Revision 0.3 - 11th December 2009

Part 1: Introduction

1. Background

These articles describe how to interface PS3 and Wiimote game controllers directly onto the Arduino. These controllers have not previously been directly connected to an Arduino due to the USB and Bluetooth interfaces and protocols used, and their relative complexity. The game controllers are a good match to the capabilities of the Arduino, and to the imagination of Arduino users, they have not only buttons and joysticks, but also motion sensing, and other goodies which deserve to be in the hands and ideas of Arduino users. Also I hope these articles provide some guidance for people to develop other USB host and Bluetooth applications for small embedded processors

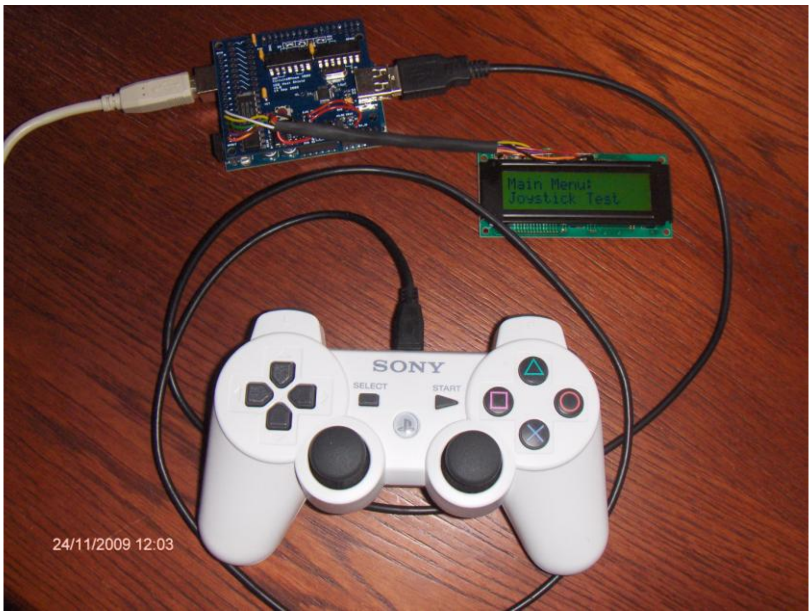
Arduino already comes in versions which have USB and Bluetooth interfaces, but these are not suitable for our chosen game controllers. When used in USB mode, the PS3 controller is a USB device and needs to talk to a USB host. The normal Arduino USB interface is also a device, so not compatible. When in Bluetooth mode, both of these game controllers use the Bluetooth HID protocol and this is not compatible with the commonly used Bluetooth modules which support only the RFCOMM Serial Port Protocol. Arduino users have successfully used a PC as an intermediary between the game controllers and the Arduino, so the PC is host to both devices.

Recently a new shield has been developed by Oleg at Circuits@Home, which provides a USB host interface directly onto the Arduino. This shield is based on the MAX3421E USB controller IC from Maxim, which can operate in both device and host modes. Oleg has also created libraries for the basic functions of the MAX3421E and USB host operation. I have previously used the MAX3421E with success on other processors and decided to try to add support for our game controllers using this shield. There are significant challenges to this due to the complexity of the protocols and the limited flash and SRAM space on the base Atmega168 based Arduino.

The code developed here is licensed under the LGPL License and should be fairly easy to port to other USB host controllers. Given the recent explosion in availability of microcontrollers with integrated USB otg or reduced host, from Atmel, Microchip and others, then I would expect quite a few derived versions.

You will see as the code develops the fairly high complexity of the protocols involved, and such development effort is only really justified in a case like this where no off the shelf alternative has been found. There are integrated USB host solutions from GHI Electronics and FTDI Vinculum, but these again do not support the Bluetooth HID yet. They do support many other devices and classes including mass storage, serial, and HID, they can be interfaced to Arduino and do not put such a drain on the memory and programming of the Arduino.

2. The PS3 Controller



I have used two PS3 game controllers; the first is a Sony Six Axis Dual Shock 3 Wireless controller, and the second is the Madcatz Wireless six axis controller. The Sony has a USB connector and also operates in Bluetooth mode for wireless. The Madcatz comes with its own USB dongle, so uses a proprietary wireless protocol instead of Bluetooth. The PS3 controller is well documented and supported already with drivers and applications on Windows and Linux besides the PS3 itself, so there is plenty of interface information available.

The PS3 controller has a large number of buttons which can be read as digital values and some as analog pressure values. There are two joysticks which can be read with analog values to 8 bit resolution, and also 4 LED's which can be remotely controlled. There are additionally a 3 axis accelerometer and a single axis gyro which give outputs based on the motion of the controller. The Sony controller additionally has two rumble motors.

The following sources have been used to find information and code examples for the PS3 controllers:

http://wiki.ps2dev.org/ps3:hardware:sixaxis

http://python-ps3.svn.sourceforge.net/viewvc/python-ps3/bootable/sixaxis/connection.c?view=markup

http://www.pabr.org/sixlinux/sixlinux.en.html

http://onakasuita.org/ps3

http://www.motioninjoy.com

3. Wiimote Controller

The Wiimote has less buttons than the PS3 controller and the characteristic stick shape. Though less well endowed with buttons and joysticks than the PS3 controller it has a 3 axis accelerometer, rumble, Infra-Red Camera, speaker and LED. It also has an expansion port where additional devices including 3 axis Gyro, and remote joystick can be connected.
The Wiimote does not have a direct USB connection and only works under Bluetooth.

