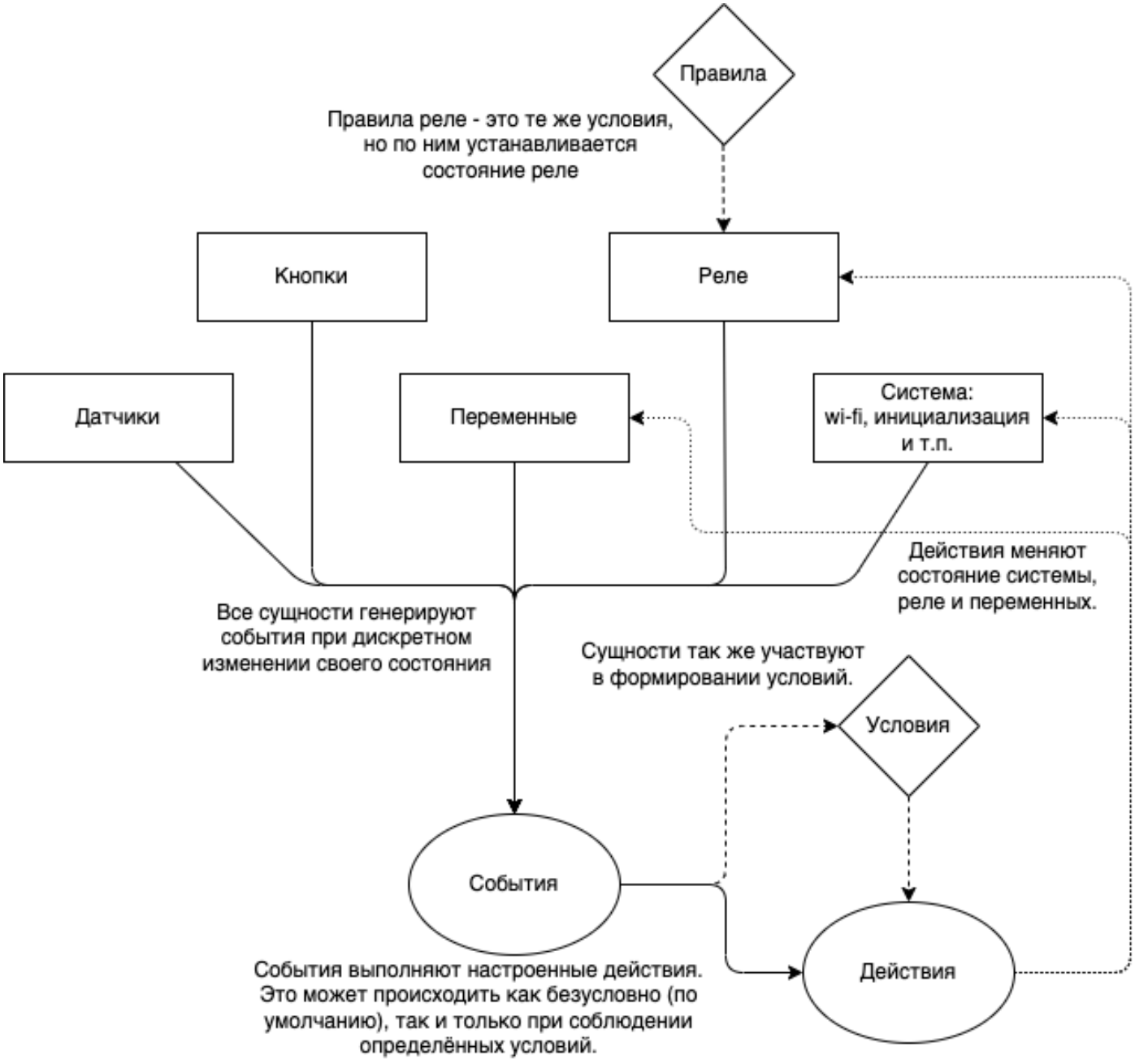
Логика управления и принятия решений

Cliff

6 June, 2022

Оглавление

1 Логика управления и принятия решений



1.1 Переменные

Переменные нумеруются от 1 до 255, хранят какое-то целочисленное значение, которое может применяться в условиях и событиях, и над которым можно совершать какие-то операции.

Простейшие операции – добавление единицы и убавление на единицу с указанием цикличного предела. Т.е. «добавить единицу с циклом 3» означает, что как только переменная при увеличении превысит значение «3», будет сброшена в «0» и цикл пойдёт сначала.

