Pd-L2Ork Raspberry Pi Toolkit as a Comprehensive Arduino Alternative in K-12 and Production Scenarios

Ivica Ico Bukvic, D.M.A.
Virginia Tech
Institute for Creativity, Arts, and Technology
Blacksburg, VA, USA
ico@vt.edu

ABSTRACT

The following paper showcases new integrated Pd-L2Ork system and its K12 educational counterpart running on Raspberry Pi hardware. A collection of new externals and abstractions in conjunction with the Modern Device LOP shield transforms Raspberry Pi into a cost-efficient sensing hub providing Arduino-like connectivity with 10 digital I/O pins (including both software and hardware implementations of pulse width modulation) and 8 analog inputs, while offering a number of integrated features, including audio I/O, USB and Ethernet connectivity and video output.

Keywords

Pd-L2Ork, Raspberry Pi, Toolkit, Arduino Alternative, K-12

1. BACKGROUND

Since its introduction, Raspberry Pi (RPi) [1] platform has gathered a sizable following. Arguably a part of its charm is in that it offers a comprehensive operating system in a form of Linux [2] builds with a variety of preinstalled software packages, such as PdPi [3] and Satellite CCRMA [4], with onboard USB, Ethernet, audio, and video connectivity, as well as general-purpose input/output (GPIO) pins for interfacing with a broad array of equipment and sensors. Even though it is priced competitively to similarly equipped Arduino boards [5], its more widespread adoption in informal learning environments, such as Maker workshops, where it could serve as a viable alternative to the Arduino platform has been conspicuously slow. This slump may be primarily attributed to the lack of legible and easy access to individual GPIO channels and platform's inability to seamlessly interface with analog sensors.

2. RPi SHIELDS TO THE RESCUE

To address the aforesaid deficiency, we've seen an introduction of a number of third party accessories and shields. The most cost-efficient are the bridging interfaces between the GPIO and various breadboard configurations, e.g. Adafruit's prototyping accessories [6]. Intermediate solutions consist primarily of integrated shields whose focus is on easier access to digital pins and/or additional dedicated analog inputs, such as the Modern Device LOP [7]. Some go as far as to provide a comprehensive, albeit costly Arduino-like functionality built on top of RPi's GPIO [8].

The two lingering challenges associated with the aforesaid third party solutions is their inherent cost that often exceeds that of RPi itself, as well as lack of an integrative programming environment that can harness the full potential of RPi's connectivity. The cost predicament once again places such solutions in the dubious realm particularly in respect to providing Arduino functionality. This is not to suggest Arduino solutions need to be supplanted—there are undoubtedly scenarios where RPi's added functionalities and consequently their cost overhead are unnecessary. It appears, despite the ability to utilize such integrative environment in conjunction with built-in onboard A/V functionality through a comprehensive operating system that also supports a variety of programming environments to interface with peripherals, RPi's usability remains hampered by the lack of the aforesaid comprehensive access to digital and analog sensors.

The second challenge of the RPi+shield solutions is ability to seamlessly integrate the aforesaid connectivity through a single, ideally comprehensive development environment. This problem has been in part addressed through projects like Python GPIO module (RPi.GPIO) [9] that provides comprehensive access to GPIO infrastructure but suffers from limited audio functionality of the python environment, and Miller Puckette's RPi port [10] of the ubiquitous Pure-Data real-time visual programming environment [11] that provides only basic sysfs access to GPIO pins. The solution, while providing access to audio and to a lesser extent video, falls short by offering only low speed access to GPIOs via sysfs filesystem [12]. This precludes a high speed GPIO communication consequently preventing access to pulse width modulation (PWM) [13], beyond the single pin with hardware PWM implementation that remains accessible through sysfs. More importantly, the existing implementation provides no simple way of interfacing with analog sensors whose varying voltage requires an analog-to-digital (A/D) converter [14].

It is therefore becoming increasingly apparent that a number of hurdles remain for RPi to be considered a board capable of serving as an Arduino replacement in a broad array of scenarios, with particular focus on interactive multimedia art and installations: the lack of easy access to GPIOs, an ability to interface with analog sensors, a comprehensive integrative software development environment, and an ability to achieve this at a competitive price point.

