Использование портов ввода-вывода GPIO микрокомпьютера Raspberry Pi

А. ШИТОВ, г. Амстердам, Нидерланды

В журнале "Радио" была опубликована статья об одноплатных микрокомпьютерах Raspberry Pi (Кутепов И. Микрокомпьютер Raspberry Pi. — Радио, 2014, № 1, с. 17—22). Автор предлагаемой вниманию читателей статьи рассказывает об устройстве портов ввода-вывода и о том, как их программировать для работы с внешними устройствами.

Со времени последней публикации в журнале модельный ряд Raspberry Pi заметно обновился. Сегодня доступны три версии устройств: Raspberry Pi, Raspberry Pi 2 и Raspberry Pi 3, каждая из которых представлена в нескольких вариантах с небольшими отличиями.

Внешний вид моделей первого, второго и третьего поколений показан на **рис. 1**—**рис. 3**.

Помимо основного ряда, выпускаются модели Raspbery Pi Zero и Raspberry Pi Zero W, а также три варианта платы Compute Module, — все они достойны отдельной статьи.

Основные параметры Raspberry Pi различных версий перечислены в **табл. 1** [1, 2]. Как видно из таблицы, новые модели существенно превосходят первые и по мощности ARM-процессора, и по числу USB-портов, и по возможностям беспроводного подключения к сети. Цена при этом практически не изменилась, поэтому имеет смысл приобретать последнюю доступную модель.

Произошли отличия и в числе портов ввода-вывода общего назначения GPIO (General Purpose Input-Output). В первых моделях основной разъём GPIO имел 26 выводов, а начиная с Raspberry Pi 2, микрокомпьютеры оснащены 40-контактным разъёмом. Разработчики постарались сделать эти разъёмы максимально совместимыми: а именно, 26 контактов моделей Raspberry Pi A и B совпадают по назначению с частью 40-выводного разъёма; совпадающие части выделены на **рис. 4**. Это не совсем так для моделей A и B первой ревизии — у них на месте GPIO 2 и GPIO 3 находятся

Рис. 1

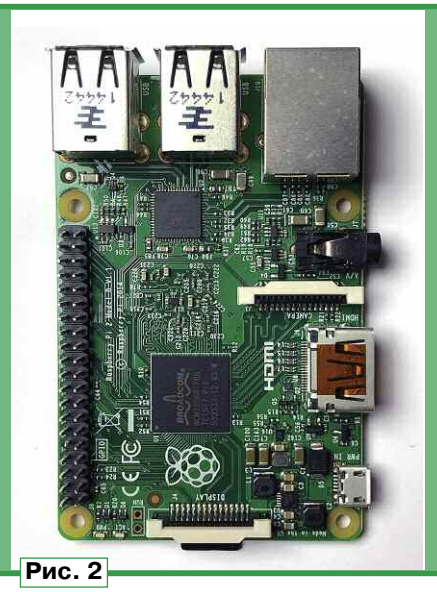

Рис. 2

Рис. 3

Таблица 1

Версия и модель Raspberry Pi	Дата выхода	Тип "системы на кристалле" (SoC)	Частота процессора, МГц	Число ядер	Объём ОЗУ, МБ	Число разъёмов USB	Наличие порта Ethernet	Наличие и тип Wi-Fi	Наличие и тип Bluetooth	Потребляемый ток от источника питания 5 В, мА	Тип карты памяти	Число выводов на разъёме GPIO
A	февраль 2013	BCM2835	700	1	256	1	нет	нет	нет	300	SD	26
A+	ноябрь 2014	BCM2835	700	1	512	1	нет	нет	нет	200	MicroSD	26
B	март 2012	BCM2835	700	1	512	2	есть	нет	нет	700	SD	26
B+	июнь 2014	BCM2835	700	1	512	4	есть	нет	нет	200—350	MicroSD	40
2 B	февраль 2015	BCM2836, BCM2837 (версия 1.2)	900	4	1024	4	есть	нет	нет	200—850	MicroSD	40
3 B	февраль 2016	BCM2837	1200	4	1024	4	есть	802.11n	2.0/4.1	300—1300	MicroSD	40
3 B+	март 2018	BCM2837B0	1400	4	1024	4	есть	802.11ac, 2.4 ГГц и 5 ГГц	2.0/4.1/4.2 LS BLE	450—1200	MicroSD	40