The following sources have been used to find information and code examples for the Wiimote Controllers:

http://en.wikipedia.org/wiki/Wiimote

http://homepage.mac.com/ianrickard/wiimote/wiili_wimote.html

http://libwiimote.sourceforge.net

4. Arduino

I chose to use the lowest spec AtMega168 based Arduino and this forced me to be always conscious of the amount of Flash and data memory used. I made extensive use of the MemoryFree library to keep track of the usage of the 1K of RAM. I tried to limit were possible the number and size of data buffers, and to keep the number of text strings as low as possible.
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1213583720;start=all
I later modified the MemoryFree to allow capture and storage of minimum values. This enables the viewing of free memory at the deepest level of the function calls, where any problems occur.
http://github.com/ribbotson/USB-Host/tree/master/Memory_Free/

In terms of processing power the Arduino does not seem to limit performance. Some complexity is added to the code because it is necessary to avoid blocking code. Blocking code would prevent some of the background processes running. These processes are not absolutely time critical, but must be called regularly. The time taken to write a serial string is usually OK, the time to wait for a key press would not be.

5. USB Host Shield

The Circuits at Home USB host shield is based on the Maxim MAX3421E USB controller IC. This IC can operate in device and host modes, though the Arduino library only supports the host mode. The USB host shield offers a number of interface options for different Arduino types using 3.3V or 5V power supplies. There is also two 8 bit GPIO ports and a library to support an LCD display on this port.

The USB Host shield is described on the Circuits@Home website here:
http://www.circuitsathome.com/category/mcu/usb
Full construction information is provided including schematic and PCB layout, or PCB and built boards are available in the on-line store.

The USB shield uses the following Arduino hardware resources which are then not available for other purposes:

Digital Pin 7 used for MAX_RESET
Digital Pin 8 used for GPX
Digital Pin 9/PWM used for INT
Digital Pin 10/PWM used for SS
Digital Pin 11/PWM used for MOSI
Digital Pin 12 used for MISO
Digital Pin 13/LED used for SCLK

Power the USB host shield is takenr from the Arduino and also powers the USB device. Care must be taken to ensure the total supply current required is available from the Arduino and its power source.

The MAX3421 is documented in Datasheets, programming manual and application notes by Maxim:

http://www.ghielectronics.com/downloads/general/MAX3421E.pdf
http://www.hdl.co.jp/ftpdata/utl-001/AN3785.pdf
http://www.maxim-ic.com/quick_view2.cfm/qv_pk/3639/t/do

Oleg at Circuits@Home has provided Arduino libraries for the MAX3421E function and the USB host function. Small modifications were necessary in these to support the game controllers, mainly in the NAK handling to ensure non blocking behaviour. The changes should not interfere with existing applications using the libraries and may be incorporated to the standard library.
http://github.com/felis/USB_Host_Shield

The library is based on code from Maxim for the ARM processor, but this original code is not complete:
http://www.maxim-ic.com/tools/evkit/index.cfm?EVKit=623

USB is a complex protocol and even though much is done automatically by the MAX3421E, some knowledge of the USB protocol is needed. As primers, these are good sources:

http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf
http://www.beyondlogic.org/usbnutshell/usb1.htm

After that the best references are the USB protocol specifications:

http://www.usb.org/developers/docs/usb_20_052709.zip

http://www.usb.org/developers/hidpage/

6. Bluetooth Dongles

Bluetooth USB dongles come from many suppliers and there is no way to support them all in a simple embedded application. I have used devices which are based on the Cambridge Silicon Radio(CSR) Bluecore 4. These devices are common and low cost at less than $5 each. The dongle is used is based on the BC41B143A chip. The data sheet is here:
http://pdf1.alldatasheet.com/datasheet-pdf/view/114802/ETC/BC41B143A-ANN-E4.html
The dongle conforms to the Bluetooth specification for an HCI interface and is available here:
http://www.bluetooth.com/NR/rdonlyres/1F6469BA-6AE7-42B6-B5A1-65148B9DB238/840/Core_v210_EDR.zip

http://www.bluetooth.com/NR/rdonlyres/0BE438ED-DC1B-41D1-AAC0-1AAA956097A2/980/HID_SPEC_V10.pdf

The dongle used came packaged like this:

A more general Bluetooth protocol tutorial is here:
http://www.palowireless.com/infotooth/tutorial.asp

7. My Development Process and Following Articles

I decide to split the development and these articles into the following steps:

Part 2: Develop the USB interface to the PS3 controller. This allowed me to gain experience of the Arduino and the USB shield. It also provides a quick working wireless solution using the Madcatz game controller.

Part3: Develop the Bluetooth USB and HCI interface used in the support of the Wiimote and PS3 game controller, and also some utilities needed to analyse and configure these devices .

Part 4: Develop the Wiimote support over Bluetooth

Part 5: Develop the PS3 support over Bluetooth