Переменные могут быть запущены в режиме «таймер» - каждую секунду автоматически будет прибавляться единица, и как только значение достигнет заданной величины, таймер будет остановлен или сброшен в «0» с повторением цикла, а переменная сгенерирует соответствующее событие.

В условиях переменные работают в двух вариантах: можно сравнивать с заданной величиной само значение переменной (больше/меньше/равно) либо можно сравнивать с заданным целым остаток от деление на другое заданное целое.

Переменные не требуется отдельно создавать. Это происходит автоматически, если им менять значение.

1.2 События

Генерируются различными сущностями (кнопками, реле, различными датчиками и т.п.), могут выполнять любые действия.

- **init**

  Старт устройства: сработает в момент, когда полностью закончилась инициализация, непосредственно перед тем, как начать цикличную обработку.

- **initctrl**

  Самое первое после старта устройства подключение к серверу управления – может быть полезно для удалённой инициализации устройства.

- **ctrl [on | off]**

  Появилось или пропало управление через сеть.

- **wifi [on | off]**

  Появилось или пропало соединение к wifi-сети (любой).

- **wifi ssid <ssid>**

  Успешное подключение к wifi-сети <ssid>.

- **onewire(0123456789abcdef) [found | connect | disconnect]**

  Подключение или отключение onewire-устройства с определённым адресом.

  - found – появление устройства без ожидания его стабильного подключения.
  - connect – стабильное подключение в течение 1,5 сек, следует такое событие с задержкой 1-2 сек после found.
  - disconnect - отключение устройства (если не отвечает несколько циклов подряд).

  Порядок следования событий такой: found > connect > disconnect.

- **relay(N) [on | off]**

  Включение или выключение реле N.

- **relay(N) force [on | off | schedule | clear]**

  Изменение состояния триггера «принудительное включение» для реле N.

- **relay(N) schedule [on | off | force | clear]**

  Изменение состояния триггера «по расписанию» для реле N.

- **button(N) single**

  Одиночное нажатие на кнопку N.

  Например: button(5) single

- **button(N) double**

  Двойное нажатие на кнопку N.

- **button(N) long**

  Длинное (не менее 3 сек) нажатие на кнопку N.

- **button(N) pushed**

  Кнопка N была нажата (любое нажатие на кнопку, может случиться одновременно с событием button(N) single).

- **button(N) released**

  Кнопка N была отпущена.

- **var(N) set [XX | any]**

  Любая (в т.ч. по таймеру) установка переменной N в значение XX или вообще любое.

- **var(N) timeout**

  Окончание цикла работы таймера у переменной N.

- **gsm([NN | any]) call**

  Входящий вызов с номера NN или любого номера.

- **gsm([NN | any]) sms**

  Входящая SMS с номера NN или любого номера.

  Это событие не генерируется при отправке сообщений вида «info», «relay», «temp», «do …».

  Текст сообщения пока никак не обрабатывается.

- **loclink(TOKEN) [connect | disconnect]**

  Подключение или отключение устройства с токеном TOKEN, которое передаёт нам свои данные.

- **exec N**

  Это событие может быть вызвано либо командой do exec N, либо функцией API exec(num => N), служит для объединения исполнительных команд в функции.

На один и тот же источник события может быть настроено несколько событий. Например, если требуется обработать различный набор условий. Т.е. элементы-события, например с кодом button(2) single, могут повторяться сколько угодно.

1.3 Расписание

Частный случай событий, которые генерируются в определенные моменты времени. Для указания периодичности выполнения используется формат «crontab».

1.4 Формат crontab

Периодичность задания указывается текстовой строкой в том же формате, как и в файле crontab для Unix-систем. Отличие только в использовании секунд.

Общий формат – через пробел указываются числами: секунда, минута, час, день месяца, месяц, день недели.

Каждая из этих величин указывается либо явно (например: 1), либо маской «*» (любая цифра), либо диапазоном (например: 5-8). После значения можно указать периодичность через дробь (например: /5). Также можно задать несколько вариантов, разделённых запятыми.

Пример указания каждой величины (например, «секунды»):

- 1 – каждый раз, когда секунда равна «1»,
- 5-8 – если значение секунды лежит в диапазоне от «5» до «8» включительно,
- */5 – каждые 5 секунд («0», «5», «10» и т.д.)
- 4-15/5 – каждые 5 секунд, начиная с «4» и заканчивая «15»-ой («4», «9», «14»),
- 4/5 – эквивалентно записи: «4-59/5» - т.е. каждые 5 секунд, начиная с «4»,
- 5,6,9 – только в «5»-ую, «6»-ую и «9»-ую секунды.

