
    curr_prog = 1

    #print(gpio.VERSION)
    #print(gpio.RPI_INFO)
    try:
        if USE_LCD:
            screen = lcddriver.lcd(0x27)
        if USE_OLED:
            screen = oledssd1306.lcd(0x3c)

    except:
        Beep(T)
```

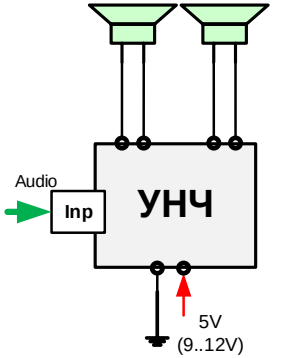
```
        Beep(T)
        Beep(T)
        gpio.cleanup()
        print(curr_prog)
        sys.exit(1)

    screen.display_string = screen.lcd_display_string
    screen.clear = screen.lcd_clear

    screen.lcd_clear()

    screen.lcd_display_string(f"# = {curr_prog}", 2)

    exit_stat_str = "???"

    pred_t = -1
    WAIT_TIME = 10 # Время ожижания ввода команды
    TCNT = WAIT_TIME

    #
    # Проверка клавиатуры
    #
    n = sum([gpio.input(PIN_INP[i]) for i in range(len(PIN_INP))])
    kbd_not_found = (n!=len(PIN_INP))

    while True:
        if kbd_not_found:
            exit_stat_str = "Keyboard not found"
            break
        #
        # Разбираемся со временем
        #
        ct = int(time.time())
        if pred_t!=ct:
            pred_t=ct
            screen.lcd_display_string(f"P.SELECTOR | cnt: {TCNT} ", 1)
            TCNT-=1
            if TCNT<0:
                exit_stat_str = "Waiting time expired"
                break

        #
        # Считываем кнопки
        #
        n = ReadButton()
        if n<0: continue
        if n==0: # Up
            curr_prog += 1
            if(curr_prog>PROGR_NUM): curr_prog = 1
        if n==1: # Down
            curr_prog -= 1
            if(curr_prog<1): curr_prog = PROGR_NUM
        if n==2: # Mode
            pass
        if n==3: # Start Button
            exit_stat_str = f"Ok"
            break
        screen.lcd_display_string(f"# = {curr_prog}", 2)
        TCNT = WAIT_TIME


    screen.lcd_display_string(exit_stat_str, 3)
    #
```

```
    # Выполняем программу curr_prog
    #
    screen.lcd_display_string(f"Exec prog {curr_prog}...", 4)
    print(curr_prog)
    '''
    if curr_prog==1:
        ....
    if curr_prog==2:
        ....
    if curr_prog==3:
        ....
    if curr_prog==2:
        ....
    '''
    screen.lcd_display_string(f"Exec prog {curr_prog}... Done", 4)

    Beep(T)

    gpio.cleanup()
```

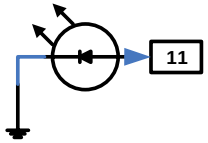
Терминал RaspPi



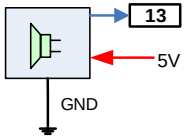
Обозначение - GPIO.BOARD



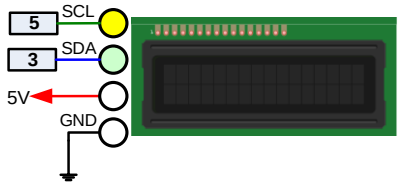
Индикатор



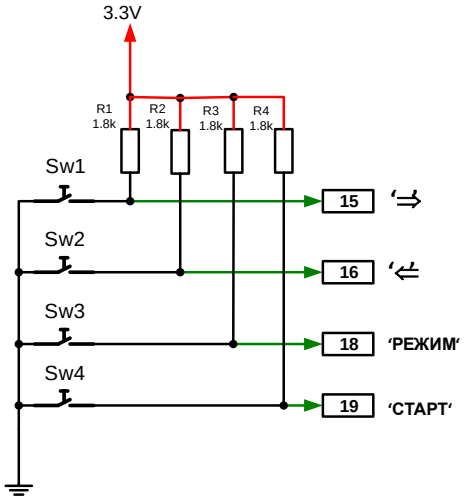
Пищалка



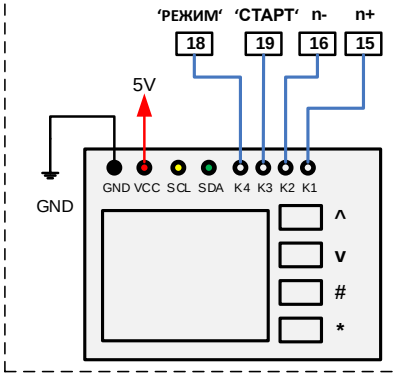
LCD + Самодельная клавиатура



Кнопки выбора программы



OLED SSD1306



07.11.2024