Pduino and other Arduino Interfaces for Pd

Marius Schebella
Brooklyn Polytechnic University
333 Jay Street
Brooklyn, NY 11201
marius.schebella@gmail.com

ABSTRACT

In this paper, I will describe the Pduino interface and some other possibilities to connect Pd and the Arduino board and show examples of usage in different projects.

Arduino is an open-source physical computing input/output (I/O) device used to develop interactive objects and instruments. Pduino[15] (written by Hans-Christoph Steiner) is a Pd interface for the Arduino, consisting of several Pd abstractions and the Arduino firmware called "Firmata"[9].

Keywords

Pd, Arduino, Pduino, Physical Computing, Sensor Interfaces

1. INTRODUCTION

Pd processes data. Data can be received from an outside source and can be sent to an outside source.

Input and output data would be sound, video, human interface data, keyboard commands, mouse events, joystick data, MIDI messages, or network communication data. One class of data is received from and sent to the world of physical computing devices; sensors, electronic units, motors, robots or instruments. There is no existing standard for the communication between physical computing devices and microcontroller interfaces (such as Arduino). Additionally there is no existing standard for communication between microcontroller interfaces and Pd. Input and output data is determined by the input and output devices the user chooses.

2. PHYSICAL COMPUTING

Sensors are the most common input devices. They sense and measure; linear or spatial position, orientation, movement, distance, weight, force, pressure, humidity, temperature, light intensity, air flow, electrostatic charge and much more. Another very common input device would be a switch.

Output devices are all kinds of motors, servos, solenoids, electric machines, robotic devices, instruments or lights. Input and output devices usually are not directly hooked up to the computer but need a small electric circuit including a microcontroller that acts as an interface. The interfaces then talk to the computer via a serial connection like USB Bluetooth, MIDI or a network interface.

Setups with physical computing devices are used in interactive installations or for live performances by musicians, dancers and so on. During the last few years it became more easy for artists to access these technologies and use them for their own purposes. Many new hardware solutions have been developed. The Arduino is the focus of this paper.

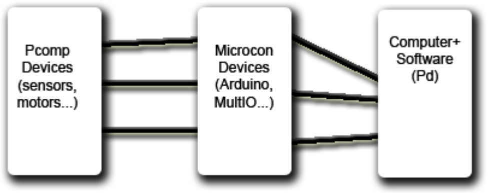
Pcomp Devices (sensors, motors...)
Microcon Devices (Arduino, MultIO...)
Computer+ Software (Pd)

Figure 1: Physical Computing Devices can send/receive serial or analog data, Microcontroller Devices use USB, Bluetooth, MIDI... to communicate with Computer

3. I/O DEVICES

Arduino is one hardware solution for sensor input. Other Microcontroller Devices would be Christian Klippel's MultIO[14], the Create USB Interface[5], STEIM's junXionbox[11] (these devices talk to the USB port), Doepfer[7], Miditron[13] (they talk to the MIDI port) or other devices which use the serial or parallel port, ethernet and Bluetooth. It is easy to build individual interfaces, based on one of the common microcontrollers like PIC, Basic Stamp, Atmel (for example the Arduino on a breadboard[2]).

Pd's first external objects for serial communication appeared between 1998 and 2000. `[serialctl]`[1] by Guenter Geiger, initially written for touchscreen devices, and Winfried Ritsch's initial `[comport]` object, which remains the main object for serial communication.

4. ARDUINO FEATURES

The Arduino board was released around August 2005. It is based on the ATmega8 chip and is usually connected to

[1]a word in square brackets denotes a Pd object

the computer via USB. It has 13 digital pins, which can serve both as in and outputs, 3 of which allow Pulse-width modulation (PWM) output, and 6 analog inputs. The reference designs for Arduino are distributed under a Creative Commons license Attribution-ShareAlike 2.5.

The Arduino soon became widely spread. There is extensive online documentation with instructions for sensors and devices used to measure or control, and there is quite a big internet community (Arduino forum[1]) where people can find support. From the very first Arduino was used as a quasi standard tool for teaching "Physical Computing" at universities or in workshops. Using the Arduino programming language[3] is an easy way to write firmware for the microcontroller. It is an implementation of Wiring, itself built on Processing.

Of course the Arduino board is not the solution for everything, the communication speed of the USB driver is limited by some technical constrictions (see section 6). The number of pins is limited (for example if you want to build interfaces with more controllers the MultIO[14] would be a better choice), but in general Arduino is a great learning tool, easy to use, robust and cheap.

Shortly after its release people started to use it in connection with Pd, writing firmware for their personal needs, one of the first examples was presented in Barcelona (2006) by Alex Posada and David Cuartielles.

In early 2006 the first version of Pduino was released by Hans-Christoph Steiner.

