Interfacing Arduino to a Cellular Phone

By Oleg Mazurov

One of the main motivations for adding asynchronous CDC support code to rev.2.0 of USB Host Library was to be able to use cell phones in Arduino projects – establish simple data exchange via SMS, take pictures or connect to the Internet. Second hand phones are inexpensive yet quite capable. Also, m2m (machine to machine) SIM cards start at $4-$6/mo, some even allow for free incoming SMS. All that makes a cell phone an attractive communication option for hobby projects. In this post, I will be talking about basics of cell phone control using data port and AT commands. I will also present simple terminal emulator sketch – to use the code you will need an Arduino board, USB Host Shield, as well as USB Host Shield 2.0 library.

Motorola RAZR talks to Arduino

Modern (<10 year old) phones have standard gsm chip interface implemented and accessible via so-called "data port". the oldest implement ttl level asynchronous serial by means of "custom" usb data cable, which is just proprietary connector on one end, other usb-to-serial converter (almost always prolific pl2303) between them. newer cell built-in. motorola usually terminate port mini-usb connector, others, like samsung sony ericsson, use cable. in these almost cdc acm type. many functions phone can be accessed at commands, similar to commands used control hayes modems. are defined 3GPP TS 07.07 (look for the latest version, which is 7.8.0). Cell phone manufacturers also define their own AT commands. In documentation AT commands are usually presented in uppercase, however, most phones accept lowercase just as well. A command shall be followed by CR,LF (usually Enter key). If a command is accepted, OK is returned, along with response. If command is not recognized, ERROR is returned. Some commands will be accepted in certain phone states and rejected in others.

There are several variants of invoking each command:

1. **Execute** This variant is used for commands which require no parameters. The format is **AT***command*. For example, the following command returns power source and battery charge level:

   ```
   AT+CBC

   +CBC: 2,66

   OK
   ```

   The response means that a phone is powered from external power supply (2) and battery level is 66 percent.

2. **Test** This variant is used to query parameters and their values for the command. The format is **AT***command***=?**.

   ```
   AT+CIND=?
   +CIND: ("Voice Mail",(0,1)),("service",(0,1)),("call",(0,1)),("Roam",(0-2)),("signal",
   (0-5)),("callsetup",(0-3)),("smsfull",(0,1))

   OK
   ```

   This command outputs all indicators available on the phone screen, and their possible values.

3. **Get** This variant is used to query current settings for the command. The format is **AT***command***?**.

   ```
   AT+CIND?
   +CIND: 0,1,0,0,4,0,0

   OK
   ```

   Comparing to output of previous “Test” variant of the same command we can see that “Service” indicator on the phone screen is on and “signal” indicator is at level 4.

4. **Set** This variant is used to change settings for the command. The format is **AT***command***=***param,param...*.

   ```
   AT+CKPD="1"
   OK
   ```

This command simulates a keypress on phone keypad. The key pressed is number “1”.

5. **Unsolicited response** is output of some event other than command result. The format is **+COMMAND:** *result*. It is th same as command response sans command itself. Take a look at the following example:

   A `+CMER` command enables or disables sending of unsolicited result codes in the case of key pressings, display changes, and indicator state changes. The parameters for this command are as follows:

   ```
   AT+CMER=?
   +CMER: (0,3),(0,1,2),(0),(0,1,2),(0)
   OK
   ```

   First parameter sets output mode, next three parameters turn output on or off for keypad, display and indicator. Last parameter controls buffering. On Motorola RAZR, the default state of this command is as follows:

   ```
   AT+CMER?
   +CMER: 0,0,0,0,0
   OK
   ```

   which means all responses off. Now, if we turn reporting on and set keypad to on:

   ```
   AT+CMER=3,1,0,0,0
   OK
   ```

   and start pressing buttons on the phone, we will see the following:

   ```
   +CKEV: "E",1
   +CKEV: "E",0
   +CKEV: "1",1
   +CKEV: "1",0
   +CKEV: "2",1
   +CKEV: "2",0
   +CKEV: "3",1
   +CKEV: "3",0
   +CKEV: "E",1
   +CKEV: "E",0
   ```

   Here, “E” means “Red” button (the one used to turn phone on/off, terminate a call, abort an action and many other things), “1”, “2”, and “3” are numerical buttons, and number after the comma means “press” if 1 and “release” if 0. It is now evident that I first pressed “Red” button to turn screen on, then pressed “123” and then pressed “Red” button agai to erase the digits.