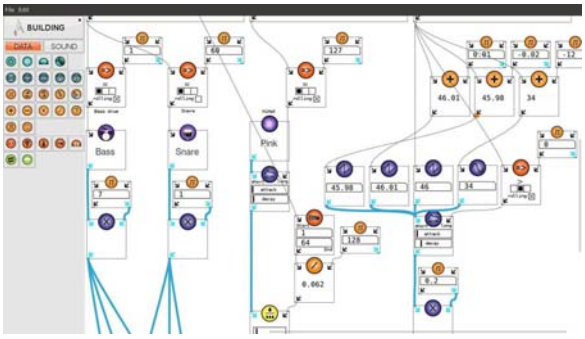

Figure 1. Pd-L2Ork K-12 module demo patch.

3. INTRODUCING PD-L2ORK FOR RPI

Pd-L2Ork [15] is a Pure-Data variant. Originally introduced as the audio-visual foundation for Virginia Tech's Linux Laptop Orchestra (L2Ork) [15][16], it has since grown into a project of its own with a comprehensive set of new features and bug fixes. More recently, it has been extended with a K-12 educational module [17] (Figure 1), enabling its use in a number of informal learning scenarios, including a three-year partnership with the Boys & Girls Club of Southwestern Virginia [17] and two Maker workshops [18]. The philosophy behind the project is the focus on usability and turnkey solutions, particularly when it comes to K-12 learning environments where the fault tolerance threshold is particularly low. This is also reflected in K-12 modules seamless integration of connectivity with various controllers, such as Nintendo Wiimotes [19] and Arduino boards using SARCduino firmware [20]. The latest addition to the Pd-L2Ork ecosystem is the RPi build that, in addition to supporting RPi-Arduino connection, also includes comprehensive access to RPi's GPIOs and with the addition of the MCP3008 A/D converter analog inputs.

3.1 Shield

The Pd-L2Ork implementation relies on a more comprehensive intermediate approach to RPi shields, namely a relatively affordable Modern Device LOP (NOP version) shield (Figure 2), providing easy access to 10 digital I/O pins, as well as 8 analog inputs using integrated MCP3008 A/D converter. Although the same is easily achievable through the use of a breadboard, the attention here is on an integrated approach that shifts focus away from dealing with rudimentary wiring towards more robust error-safe solutions ready for both introductory informal learning environments and real-world production-level implementations. Latter in particular require a level of stability and reliability that is difficult to achieve with prototyping hardware, such as breadboards. The said LOP board offers a few additional convenience features, like the co-located and labeled ground, power, and GPIO pins, increased current throughput of approx. 300mA instead of the usual 30mA (thus allowing for interfacing with elements that require greater power draw, provided RPi is supplied adequate power via a dedicated USB power supply), and perhaps most importantly protection from short circuiting the board due to faulty wiring which is also reflected by a subtle surface-mounted light emitting diode (LED).

Another notable feature of the NOP variant of the LOP shield is ability to immediately access GPIO 4 pins using 4 microswitches on the right. This can be handy in providing a simple access and configuration interface in headless situations, as well as an ability to test the connectivity without any additional sensors. In addition to co-located per-pin access to 3.3V, as is the case with other shields, LOP exposes one auxiliary 5V pin directly off the GPIO connector (the top-left pin on the Figure 2).

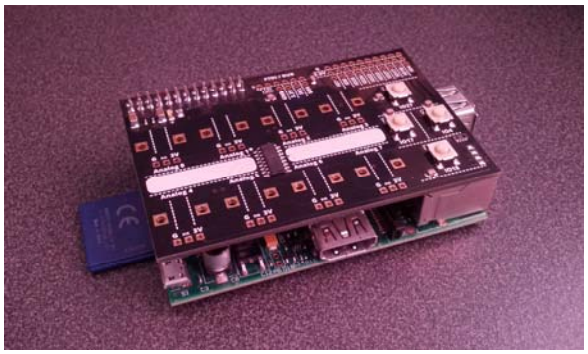

Figure 2. Raspberry Pi with a LOP shield (NOP variant).

