Connecting your Arduino board to an Android Phone

Introduction

I was on an electronics store's website and an advert for an Arduino USBDroid caught my eye. So after a bit of research I purchased one and was blown away with it. Up until then, I had used PIC micro's for doing stuff I wanted, but was always having to re-create common circuits just to get started.
BUT despite the USBDroid website (www.freetronics.com) saying that you can connect it to your android phone, it was left at that pretty much, and I was looking at buying either a Bluetooth shield or a Wi-Fi shield. That was, until I stumbled onto another site that talked about using the ADB (Android Debug Bridge) , and it told you, sort of, how to do it. Eventually, I got it all sorted and thought in the spirit of the free knowledge I have feed off I would pay it back. This document is about how "I" did things. They might not be the best ways, but they worked for me, and I am always keen to hear how things could be done better ☺ One point to note, even though I am using the USBDroid, this will also work with the Arduino USB shield as well I am told. Also, I am using an HTC Sensation, but it is meant to work on any android device. In immortal words of Carl Pilkington, "I that somewhere".

Getting started

So, what to you need to get started? You need Basic for Android, and the ADB libraries for the Arduino. It will also pay to have a look at this site for the MicroBridge. http://code.google.com/p/microbridge/

On the above website, go to the Downloads Tab and select "MicroBridge-Arduino.zip" and download it. Once you have done that, you need to extract the ADB folder contained in the .zip file to the Libraries folder where you have downloaded your Arduino IDE to.
There are examples under the ADO folder. Open the demo one, and try to build it. If it fails, which I think it will, you need to alter some .h and .cpp files. But don't worry, it's a simple thing to do and you won't need to know C or C++ programming to do it ☺

Open the "adb.h" file in your Arduino IDE. Look for "wiring.h" and change it to "witing_private.h" and save it, then do the same with "usb.cpp" and "usb.h".
Why do you have to change these files? It seems that a file changed its name in the latest Arduino release. I am not sure when it changed; I just know I found I had to alter these files. It wasn't mentioned in any websites I looked at, and since I am a developer by trade, I figured out what I needed to do.

This should be all we need to get started.

Practical Example

So now you have done a little bit of donkey work, it's time to get things happening!
I fortunately had a project in mind that actually kicked off this whole Arduino thing for me.
My son has a science project to do and he (or we) chose free energy. I wanted the ability to display a voltage on a screen to monitor input voltage from a device. As it also turned out I needed to display temperature too.
So I started at with the easy stuff first ☺ Get the Arduino to talk to my phone.

The first thing I did was code up the Android app in B4A. The secret or not so secret thing is, the Android app and the Arduino app use sockets to talk to each other which makes life a LOT easier!

Pretty much I create just a standard "ServerSocket" as below

```
Sub Activity_Create(FirstTime As Boolean)
  If ServerSocket1.IsInitialized = False Then
    ServerSocket1.Initialize(4567, "ServerSocket1")
  End If
  Activity.LoadLayout("1")
End Sub
```

As you can see, I am using port 4567. You don't HAVE to user 4567, you can pretty much use any port so long as it's not in use by something else as far as I can tell.

Then I handle a new connection:

```
Sub ServerSocket1_NewConnection (Successful As Boolean, NewSocket As Socket)
  If Successful Then
    Socket1 = NewSocket
    Timer1.Enabled = True
    InputStream1 = Socket1.InputStream
    OutputStream1 = Socket1.OutputStream
    ToastMessageShow("Connected", True)
  Else
    Msgbox(LastException.Message, "Error connecting")
  End If
  ServerSocket1.Listen 'Continue listening to new incoming connections
End Sub
```

And then, I use a timer to check for waiting data:

```
Sub Timer1_Tick
  If InputStream1.BytesAvailable > 0 Then
    Timer1.Enabled = False
    Dim msg As String
    Dim buffer(255) As Byte
    Dim T As String
    Dim V As String

    InputStream1.ReadBytes(buffer, 0, 255)
    msg = BytesToString(buffer, 0, buffer.Length, "UTF8")

    Dim DataFromPort() As String
    DataFromPort = Regex.Split("\|", msg.Trim())

    Dim l As List
    l.Initialize2(DataFromPort)


    T=l.Get(0)
    V=l.Get(1)

    lblName.Text = "Temp: " & T
    lblSize.Text = "Volt: " & V

    Timer1.Enabled = True

  End If
End Sub
```

As you can see I am sending the voltage and temperature readings through delimited by a "|" character and then assigning the split values to a list then getting the items by index. The index of course is known.

And that's pretty much it for the Android programming side of things. However, there is one more thing you need to do. On your android phone, got the "settings" then select the "Applications" menu, then select "Development" and now turn on the "USB Debugging" option. This is the key to the whole thing. If this is not on, nothing's going to happen ☹. I also turn on the "Stay Awake" option too. It stops the screen turning off.

So if you are happy with your B4A app Load it on to the phone.

Now for the Arduino side of thing.
As I mentioned earlier, I am measuring voltage and temperature for my son's science project. Funny how "I" seem to have spent more time on it that him so far ! And all I got from him was a "oh ok" from him when I showed him what I had done ! kids…….

I won't go into the detail of how I capture the temperature and voltage so much as to highlight what you need to do to be able to talk to the Android phone and your program running on it.

First of all, you need to include the adb.h SPI.h file as below at the top of your project file:

```
#include <SPI.h>
#include <Adb.h>
```

Then you set up the new connection object:

```
Connection * connection;   //  remember its case sensitive
```

In the "void setup()" procedure you need to initialize the ADB object and create a connection to the application on your phone:

```
Void setup(void)
{
  ADB::init();
 connection = ADB::addConnection("tcp:4567", true, adbEventHandler);
}
```

Then you need to handle any incoming data events:

```
void adbEventHandler(Connection * connection, adb_eventType event, uint16_t length, uint8_t * data)
{
  if (event == ADB_CONNECTION_RECEIVE)
 {
    //do something if needed
 }
}
```

Now you set up the loop procedure:

```
Void loop(void)
{

  //get the temperature and voltage
 String sTemp = ReadVoltage();

 //Need to convert it to a charArray
 char cTemo[100];
 stemp.toCharArray(cTemp,100);

 //Write the value out to the tcp port
 Connection->writeString(cTemp);

 //Now poll the ADB
 ADB::poll();
}
```

And now you program up you Arduino, plug in your phone into the USB port and turn on the application you wrote in B4A for the phone.

One thing I have noticed is, it can take a few seconds for the phone to recognize its got something connected to the USB port. I have not really had a play with anything to see if I can speed things up as of yet.

And so there you have it. I HOPE this has been helpful and understandable to read and follow.
I can supply any source code upon request and feel free to comment or offer suggestions on all the above. For me it still early days and I still have a lot to learn, but I haven't been this excited about electronics since by first electronics project 35 years ago….I was 7 ☺
My email is jscoulter@gmail.com

Thanks, Jeremy