The following sketch is a simple terminal program. Only main loop is shown, Full text is available in examples directory on gitHub. Compile it, load, attach your phone to the USB host shield and open the terminal window. If a phone is detected successfully, sketch outputs configuration descriptor and waits for keyboard input.

```
void loop()
{
    Usb.Task();

    if( Usb.getUsbTaskState() == USB_STATE_RUNNING )
    {
       uint8_t rcode;

       /* reading the keyboard */
       if(Serial.available()) {
         uint8_t data= Serial.read();
         /* sending to the phone */
         rcode = Acm.SndData(1, &data);
         if (rcode)
            ErrorMessage<uint8_t>(PSTR("SndData"), rcode);
       }//if(Serial.available()...

       delay(50);

        /* reading the phone */
        /* buffer size must be equal to max.packet size */
        uint8_t buf[32];
```

```
23         uint16_t rcvd = 32;
24         rcode = Acm.RcvData(&rcvd, buf);
25          if (rcode && rcode != hrNAK)
26             ErrorMessage<uint8_t>(PSTR("Ret"), rcode);
27
28             if( rcvd ) { //more than zero bytes received
29               for(uint16_t i=0; i < rcvd; i++ ) {
30                 Serial.print(buf[i]); //printing on the screen
31               }
32             }
33         delay(10);
34     }//if( Usb.getUsbTaskState() == USB_STATE_RUNNING..
35 }
```

Lines 10-16 contain sending part of the terminal. Keyboard buffer is checked and if not empty, a character is read and sent to the phone (line 13, `Acm.SndData()`). The rest of the loop() is reading the phone and it is a little more complicated.

When USB host sends IN transfer request to the phone, it doesn't know how much data will be received. It could be one byt or it could be a big packet. That's why the buffer allocated in line 22 must be equal to the maximum possible chunk of data, i.e. endpoint's Max.packet size. Most phones have Max.packet size of 32 bytes, one notable exception being Samsung (64 bytes). If you see strange symbols and occasional Arduino resets while outputting large chunks of data, check Max.packet sizes of bulk endpoints and increase buffer size accordingly.

The request is sent in line 24. One important property of `Acm.RcvData()` function is that it returns actual number of bytes received from endpoint. The loop in lines 28-32 uses `rcvd` as a loop counter; this way, only part of the buffer filled during the transfer will be printed.

Now let's start our terminal. Once a terminal session is open, commands can be entered from the keyboard. To check if connection is live, type `at` and press `Enter`. If phone is alive, you should see it replying with `OK`. The next command to try i `AT+CLAC`. This command outputs all supported commands. Other commands can be tried "by hand"; also, it is possible to ge key codes with `AT+CMER`, as was described earlier.

As cool as it may look, entering commands from the keyboard is not very useful for automation. It is desirable to have functions to call numbers, send and receive SMSes and browse the Internet. Luckily for us, such software has already being developed. There are several GSM shields on the market, and GSM/GPRS modules these shields are based on are almost identical to GSM phones. For example, I was able to follow most part of this tutorial using Motorola RAZR phone. Some commands are different but most work. Many commands described on this page also work. Finally, there is very well written GSM Playground Arduino Library, which can easily be modified to use USB methods to send/receive data to the phone. I'm planning on looking into it after finalizing new USB Host library.

Finally, I want to talk a little about compatibility. I checked this sketch with Motorola RAZR, Motorola V220 and Samsung A-777. Motorola phones work out of the box, Samsung requires changing buffer size. I also tried Sony Ericsson TM-506 and TM 717 – the configuration descriptor in these phones is very complex and the `Init()` method won't initialize them correctly. This will be fixed in the future. I'm also curious about other phones – please let me know if your phone worked with this sketch.

Oleg.