3.2 Pd-L2Ork Externals

The software implementation relies upon two externals: disis_gpio and disis_spi. The prefix "disis_" is used to distinguish this implementation from other externals bearing similar name and purpose.

3.2.1 disis_gpio

disis_gpio (Figure 3) external is in part based on the WiringPi library [21], reimplementing PWM code to allow for dynamic instantiation of supporting timing threads. For other aspects of GPIO connectivity, disis_gpio statically links against the WiringPi library to allow for an easily transportable monolithic external. The object aims to capture the basic sysfs style connectivity with individual gpios, using Raspberry Pi's original pin numbering scheme. It is only available if Pd-L2Ork is run with sudo privileges. This is because PWM functionality requires faster access to the device than what sysfs interface offers and is only possible through direct reads and writes to device's physical memory. If Pd-L2Ork is started without sudo privileges, the object will fail to create and will output an error in the console together with a suggested solution. Starting external on a computer without the necessary hardware will also generate a console error and terminate external instantiation. Before being able to use this external, the system needs to load *gpio* driver by issuing the following command:

```
sudo modprobe gpio
```

Alternately, "gpio" can be added on the last line of the /etc/modules system file which will ensure that the driver is automatically loaded when the system boots.

To enable a gpio pin in digital (non-PWM) mode user needs to:

1. export the desired pin (or identify it using optional creation argument)
2. specify pin's data direction (in vs. out; can be changed at any time)
3. open the pin
4. use "bang" or a metro object to retrieve pin value at a desired interval OR send values 0 or 1 to the pin to set its output value

disis_gpio will output connection status through the right inlet and data values through the left inlet (when using input mode). RPi has one hardware PWM-enabled pin (18). Its functionality can be accessed using "togglepwm" message (available only when in output mode) after which sending values between 0 and 1023 using the "pwm" message will determine PWM behavior (1023 being 100% on and 0 being off). The selected ranges are designed to correspond with Arduino's default behavior as per SARCduino's firmware [20] to allow for near-seamless transition between the two devices. This is particularly important as Pd-L2Ork also offers seamless integration with Arduino boards using the aforesaid firmware both in default and K-12 modes.

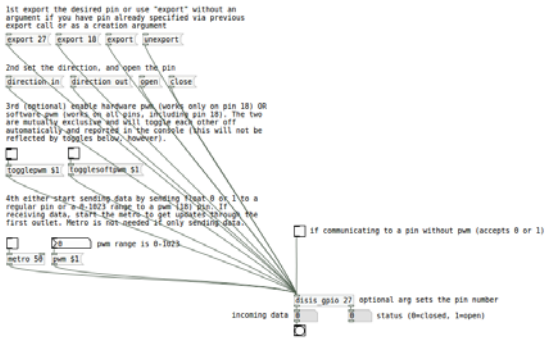

Figure 3. disis_gpio external help file.

The software PWM implementation supports all pins, including the aforesaid pin 18 mainly due to discrepancy between the software and hardware implementation. Latter offers logarithmic scaling of values and consequently a much finer grained control in the lower range that is impossible to replicate in software without requiring an impractically high CPU overhead. The software implementation therefore utilizes a hardwired 1000Hz pulse, providing a reasonably high resolution made possible by spawning a separate high-priority clock thread with direct access to device's memory. In this version, however, LEDs tend to light up almost immediately at a value of 1 and continue to gradually increase in brightness up until the value 1023. Due to the ensuing scaling discrepancy, mixing software and hardware iterations of PWM is not advised. Consequently, all 10 GPIO channels can be used for PWM-based analog sensors, as well as digital I/O.

When closing a pin, one can issue a "close" message, followed by an "unexport" message. Alternately, the same can be accomplished by simply immediately invoking the "unexport" command—the rest will be taken care of automatically by the external. Additional built-in checks ensure that the proper pin closing steps are taken when abruptly deleting object, or in other similarly extreme situations, to ensure that the overall environment remains stable and unaffected by the unusual course of action.