Величины по маске «*» в конце строки можно не указывать.

Пример «Выполнять каждую минуту»:

```
1
```

Пример «Выполнять раз в час в 30-ую минуту»:

```
0 30
```

Пример «Выполнять каждые 3 минуты»:

```
0 */3
```

Пример «Выполнять каждый день с 18:00 до 20:00 раз в час»:

```
0 0 18-20
```

Пример «Выполнять в 8:00 в рабочие дни»:

```
0 0 8 * * 1-5
```

Пример, который никогда не выполнится:

```
0 0 1 30 2
```

1.5 Действия

Могут выполняться в событиях

- **exec N**

  Вызывает событие exec N, где N от 1 до 255.

- **relay(N) [on | off | toggle]**

  Устанавливает для реле N текущее состояние в on, off или переключает между ними, произойдёт это только если по правилам текущее состояние: «current» или «nonset».

  Если правилами реле на момент вызова команды после всех вычислений установлено какое-то явное состояние реле («on» или «off»), то переключения реле не произойдёт, а в логи будет записана ошибка.

- **relay(N) impulse M**

  Устанавливает для реле N текущее состояние в on на время M в миллисекундах от 0 до 65000.

  Как и команда /relayset, на устройствах с поддержкой текстовых конфигов (работают правила реле) эта команда меняет состояние реле только если правилами установлено значение «current» (не менять текущее значение).

  Данную процедуру можно реализовать через переменные. Однако, т.к. эта процедура часто требуется, она вынесена в более простую реализацию в виде одной команды.

  В отличие от работы переменных, через эту команду можно указать более точное время работы, однако, оно не будет срабатывать с точностью до одной миллисекунды. Связано это с процедурой работы ядра прошивки. Реальная погрешность срабатывания равна примерно 50-100мс, зависит от сложности настроек на устройстве.

  Если в качестве интервала указан «0», то реле включится и сразу выключиться. Для расширителя портов в этом случае время между включением и выключением может составлять до 50 мс – это вызвано временем передачи команды по шине i2c. Для пинов на самом микроконтроллере задержка около 1-5 мс.

- **relay(N) [forceon | forceoff | forcetoggle | noforce]**

  Устанавливает для реле N триггер «принудительное включение» в необходимое положение (например: relay(2) noforce – обнулить триггер для реле 2).

- **relay(N) [scheduleon | scheduleoff | scheduletoggle | noschedule]**

  Устанавливает для реле N триггер «по расписанию» в необходимое положение.

  Например: relay(2) scheduletoggle – переключить триггер между состояниями on-off для реле 2.

- **var(N) reset**

  Сбрасывает значение переменной N.

- **var(N) set X**

  Устанавливает значение переменной N в X.

- **var(N) inc X**

  Прибавляет к переменной N единицу: если X положительное число, то переменная будет циклично увеличиваться в диапазоне от 0 до X, при отрицательном значении – от X до 0.

- **var(N) dec X**

  Уменьшает переменную N на единицу: в диапазонах от X до 0 и от 0 до X при положительном и отрицательном X – соответственно.

- **var(N) timeout X**

  Запускает таймер – каждую секунду значение будет увеличиваться в диапазоне от 0 до X, и через X секунд будет сгенерировано событие «var(N) timeout», после чего переменная будет сброшена, а таймер остановлен.

- **var(N) timer X**

  Запускает такой же, как и «timeout», таймер, но таймер не будет остановлен, а продолжит циклично генерировать событие «var(N) timeout» каждые X секунд.

- **display [next | prev | auto]**

  Переключение дисплея:

  - displaynext – переключить на следующую страницу,
  - displayprev – на предыдущую страницу,
  - displayauto – режим авто переключения страниц.

- **gsm(NN) call**

  Звонит на номер NN.

- **gsm(NN) sms TEXT TEXT**

  Отправит на номер NN SMS с текстом TEXT TEXT (всё, что находится после «sms» с пробелом до конца строки, считается текстом).

- **gsm hangup**

  Сбросит текущий вызов (актуально и для входящих, и для исходящих).

- **gsm restart**

  Перезагрузит и переинициализирует модем (перезагрузка самого модема пока не реализована, но все остальные процедуры сброса и переинициализации выполняются).

- **pwm(N) value X [time T | speed S]**

  Устанавливает для ШИМ N относительную ширину импульса в X (0..65535) мгновенно или за время T, или со скоростью S (см. аналогичную команду API).

