



ilusder 22 апреля 2015 в 10:47

Управление GSM модулем с AVR

DIY или Сделай сам

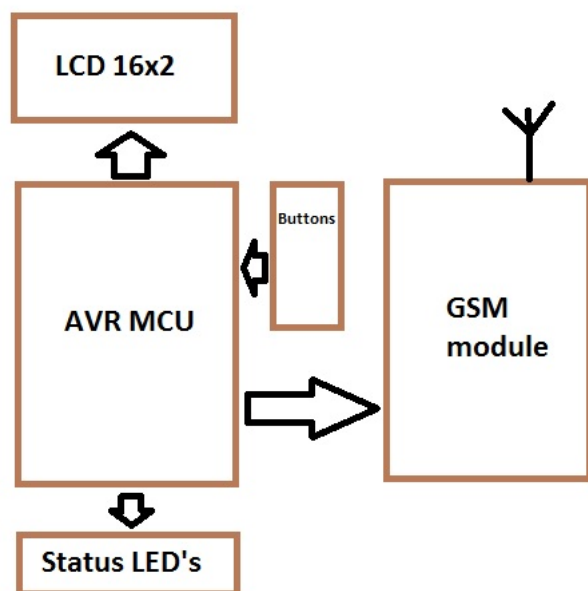
Из песочницы

Идея проекта: спроектировать устройство на базе микроконтроллера AVR для управления готовым GSM модулем (я выбрал модуль TC35 с SIEMENS, но можно использовать любой другой, если используется связь через последовательный порт RS232). Устройство должно быть компактным, минимально простым и надёжным.

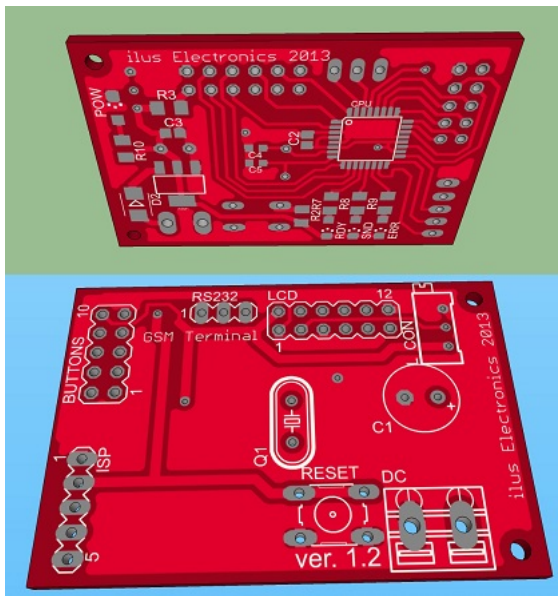
Отправка заранее записанного в память сообщения на указанный номер должна выполняться после нажатия кнопки. Всего нужно было 6 к для отправки на 6 различных номеров. Для индикации процессов были выбраны 3 светодиода (Ready, Send, Error), но в последствии был добавлен алфавитно цифровой LCD 16x2 (скорее, для отладки устройства, чем для обычного использования).

Проектировалось всё дело на плате Pinboard II (Rev 2) со стандартным процессорным модулем на ATmega16. На готовом устройстве схема немного другой (и микроконтроллер использовался ATmega8). Программа писалась в AVR Studio 4.19. В проекте были использованы разл заголовочные файлы (#include) для переключения между Pinboard и готовым устройством.

Общая схема системы:



Для контроллера была выпилена такая платка:



Времени было много, поэтому в последствии я заказал платы у китайцев:



А когда с железом было закончено, следом пошёл процесс программирования. Всё написано на Си под AVR Studio 4.19. Полный проект выкладываю в конце статьи, если кому интересен полный код. Но пока поговорим об общении с GSM модулем.

Полный перечень AT команд есть на каждый модуль в его документации. Но отправка сообщения происходит несколькими командами.

```
//команда:
AT+CMGF=1<enter>
//ответ модуля:
OK<enter>
```

Переводит модуль в текстовый режим. Цифровой режим я пока не освоил (пока не было необходимости). Ответ модуля на начальных стад проекта никак не использовался. Но потом (когда был написан дешифратор команд) служил условием продолжения отправки или сообщен ошибке протокола. Идём дальше:

```
команда:
AT+CMGS=(номер телефона)<enter>
ответ модуля:

>
отправляем сообщение:
Hello, GSM module!<ctrl-Z>
ответ модуля:
+CMGS: 62

OK
```

После набора сообщения, нужно отправить не Enter (0x0D) а CTRL-Z (0x1A). Ответ модуля после отправки содержит порядковый номер отправляемого сообщения.