3.2.2 disis_spi

disis_spi (Figure 4) complements disis_gpio by allowing users to easily interface with the LOP's onboard MCP3008 A/D converter. It takes its name from the underlying driver that needs to be loaded before MCP3008's inputs can be registered and accessed. The code borrows from a number of examples found online and consolidates them into a comprehensive Pd-L2Ork external. Like disis_gpio, disis_spi relies upon kernel drivers that are not loaded by default. To load them manually, we need to issue the following commands:

```
sudo modprobe spi-bcm2708
sudo modprobe i2c-bcm2708
```

Alternately, said drivers can be once again added to the /etc/modules system file by listing them one driver name per line to ensure that they are automatically loaded on system boot. It is worth noting that some builds deliberately blacklist aforesaid drivers, preventing them from loading under any conditions. To fix this, simply remove them from the /etc/modprobe.d/raspi-blacklist.conf file.

Given the uniform nature of the incoming data stream, the external is considerably simpler than disis_gpio—the initialization is a simple 2-step process:

1. open default device /dev/spidev0.0 (or specify alternative one)
2. send "bang" or use metro to retrieve current values which are outputted through outlets corresponding to 8 channels/pins

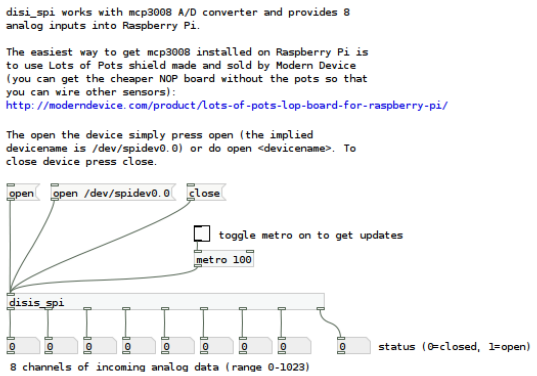

Figure 4. disis_spi external help file.

device simply send a "close" message. Once again, the individual channel data conforms to the aforesaid 0-1023 range.

Consequently, disis_spi when coupled with the MCP3008 A/D converter provides connectivity with up to 8 analog sensors and devices. Taking into an account the additional 10 GPIO-based channels capable of PWM, the ensuing RPi+LOP shield offers a compelling Arduino alternative.

3.3 Applicability

Even though the externals have been conceived primarily with Pd-L2Ork and Modern Device NOP variant of the LOP shield in mind, they are compatible with the vanilla Pure-Data [11] environment and other vanilla-centric forks (e.g. Pd-Extended [22]). More so, both externals should be capable of interfacing with any other breadboard-based approach to accessing either GPIO or inputs from the MCP3008 A/D converter.

4. TESTS AND CONCLUSIONS

Each software PWM thread with runtime updates generates around 2% CPU overhead on a Rev. 2 RPi board suggesting a maximum of 20% of CPU for 10 software PWM instances running concurrently. The actual overhead will vary depending on the rate of change.

While the RPi+shield combo is relatively affordable (est. $35 and $23, respectively), it is still almost double the price of an Arduino Uno (est. $30). Hence, despite the added sensor connectivity (10 PWM-enabled GPIO channels and 8 analog channels), the newfound gadget's true potential will be likely in scenarios that call for access to a comprehensive operating system, specific toolkits (e.g. Pd-L2Ork), and/or RPi's added connectivity (e.g. audio I/O and video). Apart from unique advantages, such as the aforesaid comprehensive operating system and a relatively broad choice of development tools that in certain scenarios may prove sufficient in rationalizing the cost difference, in projects requiring direct access to audio and video, the Arduino price advantage quickly dissolves due to the cost of the supporting accessories needed to provide proper audio (e.g. Wave shield [23]; est. $22), and/or video I/O (e.g. Video Game Shield [24]; est. $23).