GPIO 0 и GPIO 1 — но, скорее всего, читатель не столкнётся с этими моделями Raspberry Pi.

В основе разных моделей лежат "системы на кристалле" (SoC) BCM2835, BCM2836 и BCM2837. Сами по себе они имеют 54 порта ввода-вывода, однако в Raspberry Pi используется только часть из них, причём на 40-контактном разъёме несколько выводов подключены к общему проводу либо к линиям питания +3,3 и +5 В, а на долю GPIO приходится лишь 28 контактов. Программным путём каждому выводу GPIO может быть назначено от двух до шести альтернативных функций (например, работа в последовательном интерфейсе).

Рис. 4

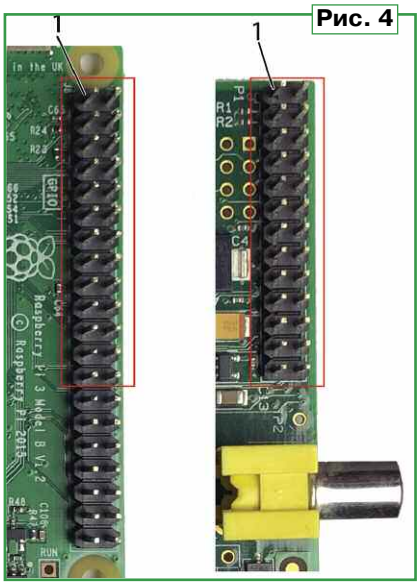

Таблица 2

Основная функция	Номер контакта		Основная функция
+3,3 В	1	2	+5 В
GPIO 2	3	4	+5 В
GPIO 3	5	6	GND
GPIO 4	7	8	GPIO 14
GND	9	10	GPIO 15
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
+3,3 В	17	18	GPIO 24
GPIO 10	19	20	GND
GPIO 9	21	22	GPIO 25
GPIO 11	23	24	GPIO 8
GND	25	26	GPIO 7
GPIO 0	27	28	GPIO 1
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

Назначение выводов GPIO 40-контактного разъёма показано в **табл. 2**. При работе с устройством следует иметь в виду, что номера GPIO, во-первых, не совпадают с номерами физических выводов, и, во-вторых, в 40-контактном варианте выведены все GPIO с 0-го по 27-й. В некоторых источниках вместо GPIO используют сокращение BCM (Broadcom Pin Number, по названию производителя процессора).

Внутреннее устройство

Рассмотрим упрощённую блок-схему [3] внутреннего устройства выводов GPIO (**рис. 5**). Состояние, направление работы и назначение выводов контролируются несколькими 32-разрядными внутренними регистрами, названия которых начинаются с букв GP. В зависимости от назначения, управляющие регистры работают в режимах чтения, записи или чтения и записи.

В этой статье мы рассмотрим работу GPIO только в режиме основной функции (двоичных входов и выходов). Управляющие узлы каждого контакта содержат две части — обслуживающие, соответственно, режимы вывода и ввода.

После включения питания выводы GPIO устанавливаются в режим чтения. Перевод в другие режимы происходит при записи соответствующих значений в регистры GPFSEL0—GPFSEL5 (GPIO Function Select registers).