Для отправки команд в модуль в модуль и получения ответов я использовал два кольцевых буфера со входящими и исходящими индексами

Для большей понятности кода приведу заголовки:

```
#define BUF_SIZE 128
#define BUF_MASK (BUF_SIZE-1)
#define IN_BUF_SIZE 64
#define IN_BUF_MASK (IN_BUF_SIZE-1)

volatile char buffer[BUF_SIZE]="";
volatile char inbuf[IN_BUF_SIZE]="$"; //inner buffer of USART
volatile uint8_t ind_in=0, ind_out=0, rxind_out=0, rxind_in=0, mess = 0;
```

Запись строк или отдельных символов в буфер производилась обычными функциями:

```
//sending RS232 with interrupt
void SendByte(char byte)
{
    buffer[ind_in++] = byte;
    ind_in &= BUF_MASK;
}

void SendStr(char *string)
{
    while (*string!='\n') //check if End
    {
        SendByte(*string);
        string++;
    }
}
```

А отправка производится через обработчик прерывания:

```
//Sending data from buffer
ISR (USART_UDRE_vect)
{
    UDR = buffer[ind_out++]; //запись из буфера
    ind_out &= BUF_MASK; //проверка маски кольцевого буфера
    if (ind_in == ind_out) //если буфер уже пуст
    {
        UCSRB &= ~(1<<UDRIE); //disable instrupt UDR empty
        UCSRB |= (1<<RXEN); //разрешение приёма после окончания отправки
    }
    sei ();
}
```

Теперь для отправки нужно записать нужную команду в буфер (включая конечный символ \n), а затем включить прерывания опустошения регистра отправки (UDR):

```
SendStr ("AT+CMGF=1\n");
SendByte(CR); //отправляем <Enter> (0x0D)
UCSRB &= ~(1<<RXEN); //Запрещаем приём на время отправки
UCSRB |= (1<<UDRIE); //Включаем прерывание и происходит отправка
```

Пока идёт отправка, можно отправить надпись на LCD или просто подождать (delay).

Писать в это время в буфер нельзя. Опытным путём обнаружил, что модуль не успевает обработать сплошной поток команд. А остановка происходит, когда буфер пуст (входящий и исходящие индексы равны).

И таким образом мы отправляем сообщение. В зависимости от нажатой кнопки (в главном цикле я сканирую порт) происходит отправка

сообщения:

```
while (1)
{
    tmp = PINC;
    switch (tmp)
    {
        case 1:  send_sms(0,NUM1); break;
        case 2:  send_sms(0,NUM2); break;
        case 3:  send_sms(0,NUM3); break;
        //и так далее...
        default: break;
    }
    Ready_Snd (); //перевод обратно в режим готовности
}
```

В функцию отправки я посылаю номер выбранного сообщения (их у меня 2 типа) и номер телефона. Можно даже отправить команду на звонок теми же AT командами. Всё зависит от необходимой функции.

Теперь о получении команд с модуля.

Модуль отправляет множество команд. Например, OK, RING, ERROR...

Иногда нужно, чтобы при получении команды контроллер смог опознать её и выполнить какое-то действие. Например, получен входящий з
Модуль при этом отправляет в контроллер:

```
RING

RING

RING
```

В зависимости от настроек модуля, может отправлять ещё и номер того, кто звонит. Пока нет никакой программы, контроллер ничего с этим сделать не сможет (в лучшем случае) или (в худшем) сделает что не то, а то и вовсе зависнет (не сможет выйти из прерывания).

Требования к коду обработки:

1. Минимальное количество времени на сохранение полученных команд. Никаких задержек в программе прерывания быть не должно. Пот с полученным массивом будем делать что угодно.
2. Сохранение всех полученных команд в одном буфере. Для разделения отдельных будем использовать символ \$.
3. Распознавание распространенных команд в числовые коды. Например, OK будет 1, ERROR — 4, RING — 2.