5. REAL-WORLD IMPLEMENTATIONS

The newfound environment served as the backbone of Virginia Tech Institute for Creativity, Arts, and Technology (ICAT) Maker workshop that took place in March 2014. In the summer 2014, the same will also play a role in our annual multisite K-12 Makkr camp. Finally, the newfound infrastructure will also serve as a foundation for a commissioned communal arts installation "Cloud" (in collaboration with an Artist and Architect Aki Ishida) set to premiere in Ballston district, Virginia, near Washington D.C. Utilizing 50 RPis clothed in weather-resistant vessels and equipped with audio I/O (capture and playback), and light I/O (light/color sensor and LED array respectively), these artifacts will be in part fashioned and programmed by community participants using a K-12 variant that promotes rapid prototyping with minimal learning curve, ultimately generating a meta-cloud where each individual cloudlet interacts with each other through sound and light in most unsuspecting ways.

6. FUTURE WORK

The two externals have reached a level of maturity and have been integrated into a growing library of K12 abstractions that already offer connectivity with Nintendo Wiimotes and their peripherals [19], as well as SARCduino implementation of Arduino [16,18]. While the preliminary tests of the newfound current bandwidth made possible by the LOP shield suggest the board can easily power four 80mA LEDs, additional tests are warranted to determine how many sensors and emitters can RPi power stably and reliably over a longer span of time. Finally, a future iteration of disis_gpio may also focus on implementing the last remaining component—the one-wire-bus protocol [25].

7. OBTAINING SOFTWARE

Both externals come prepackaged with Raspberry Pi builds of Pd-L2Ork, together with supporting documentation. Users can access externals in the default mode, as well as through a collection of K12 abstractions and their supporting documentation designed to provide turnkey access to said functionality (Figure 5). Prebuilt deb packages for the PdPi distribution, and a detailed documentation on how to set up the system can be found on L2Ork's website at http://l2ork.music.vt.edu/main/?page_id=2288. The source can be retrieved from Pd-L2Ork's git page [26].

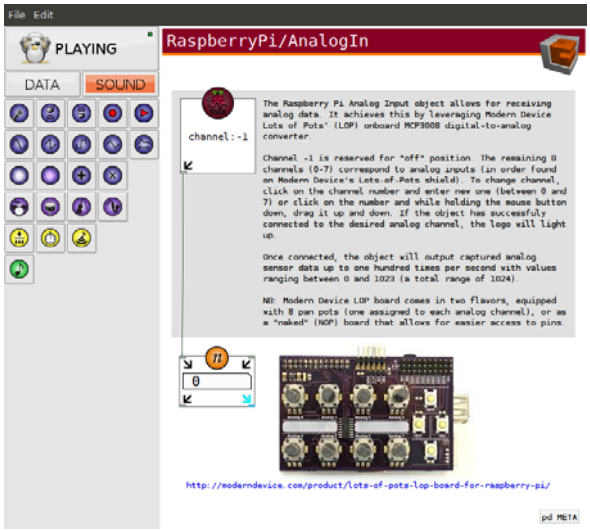

Figure 5. Pd-L2Ork K12 mode RPi analog input object.

8. ACKNOWLEDGMENTS

The author wishes to acknowledge Virginia Tech Institute for Creativity, Arts, and Technology for its support of this project, Newblankets Inc. non-profit organization for sponsoring Raspberry Pi hardware, and Ballston Business Improvement District non-profit organization for a commission that inspired this project.

9. REFERENCES

[1] E. Upton and G. Halfacree, *Raspberry Pi User Guide*. John Wiley & Sons, 2013.

[2] J. Sanders, "Linux, open source, and software's future," *IEEE Softw.*, vol. 15, no. 5, pp. 88–91, 1998.

[3] "PdPi," *pd-la*. .

[4] E. Berdahl and W. Ju, "Satellite CCRMA: A musical interaction and sound synthesis platform," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2011, pp. 173–178.

[5] M. Banzi, *Getting started with Arduino*. Beijing; Cambridge: Make:Books / O'Reilly, 2009.

[6] "Prototyping : Adafruit Industries, Unique & fun DIY electronics and kits." [Online]. Available: http://www.adafruit.com/category/105_163. [Accessed: 26-Jan-2014].