5. CONNECTING THE ARDUINO TO PD

5.1 USB

There are several possibilities to connect the Arduino to Pd. The easiest way is to use the USB port. Here is a very basic example of Arduino code, that will talk to Pd.

```
int switchPin = 2;
 // select the input pin for the switch
int ledPin = 13;
 // select the pin for the LED
int switchState = 0;
 // value data read from the switch
int ledState = 0;
 // variable of the led status

void setup() {
  Serial.begin(115200);
   // start serial communication to Pd
}

void loop() {
  switchState = digitalRead(switchPin);
   // read the state of the switch
  Serial.print(switchState, BYTE);
   // send the value to the serial port (Pd)
  if (Serial.available() > 0) {
       ledState = Serial.read();
        // receive data from Pd
       digitalWrite(ledPin, ledState);
        // turn the led on/off
   }
}
```

The code has to be uploaded to the Arduino board, using the Arduino IDE.

In Pd you create a `[comport]` object with two arguments: the number of your device and the speed of the connection `[comport 6 115200]`. [Fig. 2]

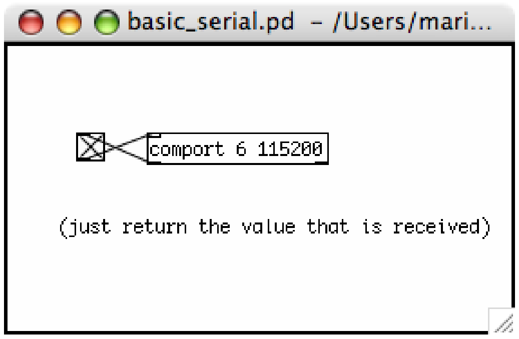
basic_serial.pd – /Users/mari...
comport 6 115200
(just return the value that is received)

Figure 2: Simple Pd connection. Receive data, toggle the checkbox and send the value back

There are two prebuilt systems that you can use, if you don't want to write your own microcontroller code. One is the SMS[18] (Simple Message System) written by Thomas Ouellet Fredericks and the other one is Pduino[15], written by Hans Christoph Steiner. With both packages, you still use the Arduino Environment to upload the firmware to the Arduino. Both solutions make Arduino easily accessible from within Pd. The SMS is an asynchronous serial communication protocol. Pd sends data to the Arduino and the Arduino reacts by returning sensor data.

Pduino is based on the Firmata firmware, which is trying to establish a broader standard of implemention in software packages as; Processing[16], vvvv[20], Python[17], Max/MSP[12] or Flash[10]. It uses a compact MIDI like message format, optimized for high speed data transfer. The Pd part of Pduino is mainly an object called `[arduino]` which depends on some abstractions included in the Pd-extended package. The object reads and spits out messages in an easy to handle message format.

5.2 Wireless

You don't necessarily have to connect the Arduino via USB cable. If you have a Bluetooth Arduino or a Bluetooth extension like BlueSmirf or if you want to use a Zigbee[4] extension for wireless data transmission, the device will show up in the device list of your computer and you can directly use it from within Pd using the `[comport]` object.

5.3 MIDI

It is also possible to assemble the arduino as a MIDI device. Fig. 3 shows the most simple setup for MIDI output[21].

6. SPEED, JITTER AND THE FIRMATA CYCLE

The bottle neck of data exchange between Pd and Arduino is within the serial communication (USB cable, bluetooth, MIDI) and the FTDI driver of the microcontroller [8]. The rule is to encode the information in as few bytes as possible and send bytes in packages that fit the driver size of 62 bytes.

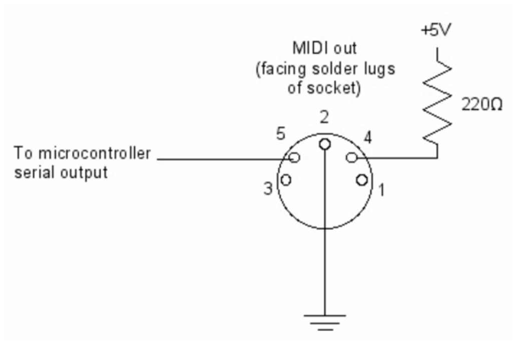

Figure 3: MIDI connections; Arduino pin 1 is connected to pin 5 of the MIDI jack, using a 220 Ohm resistor to connect pin 4 to the power (5V) and pin 3 is connected to the ground. MIDI exchanges data at a baudrate of 31250 bps.

Sensing, encoding and decoding takes much less time. This is more important higher traffic.

The computation on Arduino runs in a loop. The first step is constantly check for changing values at the digital inlets and immediately send them to Pd without delay. The second step is executed every 20 milliseconds. During that call, all data that was received from Pd is handled and the analog inputs are read and sent to Pd. That means messages from Pd to the Arduino and analog data are processed at a constant rate of 50 cycles per second. This can cause jitter of up to 20 milliseconds, but makes sure that even with high traffic, all data will be sent and recieved without delays.

7. PDS ARDUINO OBJECT

Create the `[arduino]` object with an optional argument, the device number of your Arduino. The message `[devices(`[2] will give you a list of available devices. You can select another device by sending the message `[open` ***`devicenumber`*** `(` (where *devicenumber* denotes the number of the corresponding serial devices).