Приведу заголовки из предыдущей статьи с поправками:

```
#define BUF_SIZE 128    //Исходящий буфер
#define BUF_MASK (BUF_SIZE-1)
#define IN_BUF_SIZE 64 //Входящий буфер
#define IN_BUF_MASK (IN_BUF_SIZE-1)

volatile char buffer[BUF_SIZE]="";
volatile char inbuf[IN_BUF_SIZE]="$"; //inner buffer of USART
volatile uint8_t ind_in=0, ind_out=0, rxind_out=0, rxind_in=0, mess = 0;
volatile uint8_t com_detect=0; //сюда будет записана обнаруженная команда

#define TIMEOUT 100    //на случай если команда так и не принята
```

Пишем обработчик прерывания приёма данных:

```
//recieving Data from RS232
ISR (USART_RXC_vect)
{
    uint8_t tmp;
    tmp = UDR;
```

```

    if (tmp == 0x0D)           //получен конец команды - <enter>
    {
        mess++; //one more message
        inbuf[rxind_in++] = '$'; //вставляем разделитель в буфер
        rxind_in &= IN_BUF_MASK;
    }

    else

    {
        if (tmp != 0x0A) //очистка непонятого символа с модуля
        {
            inbuf[rxind_in++] = tmp; //записываем в буфер
            rxind_in &= IN_BUF_MASK;
        }
    }

    sei ();
}

```

Теперь у нас все команды записаны в буфере. Можно в свободное время проверить переменную mess и если она не равна нулю — запуск обработчик команды. В самом проекте были добавлены команды для LCD экрана. Здесь я их пропущу за ненадобностью.

```

void rx_check_in (void)
{
    uint8_t count=0;
    com_detect = 0; //обнуление команды (чтобы не мешал предыдущий мусор)
    while (1)
    {
        if (inbuf[rxind_out] != '$') //обнаружен конец команды (разделитель)
        {
            com_detect ^= inbuf[rxind_out++]; //делаем XOR полученным символам
            rxind_out &= IN_BUF_MASK;
            count++; //считаем, сколько символов в команде
        }

        else

        {
            rxind_out++;
            rxind_out &= IN_BUF_MASK;
            code_com (count);          //!!! важная часть - декодировать команду
            break;
        }
    }
}

```

Полученные символы мы пропускаем через мясорубку. Делаем XOR операцию. Получаем таким образом уникальный код (не уверен на счёт уникальности, но пока не подводило). R^I^N^G нам даст 0x12. O^K даст 0x04. Этот код и количество символов (в команде) сохранены в переменных com_detect (глобальная) и count. Теперь запустим обработчик:

```

void code_com (uint8_t count)
{
    switch (com_detect)
    {
        case (0x12): if (count == 4) com_detect = 2; break; //R^I^N^G
        case (0x58): if (count == 5) com_detect = 3; break; //ERROR
        case (0x04): if (count == 2) com_detect = 1; break; //OK
        case (0x5C): if (count == 3) com_detect = 4; break; //ATI
        default: com_detect = 0;
    }
}

```

Распознали команду. Количество символов я ввёл для надёжности на случай если в длинной команде XOR код совпадёт. Распознаваемые команды можно добавлять. Нужно только подсчитать (или макросом) XOR код желаемой команды и присвоить ей цифру.

Теперь в com_detect у нас полученная команда. Теперь устройство может отреагировать SMS сообщением на полученный звонок:

```

while (1)
{
    if (mess != 0) //if we have mess in buffer