June 22nd, 2011 | Tags: Arduino, AT command, cell phone, gprs, gsm, MAX3421E, USB Host Shield | Category: programming, USB, USB Shield | 225 comments - **(Comments are closed)**

225 comments to Interfacing Arduino to a Cellular Phone

« Older Comments 1 2 3

Ahmad
July 3, 2014 at 2:35 pm

Hi Oleg,

Actually I have Arduino uno, Itried to connect it to Motorola v3i razer,
The cell phone worked fine in hyper terminal.
When I used the same code for Arduino nothing happed.
I connected pin 3 and 2 for Rx and Tx respectively and the ground also.
my code is this:

#include

SoftwareSerial mySerial(3, 2); // RX, TX pins

```
void setup() {

pinMode(13, OUTPUT); // Initialize pin 13 as digital out (LED)
pinMode(8, INPUT); // Our button pin
mySerial.begin(4800); // Open serial connection at baud rate of 4800

}

void loop(){

if (digitalRead(8) == HIGH){ // On button press
digitalWrite(13, HIGH); // Turn LED on.
mySerial.println("AT"); // Sends AT command to wake up cell phone
delay(500);
mySerial.println("AT+CMGF=1"); // Puts phone into SMS mode
delay(1000); // Wait a second
mySerial.println("AT+CMGW=\"+14165551234\""); // YOUR NUMBER HERE; Creates new message to number
delay(1000);
mySerial.print("Sent from my Arduino."); // Message contents
delay(1000);
mySerial.write(byte(26)); // (signals end of message)
delay(1000);
mySerial.println("AT+CMSS=1"); // Sends message at index of 1
digitalWrite(13, LOW); // Turn LED off
delay(250);
digitalWrite(13, HIGH); // Turn LED on.
delay(10000); // Give the phone time to send the SMS
mySerial.println("AT+CMGD=1"); // Deletes message at index of 1
digitalWrite(13, LOW); // Turn LED off.
delay(250);
}

}
```

Did I miss something?
Thanks

Oleg Mazurov
July 3, 2014 at 2:44 pm

Where on the phone did you connect those RX, Tx signals? Does your phone have serial input?

Ahmad
July 4, 2014 at 4:52 pm

Oleg Hi,
Actually it is the same port of charching port.
When I connected to PC via USB I could sent sms using hyper terminal.
I cut the cable and connected the green, white, and black wire as Tx, Rx,GND respectively.
I tried to swap the green and white wires, changing the baud rate, Also tried to change the pins on arduino from 2,3 to 0,1 or 3,4 …hopeless.

Thanks

Oleg Mazurov
July 6, 2014 at 12:40 pm

USB and serial are incompatible, you won't be able to get any meaningful data.

Henry
July 9, 2014 at 7:05 am

I'm trying to Galaxy S3(Android4.3) to interface with Bluno by way of USB host shield. the AndroidAccessory.cpp in ADK_release_0512 is buggy of new version as4.3. (available at http://developer.android.com/guide/topics/topics/usb/adk.html)

```
if (protocol == 1) {
Serial.print("device supports protcol 1\n");
} else {
```

Serial.print(“could not read device protocol version\n”);
return false;

I replace it with

if (protocol >= 1) {
Serial.print(“device supports protcol “);
Serial.println(protocol,DEC);
} else {
Serial.print(“could not read device protocol version”);

do you have newer version of Androidaccessory lib for such application?

Henry
July 9, 2014 at 10:51 am

I found a news version ADK package at http://developer.android.com/tools/adk/adk.html. named ADK_package_20120606.
This ADK2011 has fix the getProtocol bug. should I try ADK2012 the latest one?

but I still has issue to work with USB Host Shield, since acc.isConnected never successful, Bluno can not get command from my Samsung Galaxy S3 (4.3). how do you think about Bluno+Usb Host Shield +Android4.3?

with bde like this:
AndroidAccessory acc(“Manufacturer”,
“Project01”,
“Description”,
“Version”,
“URI”,
“Serial”);
acc.isConnected()

Oleg Mazurov
July 9, 2014 at 11:43 am

My ADK code was developed for the initial version of ADK, whatever was added to ADK after 2010(?) is not supported.