7.1 Receiving Input

In order to receive data from the Arduino you have to 'turn on' the inlets. Digital inlets are enabled all at once by the message `[digitalIns 1(`. Analog inlets are enabled one by one. For example send the message `[analogIns 4 1(` to turn on analog pin 4. Incoming messages are received at the left outlet of the `[arduino]` object.The values of the analog inputs are sent permanently and are in a range between 0 and 1. For example `[analog 4 0.423265(` means pin 4 has a current value of 0.423265. Digital inputs will only be received when they change. The syntax is 'digital *pinnumber value*' (where *pinnumber* is a single number and *value* one of 0 or 1).

7.2 Controlling Outputs

To control outputs you first have to switch the corresponding pin to "output mode". (If the pin was turned on at that moment, automatically a message "digital *pinnumber* 0 is sent.) Do this by sending `[pinMode` ***`pinnumber`*** `1(`. After that you can control the pins by messages like `[digital` ***`pinnumber value`*** `(`. Pins 9-11 also allow pulsewidth modulation (PWM), a method used to produce analog voltage output. Send a message `[pwm` ***`pinnumber value`*** `(` (where *value* is a float from 0 to 1).

By sending the messages `[info(` and `[version(` to the `[arduino]` object you will receive a message 'version *number1 number2*', where *number1* and *number2* are the Firmata version numbers.

[2]words in a square bracket and a parenthesis denote a Pd message

8. SOME PROJECTS

Here are a few examples of art pieces using Pd and Arduino.

8.1 Barbara's Radio and Barbara's Kitchen

These are two installations by Diane Ludin and Marius Schebella using Pd and Arduino. The name is chosen after the American scientist and Nobel Prize winner Barbara McClintock, who studied chromosomes and how they change during reproduction in maize. The first installation, Barbara's Radio plays genome code sequences by using morse code, which is sent over an electromagnetic field of a solenoid into a portable FM Radio and the other installation uses "kitchen equipment" of a chemical laboratory - test tubes which are hit by solenoids to produce rhythmical patterns.

8.2 Deflektor and Solenoid Concert

"Deflektor"[6] is an installation by Tim Vets and Erki De Vries with rotating panels and video projections which was shown at Z33, Hasselt, Belgium. The projection directions are controlled by servo-motor-driven mirrors. The rotation of the panels is also controlled by a Pd patch. Everything runs on a linux system with Puredata, interfacing via 2 Arduino's.

"Solenoid concert"[19] is a piece released on YouTube showing an installation by Roman Haefeli using solenoids taped to windows, heating conduit and office equipment, controlled via a Pd step sequencer.

9. ACKNOWLEDGMENTS

Miller Puckette (Pd), Guenter Geiger (serialctl, Pd externals),Winfried Ritsch (comport, Pd externals), Christian Klippel (MultIO), Hans-Christoph Steiner (Firmata, Pduino), Alex Posada (artist), Thomas Ouellet Fredericks (SMS), Tom Igoe (Author of "Physical Computing"[22], Arduino developer), Massimo Banzi (Arduino developer), David Cuartielles (Arduino developer), Gianluca Martino (Arduino developer), Dave Mellis (Arduino developer), Tim Vets and Erki De Vries (artists), Roman Haefeli (artist), Diane Ludin (artist).

10. REFERENCES

[1] Arduino forum. `http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl`.

[2] Arduino on the breadboard. `http://itp.nyu.edu/physcomp/Tutorials/ArduinoBreadboard`.

[3] Arduino software. `http://www.arduino.cc/en/Main/Software`.

[4] Axic. `http://mrtof.danslchamp.org/AXIC`.

[5] Cui. `http://www.create.ucsb.edu/~dano/CUI`.

[6] Deflektor. http://www.timvets.net/projects/erkiandtim/deflektor/deflektor.html.

[7] Doepfer. http://www.doepfer.de.

[8] Fast ftdi. http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1170939903/11.

[9] Firmata. http://www.arduino.cc/playground/Interfacing/Firmata.

[10] Flash. http://www.adobe.com/products/flash.

[11] Junxionbox manual. http://www.steim.nl/software/junxionbox/junXion%20boX%20manual.pdf.

[12] Max/msp. http://www.cycling74.com.

[13] Miditron. http://www.eroktronix.com.

[14] Multio. http://multio.mamalala.de.

[15] Pduino. http://at.or.at/hans/pd/objects.html.

[16] Processing. http://processing.org.

[17] Python. http://www.python.org.

[18] Sms, simple message system. http://tof.danslchamp.org.

[19] Solenoid concert. http://www.youtube.com/watch?v=g_hiz-Kx0kM.

[20] vvvv. http://vvvv.org.

[21] T. Igoe. Midi communication. http://www.tigoe.net/pcomp/midi.shtml.

[22] D. O'Sullivan and T. Igoe. *Physical Computing*. Course Technology Ptr, 2004.