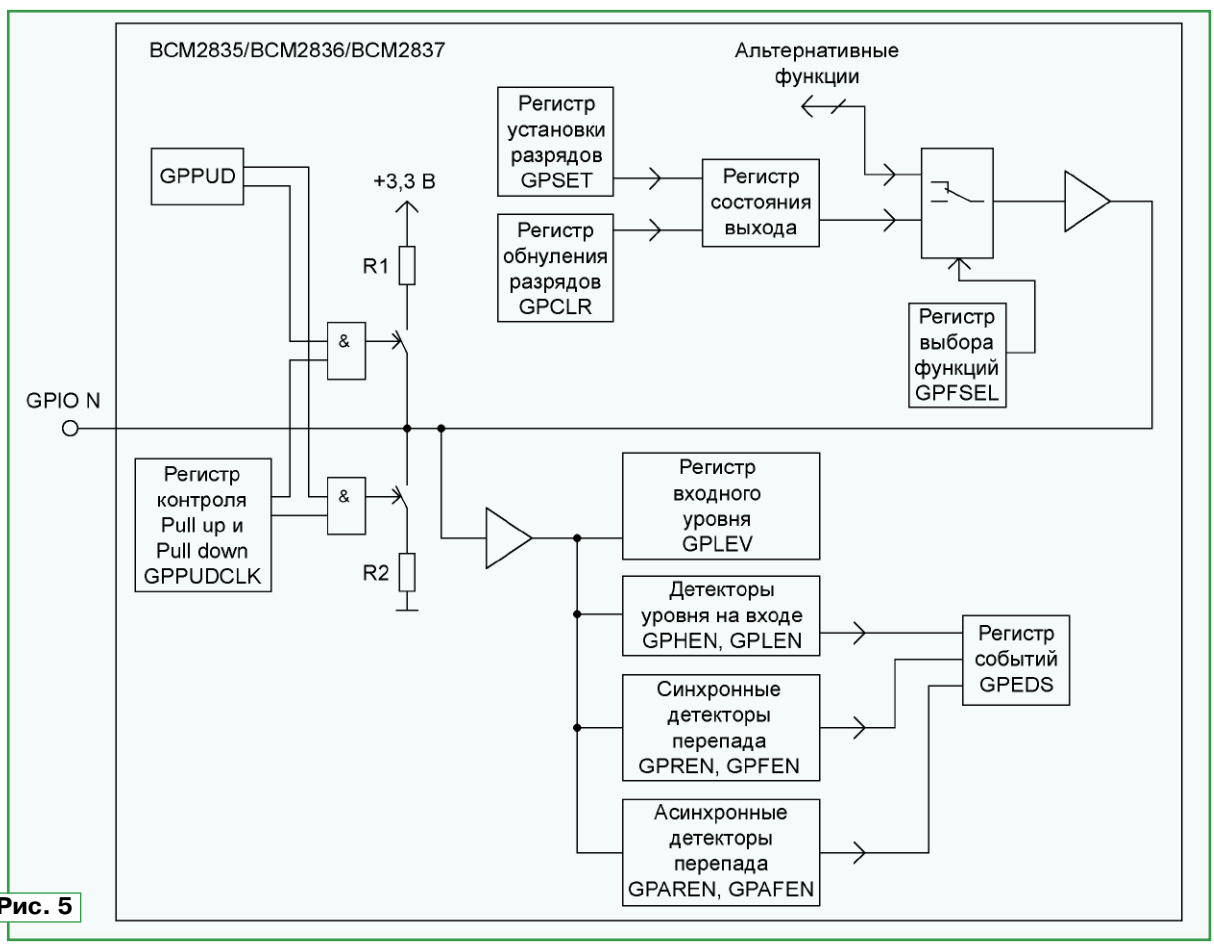

Рис. 5

Регистры GPSET0, GPSET1 (GPIO Pin Output Set), GPCLR0 и GPCLR1 (GPIO Pin Output Clear) отвечают за установку и обнуление разрядов на выводах, работающих на выход. Поскольку в Raspberry Pi используется только часть из доступных 54 выводов GPIO, мы ограничимся работой с регистрами GPSET0 и GPCLR0. Разряды этих регистров соответствуют первому банку GPIO с номерами от 0-го до 31-го, из которых доступны только первые 28. Вторая пара регистров — GPSET1 и GPCLR1 — ответственна за второй банк GPIO с 32-го по 54-й (старшие десять разрядов этих регистров не используются).