[7] "Lots of Pots (LOP) Board for Raspberry Pi," *Modern Device*. [Online]. Available: http://moderndevice.com/product/lots-of-pots-lop-board-for-raspberry-pi/. [Accessed: 26-Jan-2014].

[8] "Raspberry Pi to Arduino Shields Connection Bridge." [Online]. Available: http://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge. [Accessed: 26-Jan-2014].

[9] "raspberry-gpio-python," *SourceForge*. [Online]. Available: http://sourceforge.net/projects/raspberry-gpio-python/. [Accessed: 29-Jan-2014].

[10] M. Puckette, "Software by Miller Puckette." [Online]. Available: http://msp.ucsd.edu/software.html. [Accessed: 26-Jan-2014].

[11] M. Puckette, "Pure Data: another integrated computer music environment," *Proc. Int. Comput. MUSIC Conf.*, pp. 37–41, 1996.

[12] P. Mochel, "The sysfs filesystem," in *Linux Symposium*, 2005, p. 313.

[13] M. Barr, "Pulse width modulation," *Embed. Syst. Program.*, vol. 14, no. 10, pp. 103–104, 2001.

[14] R. H. Walden, "Analog-to-digital converter survey and analysis," *Sel. Areas Commun. IEEE J. On*, vol. 17, no. 4, pp. 539–550, 1999.

[15] I. I. Bukvic, T. Martin, E. Standley, and M. Matthews, "L2Ork : Linux Laptop Orchestra," in *New Interfaces for Music Expression*, Sydney, Australia, 2010, pp. 170–173.

[16] I. Bukvic, "A Behind-the-Scenes Peek at World's First Linux-Based Laptop Orchestra – The Design of L2Ork Infrastructure and Lessons Learned," presented at the Linux Audio Conference, Stanford, California, 2012, pp. 55–60.

[17] I. I. Bukvic, L. Baum, B. Layman, and K. Woodard, "Granular Learning Objects for Instrument Design and Collaborative Performance in K-12 Education," in *New Interfaces for Music Expression*, Ann Arbor, Michigan, 2012, pp. 344–346.

[18] B. Sawyer, J. Forsyth, T. O'Connor, B. Bortz, T. Finn, L. Baum, I. I. Bukvic, B. Knapp, and D. Webster, "Form, function and performances in a musical instrument MAKErs camp," in *Proceeding of the 44th ACM technical symposium on Computer science education*, New York, NY, USA, 2013, pp. 669–674.

[19] C. Kiefer, N. Collins, and G. Fitzpatrick, "Evaluating the wiimote as a musical controller," in *Proceedings of the 2008 International Computer Music Conference (ICMC)*, 2008.

[20] "SARCduino » MuSE." [Online]. Available: http://www.musicsensorsemotion.com/2010/03/08/sarcduino/. [Accessed: 30-Jan-2014].

[21] "WiringPi." [Online]. Available: http://wiringpi.com/. [Accessed: 29-Jan-2014].

[22] "Pd-extended — PD Community Site." [Online]. Available: http://puredata.info/community/projects/software/pd-extended/?searchterm=pd-extended. [Accessed: 08-Feb-2012].

[23] "Adafruit Wave Shield for Arduino Kit [v1.1] ID: 94 - $22.00 : Adafruit Industries, Unique & fun DIY electronics and kits." [Online]. Available: http://www.adafruit.com/products/94. [Accessed: 30-Jan-2014].

[24] "Video Game Shield Kit ID: 311 - $22.50 : Adafruit Industries, Unique & fun DIY electronics and kits." [Online]. Available: http://www.adafruit.com/products/311. [Accessed: 30-Jan-2014].

[25] R. D. Lee, "Command/data transfer protocol for one-wire-bus architecture," US5809518 A15-Sep-1998.

[26] "pd-l2ork (Pd-L2Ork)," *GitHub*. [Online]. Available: https://github.com/pd-l2ork. [Accessed: 29-Apr-2014].