- **pwm(N) width X [time T | speed S]**

  Устанавливает для ШИМ N ширину импульса в X микросекунд мгновенно или за время T, или со скоростью S (см. аналогичную команду API).

- **pwm(N) [on | off | toggle] [time T | speed S]**

  Устанавливает для ШИМ N максимальную, минимальную или обратную ширину импульса мгновенно или за время T, или со скоростью S.

- **pwm(N) stopchange**

  Для ШИМ N останавливает плавное изменение ширины импульса.

- **servo(N) set X [time T | speed S]**

  Устанавливает серво-приводу N угол в X градусов (0..360) мгновенно или за время T, или со скоростью S.

- **to(TOKEN) CMD**

  Отправляет на устройство с токеном TOKEN команду CMD.

  Команда не проверяется на синтаксис на устройстве-отправителе, т.к. он может зависеть от версии прошивки на устройстве-приёмник.

  Само устройство TOKEN должно быть известно нашему устройству. Для этого удалённое устройство должно что-либо отправлять нам. Например, данные по датчикам.

- **wifinext**

  Переключает wifi к следующей сети из списка.

- **restart**

  Перезагрузка устройства.

1.6 Условия

Условие – это выражений сравнения различных параметром устройства с заданной величиной.

Применяются как самостоятельный элемент в правилах реле либо как дополнительный элемент для событий и расписания.

- **init**

  Условие сработает только при старте устройства.

- **relayforce(N) [= | != | any] [on | off | force | schedule]**

  Проверит триггер «принудительное включение реле» на определённое значение (= или !=)или на любое установленное (только одно слово «any»). N – номер реле, если не указан, проверяется триггер того реле, в списке правил которого находится это условие.

- **relayschedule(N) [= | != | any] [on | off | force | schedule]**

  Проверит триггер «по расписанию» (аналогично relayforce).

- **relaystate(N) [= | !=] [on | off]**

  Проверит реальное состояние реле (аналогично триггерам).

- **wifissid [= | != | any] [SSID]**

  Проверит наличие подключения к определённой или любой wifi-сети.

- **wifi [yes | no]**

  Проверит наличие wifi-подключения к любой сети (условие «wifi yes» эквивалентно «wifissid any»).

- **ctrl [yes | no]**

  Проверит наличие соединения с сервером-контроллером.

- **temp(0123456789abcdef) [> | >= | < | <= | = | !=] XX.XX**

  Сравнит температуру термодатчика со значением XX.XX (адрес термодатчика задан в hex-формате, значение температуры может быть дробным).

- **onewire(0123456789abcdef) [connected | disconnected]**

  Сработает при наличии подключенным (или отключенным) onewire-устройство с определённым адресом.

- **uptime [> | >= | < | <= | = | !=] XX**

  Сравнит время работы устройства со значением XX (в секундах).

- **var(N) [> | >= | < | <= | = | !=] XX**

  Сравнит значение переменной N со значением XX;.

- **var(N) mod MM [> | >= | < | <= | = | !=] XX**

  Сравнит остаток от деления значения переменной N на MM со значением XX.

- **button(N) [pushed | released]**

  Проверит, нажата/отпущена ли кнопка N.

- **analog(N) [> | >= | < | <= | = | !=] XX**

  Сравнит значение аналогового датчика N со значением XX.

- **from(TOKEN) SENSOR [> | >= | < | <= | = | !=] XX**

  Сравнит значение датчика с именем SENSOR на устройстве с токеном TOKEN со значением XX (подробнее об использовании датчиков с внешних устройств в разделе «LOCLINK – локальная связь между устройствами»).

Примеры условий:

- Флаг «принудительное включение» для реле 5 установлен в значение «on»:

  ```
  relayforce(5) = on
  ```

- Флаг «принудительное включение» для текущего реле установлен в любое значение:

  ```
  relayforce any
  ```

- Установлено подключение к любой wifi-сети кроме «office»:

  ```
  wifissid != office
  ```

- Температура датчика «0123456789abcdef» строго выше 25.5 градусов Цельсия:

  ```
  temp(0123456789abcdef) > 25.5
  ```

- Кнопка 1 нажата:

  ```
  button(1) pushed
  ```

1.7 Правила реле

Это набор условий в определённом порядке, где к каждому условию прикреплено действие:

- Установка состояние реле. При срабатывании условия, следующие правила уже не проверяются.
- Перепрыгнуть N правил.

Действие выполняется только при выполнении условия.