Узлы, ответственные за режим ввода, несколько сложнее. Во-первых, на вход можно подключить один из двух подтягивающих резисторов — за это отвечают регистры GPPUD (GPIO Pull up/Pull down), GPPUDCLK0 и GPPUDCLK1 (GPIO Pull up/Pull down Clock). Обратите внимание, что между выводами GPIO 2 и GPIO 3 и линией питания +3,3 В постоянно подключены подтягивающие резисторы сопротивлением 1,8 кОм. Значение в регистре GPPUD разрешает манипуляцию с одним из двух резисторов, соединённым либо с линией питания, либо с общим проводом. Единичные разряды в регистрах GPPUDCLKn выбирают соответствующие выводы GPIO. Здесь индекс n в названии регистра, как и в остальных случаях далее, принимает значения 0 или 1 — для работы с Raspberry Pi достаточно одного регистра GPPUDCLK0.

Получать информацию о состоянии GPIO, работающего в режиме ввода, можно несколькими способами. Самый простой — прочитать регистр GPLEVn (GPIO Pin Level). Разряды, которые соответствуют GPIO, находящимся в режиме ввода, будут содержать текущее значение на входе: младший разряд соответствует GPIO 0, второй разряд — GPIO 1 и т. д.

Ещё несколько регистров отвечают за менее тривиальные варианты считывания данных. Регистры событий GPEDSn (GPIO Event Detect Status) получают единичные значения, когда на входах GPIO происходит одно из событий, разрешённое разрядами в перечисленных ниже регистрах. Интересно, что событие будет оставаться активным до тех пор, пока вы самостоятельно не сбросите его, записав единицу в нужный разряд GPEDSn.

При установленных значениях разрядов в регистрах GPHENn (GPIO High Detect Enable) и GPLENn (GPIO Low Detect Enable) событие происходит при наличии высокого или низкого уровня на входе.

Регистры GPRENn (GPIO Rising Edge Detect Enable) и GPFENn (GPIO Falling Edge Detect Enable) переключают соответствующие разряды, когда на входе происходит перепад с нулевого уровня на единичный или наоборот. Эти события называют синхронными, поскольку они происходят только в том случае, если в течение трёх тактов системных часов зарегистрированы состояния 011 (перепад с 0 на 1) или 100 (перепад с 1 на 0).

Аналогичная пара регистров GPARENn и GPAFENn (A означает Asynchronous — асинхронные) создаёт событие сразу же без дополнительных проверок, что позволяет отслеживать более короткие импульсы на входе.

В том случае, когда для данного входа GPIO установлены разряды более чем в одном упомянутом выше регистре, событие в регистре GPEDSn регистрируется при наступлении любой из разрешённых ситуаций. Например, если записать значения 0x01 в регистры GPREN0 и GPFEN0, то событие для входа GPIO 0 будет регистрироваться и по фронту, и по спаду уровня на нём.

C-библиотека libbcm2835

Далее мы будем демонстрировать работу устройства, используя библиотеку libbcm2835 [4]. Она написана на языке C и может быть непосредственно использована в проектах на C и C++. Разумеется, можно работать с библиотеками на языках более высокого уровня, однако выбор низкоуровневого подхода даёт возможность продемонстрировать максимально эффективный способ работы с GPIO и показать его логику на уровне внутренних регистров.