Henry
July 9, 2014 at 11:50 am

when I look in what happen to acc::isConnected. I got

Device addressed… Requesting device descriptor.
dLength ->181idVendor ->1038<-
found possible device. swithcing to serial mode

but 1038 is not a solid idProduct according to ADK2011.then it always exit.what this 0x1038 mean, in ADK2012 I know the following idProduct:
In AOA 1.0, there are only two USB product IDs:

0x2D00 – accessory
0x2D01 – accessory + adb
AOA 2.0 adds an optional USB audio interface and, therefore, includes product IDs for the new combinations of USB interfaces:

0x2D02 – audio
0x2D03 – audio + adb
0x2D04 – accessory + audio
0x2D05 – accessory + audio + adb

Kim Moore
August 11, 2014 at 10:18 am

Can you interface a cell phone to a monitor? It is for an elderly person who has difficulty with the small screen and hearing the ringing (not enough db). I would like to plug in through the serial port and create a user interface that makes the phone easier to use. Is arduino helpful for this application?

Oleg Mazurov

August 11, 2014 at 10:25 am

Ringing can be done easily – the phone outputs the word RING at the terminal when the phone rings. Not sure about the rest.

Moog
September 16, 2014 at 2:50 am

I'm making progress after following the posts by "scottyjr" (last post on march 26, 2014, like him I'm using your USB shield with a Motorola V3. I had the exact experiences as him, and I too eventually got 2 way communication, thru the arduino serial monitor, after using "putty" as well.
However once I started to follow the tutorial tronixstuff "Tutorial – Arduino Uno and SM5100B GSM Cellular" it no longer responds to the tutorial.
Using the ACM terminal, if I send-> cell.println("ATDwifes-phone"); the phone screen says "calling wifes-phone divert on" then it rings that phone.(so far so good)
if I send the command-> ATDwifes-phone the phone screen says the same but it doesn't actually ring, and it doesn't say call failed or anything like that. but in the serial monitor window, the line àáí™r-CÍý÷g¾Ë}?Ó¿ëÛM= is printed over and over.
If I use the first sketch in that tutorial "example 263.1" I just get "Starting SM5100B Communication…" then nothing else.
Non of the other sketches in the tutorial work either.
Having got so far it can't be anything much?? Hopefully you can help me please?

Oleg Mazurov
September 16, 2014 at 11:29 am

Garbage in the terminal usually means speed mismatch. I use 115200 in my sketches, make sure your terminal is set to this speed.

Moog
September 17, 2014 at 3:29 am

I've found that speed works best also,
should ATDwifes-phone work thru the serial monitor?

Moog
September 30, 2014 at 2:23 am

I still haven't got any where yet, today I got this response over and over after I sent AT+CMER=3,1,0,0 then pressed a key on the phone (using serial monitor)

```
+CKEV: "4",1
ìÙr-EÍÉ÷g>Ëm?Ó¿ëQ
‹xH±®p:¿O
```

The speed is set to 115200.
Does this help to diagnose?
Also why is putty very very slow and misses key strokes?
Can you confirm that I should be able to use the sketches in the tronixstuff "Tutorial – Arduino Uno and SM5100B GSM Cellular"

Hafiz
January 12, 2015 at 11:35 pm

Hi, I have a question, so does this mean, when we connect the phone to the Arduino Uno, it is the same with connecting Arduino GSM Shield to the Arduino Uno???

Can I make my phone to send an SMS or make a call from that phone when I have something being triggered??

Thanks

Oleg Mazurov
January 13, 2015 at 1:20 am

Yes, this was the idea.

Marcin
November 4, 2015 at 3:32 pm

very nice tutorial, it helped me to connect my T230 to arduino.

Here’s sample program (working) for sending, reading sms and making a call.

Not sure how to check if phone is charged or if it has connection with gsm provider..
http://www.vsx.pl/arduino-i-sony-ericsson-t230-wysylanie-i-odbior-sms-dzwonienie-arduino-t230-sms-call-interface-rx-tx/

« Older Comments 1 2 3