```

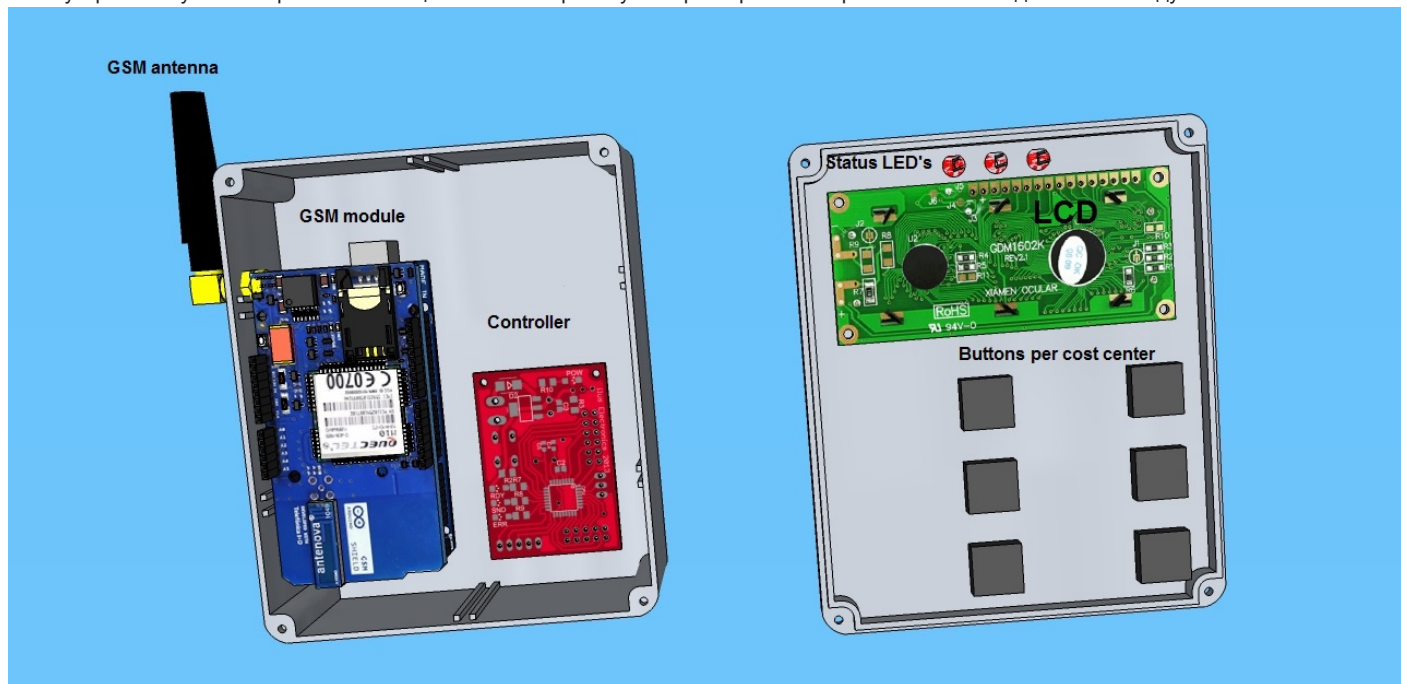
```

{
  // code
  mess--; //minus one
  rx_check_in (); //распознаём отдельную команду
  if (com_detect == 2) //если была команда RING (код 2)
  {
    //Посылаем сообщение
    // и принимаем входящие команды (OK)
    if (!send_sms (1,NUM0)) ErrMes (); //если после отправки не было команды OK
  }
  //тогда выдать сообщение о ошибке протокола
  com_detect = 0; //обнуляем команду
}

```

Так можно обрабатывать разные полученные команды.

Итог: устройство умеет отправлять сообщение на телефон и умеет реагировать на различные команды от GSM модуля.



Спасибо за внимание.

Готовый проект для Pinboard II выкладываю как пример.

Метки: GSM, SMS, avr, pinboard ii

↑ +16 ↓ 170 29,9k 13



6,0

Карма

0,0

Рейтинг

0

Подписчики

Ilya Deryabin @ilusder

Пользователь

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

5 мая в 13:13

Уязвимость в протоколе SS7 уже несколько лет используют для перехвата SMS и обхода двухфакторной аутентификации

↑ +37 38,6k 107 82

8 октября 2014 в 08:53


Управляем розеткой по SMS

↑ +3 👁 31,1k 📖 88 💬 0

11 марта 2010 в 10:39

Запасной путь для SMS

↑ +81 👁 6,4k 📖 52 💬 133




Производство электроники-ProSMD
Prosmid- профессиональное производство электроники, **плат**, РЭА. 20 лет в производстве!




Ищите датчики смещения?
Гарантия качества. В наличии на складе. Приемлемые цены. Заказывайте!

Реклама


Комментарии 13

 **progchip666** 22.04.15 в 12:16 📌 📖 ⬆


Хорошая статья, жалко устарела лет этак на десять. Помню в 2005 как раз многие, в том числе и я осваивали общение с GSM модемами, тогда можно было заработать даже на самопальной GSM сигналке на основе SMS и звонков. К сожалению это время уже ушло, сегодняшние задачи требуют перданных другими способами. Хотя как повод вспомнить бывшие дни и пошевелить мозгами статья не плохая. Радует опять же что не на основе ардуи всё сделано, а собственная плата разработана.

 **bitterman** 22.04.15 в 12:33 📌 📖 🔄 ⬆


Напрасно вы так считаете. Телеметрия всяких домовых датчиков учёта потребления, например, вполне себе на GSM/GPRS работает. Хоть смска!