Установка библиотеки libbcm2835 не вызывает сложностей (здесь и далее знак ↵, выделенный красным цветом, в конце строки обозначает перенос. Это сделано исключительно для удобства вёрстки статьи. Следующую строку следует набрать в предыдущей строке без знака ↵):

```
wget http://www.airspayce.com/↵
mikem/bcm2835/bcm2835-1.55.tar.gz
tar xzf bcm2835-1.55.tar.gz
cd bcm2835-1.55/
./configure
make
sudo make install
```

Далее приведены лишь ключевые фрагменты программ, полные исходные коды доступны в репозитории на GitHub [5].

Для компилирования и сборки программ необходимо подключить скомпилированную библиотеку, например, так:

```
$ g++ 2-level.cpp -l bcm2835
```

Прежде чем начать любые манипуляции, следует проинициализировать библиотеку и убедиться, что операция прошла успешно:

```
if (!bcm2835_init()) {
    return 1;
}
```

Режим вывода

Каждый вывод GPIO может работать как на вход, так и на выход. Рассмотрим сначала работу в режиме выхода — за него отвечают блоки в верхней части блок-схемы.

Регистры выбора режима GPFSELn доступны для прямой записи, однако в данном случае проще воспользоваться готовыми функциями библиотеки libbcm2835. В частности, для перевода контакта в режим вывода вызовите функцию bcm2835_gpio_fsel:

```
const int pin = 21;
bcm2835_gpio_fsel(pin, BCM2835↵
_GPIO_FSEL_OUTP);
```

В этом примере номер 21 соответствует выводу GPIO 21, подключённому к контакту 40 разъёма Raspberry Pi. Выбор этого контакта удобен тем, что он расположен рядом с контактом, соединённым с общим проводом — и удаётся подключить светодиод (последовательно с токоограничивающим резистором) небольшим разъёмом с двумя контактами (**рис. 6**) на краю рейки GPIO. Разумеется, вместо константы 21 может быть любое число, которое соответствует одному из доступных GPIO.

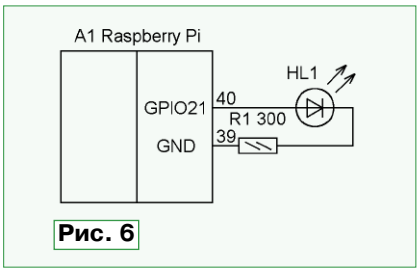

Рис. 6

Теперь посмотрим, как осуществлять вывод данных через порты GPIO. Самый простой и популярный способ — установить один из выходов в требуемое состояние. Этот способ был рассмотрен и в статье [6]. Приведём аналогичную программу для мигания светодиодом, но используя низкоуровневую библиотеку (полный код — в файле 1-blink.cpp [5]):

```
while(1) {
    bcm2835_gpio_write(pin, 1);
    bcm2835_delay(500);
    bcm2835_gpio_write(pin, 0);
    bcm2835_delay(500);
}
```

Вместо явной передачи устанавливаемого значения 1 или 0 можно воспользоваться парой функций bcm2835_gpio_set и bcm2835_gpio_clr (см. программу 4-beg-ogn.cpp). Этот способ прост, однако когда вы одновременно хотите изменить состояние нескольких выходов, вызывать несколько команд, меняющих уровень только одного выхода, неэффективно. Каждый такой вызов будет записывать новое значение в один и тот же управляющий регистр. Для решения задачи можно воспользоваться одним из двух вариантов.

Первый — вызов функции bcm2835_gpio_write_multi. Она принимает два параметра — битовую маску, которая выбирает выходы, в которых вы устанавливаете новые значения, и само новое значение — 1 или 0, например:

```
int mask = 1 << pin;
bcm2835_gpio_write_multi(mask, 1);
```

Полный код работающей программы, использующей этот подход, находится в файле 6-using-multi.cpp.

Второй вариант — произвести запись непосредственно в регистры управления состоянием GPIO. Здесь нужно иметь в виду, что для установки

на выходе высокого уровня записывают единицу в соответствующий разряд регистра GPSET0, а для установки низкого — единицы записывают в регистр GPCLR0. Обратите внимание, что для установки на выходе низкого уровня следует записать в разряды, соответствующие нужным GPIO, единицы (а не нули).