 **dion** 22.04.15 в 13:09 📌 📖 🔄 ⬆

GPRS — это уже другой совсем другой уровень. На ATmega такого нормально не добиться. Разве что использовать уже не просто GSM-модем командами а заметно более навороченный модуль, который по факту уже не модем, а полноценное устройство, которое внутри себя реализует стек, начиная от PPP и заканчивая FTP/HTTP клиентами. А наружу торчит примитивный интерфейс вроде AT+HTTP, чтобы можно было этим управлять каким-нибудь AVR или Arduino.


 **bitterman** 22.04.15 в 13:11 📌 📖 🔄 ⬆

За всех не скажу, но модемы SIMCOM (типа Sim900) имеют встроенный TCP-стек, управляемый AT-командами. Даже FTP-клиент есть готов для них всего-то и надо AVRкой AT команды принимать и посылать.

 **dion** 22.04.15 в 13:18 📌 📖 🔄 ⬆


Я как раз ссылку на такое и привел, назвав это AT+HTTP. Они еще много чего 'завернули' в AT команды, вроде TCP socket-ов. А особо навороченные туда еще и камеру с флешкой зачем-то прикрутили, звонки. Только упирается все в то, что управляется этот монстр на несколько порядков более мелким AVR-ом. Который по сути ничего сам не умеет кроме того самого AT+HTTP.

Я в итоге для экспериментов остановился на варианте обычного Raspberry Pi и старой мобилки с GPRS. И нормального TCP/IP без всяки AT+HTTP.


 **bitterman** 22.04.15 в 13:21 📌 📖 🔄 ⬆

А в идеале писать прошивку самого модема, типа user application, тогда не надо будет вообще AVR для простых проектов.


Я к «нормальному TCP/IP» с rppd и необходимостью реагировать на разрыв соединения, затупы модема (требующие его ребута) отнс не очень хорошо. Хотя и имею решение, где у меня это более-менее работает (телематические блоки в маршрутных автобусах)

 **IVAN2001** 22.04.15 в 14:54 📌 📖 🔄 ⬆


На презентации SIM900 демонстрировали работу скриптов для модема, которые реализовывали получение GPS-координат и пере, их в сокет. Как раз обошлись без внешнего контроллера.

 **IVAN2001** 22.04.15 в 14:50 📌 📖 🔄 ⬆

Мы на ATmega и обновление собственной прошивки по GPRS через SIM300 делали. И выкладывание логов на FTP. И выгрузку запакованнс буфера в сокет... т.ч. все возможно, если захотеть.

 **progchip666** 22.04.15 в 15:02 📌 📖 🔄 ⬆

Я писал про SMS. Продолжение по GPRS приветствую! Сам с удовольствием почитаю, я слегка в теме, но не на уровне профи. С удовольстви почитаю материал на эту тему. Так что ждём продолжения.

 Tutanhomon 22.04.15 в 15:01  

заказал платы у китайцев

а не подскажите где и по чем?



 ilusder 22.04.15 в 17:47    

Это где-то года 2 назад было. 25 долларов с пересылкой. Сейчас многие платы делают небольшими тиражами и не дорого.



 Delsian 23.04.15 в 15:12    

www.seeedstudio.com/service/index.php?r=pcb 10 баксов цена вопроса плюс 3 бакса мне стоила пересылка во Львов.



 Tutanhomon 23.04.15 в 15:12    

Большое спасибо



Только полноценные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Теория игр, мышиная возня и спущенная шина

 +6  1,2k  5  8

СБУ провела обыск у основателя ForkLog и изъяла всю технику из офиса

 +9  2,4k  2  8

Как прочитать большой файл средствами PHP (не грохнув при этом сервак) Хабр





 +15  2,6k  75  5

SQL Server 2017 JSON Хабр

 +13  3,2k  40  0

Держите мобильные телефоны подальше от тела, не кладите в карман и не спите вместе. Новые советы медиков-технофобов

 +5  7,5k  11  38

| Аккаунт | Разделы | Информация | Услуги | Приложения |
|--|--------------|--------------------|------------------|---|
| Войти | Публикации | О сайте | Реклама |   |
| Регистрация | Хабы | Правила | Тарифы | |
| | Компании | Помощь | Контент | |
| | Пользователи | Соглашение | Семинары | |
| | Песочница | Конфиденциальность | | |
|  © 2006 – 2017 «TM» | | Служба поддержки | Мобильная версия |  |