Для записи в регистры установки и сброса воспользуйтесь функцией bcm2835_peri_write. Операция потребует несложных вычислений, чтобы получить физический адрес регистра в памяти:

```
uint32_t* gpioBASE = bcm2835↵
_regbase(BCM2835_REGBASE_GPIO);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPSET0 / 4, 1 << pin);
```

Подробные примеры — в файлах 5-blink-peri-write.cpp, 6-beg-ogn-peri-write.cpp и 7-blink-n.cpp.

Режим ввода

Теперь рассмотрим работу GPIO в режиме чтения. При этом возможно активировать один из внутренних подтягивающих резисторов. Это очень удобно при подключении кнопок на замыкание — в этом случае не потребуется дополнительный внешний резистор.

Простейший случай — кнопка или выключатель, подключённый непосредственно между одним из GPIO и общим проводом, как показано на **рис. 7**. Поскольку контакт замыкается на общий

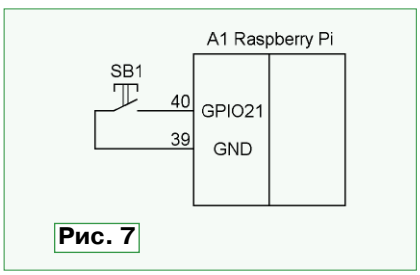

Рис. 7

провод, требуется подключить подтягивающий резистор к источнику питания:

```
bcm2835_gpio_fsel(pin, ↵
BCM2835_GPIO_FSEL_INPT);
bcm2835_gpio_set_pud(pin, ↵
BCM2835_GPIO_PUD_UP);
```

Далее читаем состояние входа (полная программа — в файле 2-level.cpp):

```
int lev = bcm2835_gpio_lev(pin);
```

Аналогично тому, как мы одновременно устанавливали несколько выходов GPIO, можно в одно действие получить состояние всех GPIO, переведённых в режим чтения. Всё, что нужно, — прочитать регистр GPLEV0:

```
uint32_t* gpioBASE = bcm2835↵
_regbase(BCM2835_REGBASE_GPIO);
uint32_t reg1 = bcm2835_peri_read↵
(gpioBASE + BCM2835_GPLEV0 / 4);
```

В переменной reg1 окажется 32-разрядное число, каждый разряд которого будет отражать текущее состояние соответствующего входа GPIO. Иллюстрация такого подхода — в программе 3-read-reg.cpp.

Анализ событий

Настало время обратиться к регистру событий GPEDS. Как было отмечено выше, этот регистр собирает информацию о событиях, разрешённых единичными разрядами в регистрах GPHEN, GPLEN, GPREN, GPFEN, GPAREN и GPAFEN. С программной точки зрения все они работают похоже. После регистрации и обработки события его необходимо очистить, записав единицу в нужный разряд GPEDS.

Во время экспериментирования не следует забывать о том, что ранее установленные значения в разрядах регистров могут генерировать нежелательные события. Поэтому при отладке включайте в программу подобную последовательность, чтобы обнулить все регистры:

```
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPHEN0 / 4, 0);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPLEN0 / 4, 0);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPREN0 / 4, 0);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPFEN0 / 4, 0);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPAREN0 / 4, 0);
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPAFEN0 / 4, 0);
```

Поведение, определённое регистрами GPREN и GPFEN, наиболее очевидно — событие фиксируется в момент фронта или спада входного сигнала.

Вернёмся к схеме на рис. 7 и подготовим регистрацию событий по спаду (т. е. при нажатии кнопки):

```
int pin = 21;
// . . .
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPFEN0 / 4, 1 << pin);
```

Как только кнопка будет нажата, в 21-м разряде регистра GPEDS будет записана единица:

```
int x = bcm2835_peri_read↵
(gpioBASE + BCM2835_GPEDS0 / 4);
```

Работа с событиями позволяет обнаружить нажатие на кнопку даже в том случае, если эта строка кода будет выполнена после отпускания кнопки. Как только мы вновь готовы к отслеживанию состояния кнопки, обнулим событие:

```
bcm2835_peri_write(gpioBASE + ↵
BCM2835_GPEDS0 / 4, 1 << pin);
```

Если в коде сделать замену и указать регистр GPFEN0, получится обработчик по спаду (он должен сработать при отпускании кнопки). Работающие примеры находятся в файлах 8-edge1.cpp и 9-timeout.cpp. После запуска программы выяснится, что событие возникает и при нажатии, и при отпускании кнопки независимо от выбора регистра. Разумеется, здесь дело в дребезге контактов, который легко исправить добавлением конденсатора параллельно кнопке — совместно с внутренним подтягивающим резистором он образует RC-фильтр (**рис. 8**). Дребезг контактов можно фильтровать и программно, выполняя два-три опроса состояния входа через 50...100 мс.

Регистры асинхронных событий GPAREN и GPAFEN работают аналогично только что рассмотренным GPREN и GPFEN, за тем исключением, что регистрируемые асинхронные события могут быть короче синхронных.

Посмотрим, что произойдёт, если задействовать регистры уровня GPHEN и GPLEN. В отличие от регистра GPLEV, который всегда показывает текущий уровень на входе, эти два регистра формируют события (то есть устанавливают единицы в разрядах GPEDS), если на входе присутствует высокий или низкий уровень. После этого событие остаётся зарегистрированным до тех пор, пока его не обнулят.

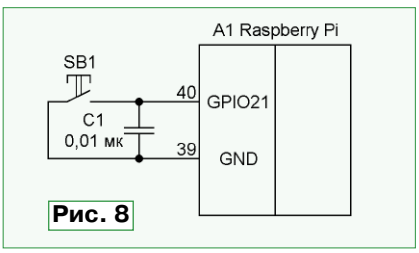

Рис. 8

В программе 9-timeout-by-lev.cpp показан пример обработки подобных событий. После нажатия на кнопку загорается светодиод, подключённый к выходу GPIO 14. Независимо от длительности нажатия на кнопку светодиод гаснет через одну секунду (одновременно с обнулением события). Однако если кнопка продолжает оставаться нажатой, светодиод погаснет, но тут же загорится, поскольку нажатая кнопка создаст новое событие. Благодаря программной задержке в 200 мс переключения светодиода заметны невооружённым глазом:

```
bcm2835_delay(200);
```

Все рассмотренные выше манипуляции с регистрами наглядны и унифицированы, однако можно воспользоваться и более специфичными функциями, которые предоставляет библиотека libbcm2835, например, bcm2835_gpio_eds_multi. Полный список доступен на странице [7].

Наконец, события, возникающие на входах GPIO, могут генерировать прерывания. Это полностью программная задача, решение которой зависит от установленной операционной системы. Один из примеров реализации прерывания можно найти в [8].

Сопряжение входов

Существенным недостатком портов GPIO Raspberry Pi (в отличие, например, от Arduino) является отсутствие какой-либо защиты на входах. Напряжение меньше 0 или больше 3,3 В, поданное на вход, мгновенно выводит этот GPIO из строя. Поэтому желательно поста-

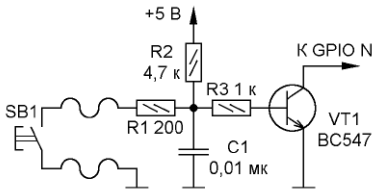

Рис. 9

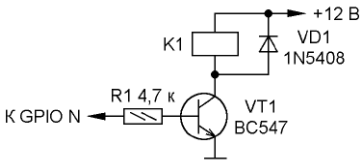

Рис. 11

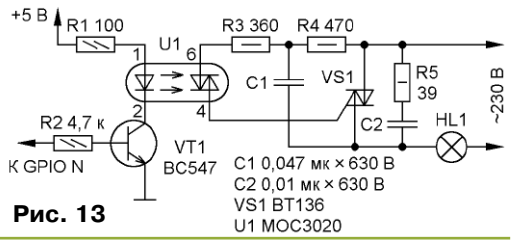

Рис. 13

вить на вход простейший сопрягающий узел на одном транзисторе (**рис. 9**). Транзистор также поможет обеспечить низкое входное сопротивление, что

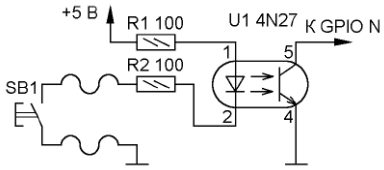

Рис. 10

благотворно сказывается при работе с длинными линиями, ведущими к датчику (кнопке, геркону или выключателю). Дополнительная RC-цепь на входе помогает отсекать импульсы, вызванные наводками в длинных проводах.

Напряжение питания +5 В допустимо взять либо непосредственно от источника питания, либо с контакта 2 или 4 GPIO.

Альтернативный вариант — применить оптрон (**рис. 10**). При необходимости цепи питания можно полностью развязать.

Сопряжение выходов

Выходы GPIO в обычном режиме находятся в одном из двух состояний — высокого или низкого уровня. Однако суммарный потребляемый ток со всех GPIO не должен превышать 50 мА. В простейших задачах достаточно подключить светодиод напря-

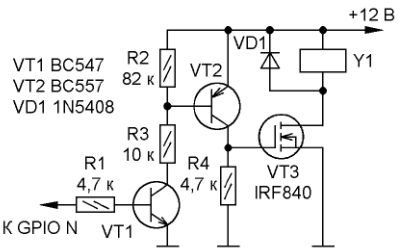

Рис. 12

мую к выводам (см. рис. 6), однако в большинстве практических задач желательно предусмотреть хотя бы простой усилитель на транзисторе (**рис. 11**), а для мощных нагрузок — ключ на мощном полевом транзисторе (**рис. 12**).

При подключении нагрузки к сети 230 В необходимо выполнить гальваническую развязку, например, с помощью тиристорного оптрона, как показано на **рис. 13**. Устройство, собранное по этой схеме, может работать с обычными лампами накаливания. Обратите внимание, что момент открывания симистора VS1 не синхронизирован с переходом сетевого напряжения через ноль, поэтому не рекомендуется использовать такой ключ со светодиодными лампами.

ЛИТЕРАТУРА

1. Raspberry Pi. — URL: **https://ru.wikipedia.org/wiki/Raspberry_Pi** (01.06.18).

2. Raspberry Pi Comparison Table. — URL: **https://www.modmypi.com/blog/raspberry-pi-comparison-table** (01.06.18).

3. BCM2835 ARM Peripherals. — URL: **https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf**, с. 89 (01.06.18).

4. C library for Broadcom BCM 2835 as used in Raspberry Pi. — URL: **http://www.airspayce.com/mikem/bcm2835/index.html** (01.06.18).

5. Исходные коды к статье. — URL: **https://github.com/ash/gpio-play** (01.06.18).

6. **Кутепов И.** Микрокомпьютер Raspberry Pi. — Радио, 2014, № 1, с. 17—22.

7. GPIO register access. — URL: **http://www.airspayce.com/mikem/bcm2835/group__gpio.html** (01.06.18).

8. Raspberry Pi And The IoT In C — Input And Interrupts. — URL: **https://www.iot-programmer.com/index.php/books/22-raspberry-pi-and-the-iot-in-c/chapters-raspberry-pi-and-the-iot-in-c/55-raspberry-pi-and-the-iot-in-c-input-and-interrupts?showall=&start=3** (01.06.18).