SPINE: A TUI Toolkit and Physical Computing Hybrid

Aristotelis Hadjakos
Center of Music and Film Informatics
HfM Detmold / HS OWL
Hornsche Str. 44
32756 Detmold, Germany
hadjakos@hfm-detmold.de

Simon Waloschek
Center of Music and Film Informatics
HfM Detmold / HS OWL
Hornsche Str. 44
32756 Detmold, Germany
waloschek@hfm-detmold.de

ABSTRACT

Physical computing platforms such as the Arduino have significantly simplified developing physical musical interfaces. However, those platforms typically target everyday programmers rather than composers and media artists. On the other hand, tangible user interface (TUI) toolkits, which provide an integrated, easy-to-use solution only support a limited set of compatible sensors. We propose a concept that hybridizes physical computing and TUI toolkit approaches. This helps to tackle typical TUI toolkit weaknesses, namely quick sensor obsolescence and limited choices. We developed a physical realization based on the idea of "universal pins", which can be configured to perform a variety of duties, making it possible to connect different sensor breakouts and modules. We evaluated our prototype by making performance measurements and conducting a user study demonstrating the feasibility of our approach.

Keywords

TUI Toolkits, Physical Computing, sensors, Arduino, Max/MSP, SuperCollider

1. INTRODUCTION

Computer-based sound synthesis has greatly enriched the possibilities that are available to today's composers and musicians. In contrast to traditional instruments, the sound is not generated by direct physical interaction with the instrument but by comparatively abstract computational processes. Although renderings of acousmatic music in concert halls with an empty stage are not uncommon, there is a strong desire to include an element of performance and interactivity. One way to do this is to let performers control the synthesis processes on stage with a physical interface. The entry barrier for developing such physical musical interfaces has been significantly lowered by the advent of physical computing platforms such as the Arduino.

Physical Computing: While the Arduino has empowered everyday programmers to develop physical interfaces, there is still a considerable gap to be bridged before reaching out to everyday composers and media artists. While they may be proficient in graphical programming languages such as Max/MSP, they usually lack the text-based programming background that is necessary in order to use the Arduino platform successfully.

NIME'14, June 30 – July 03, 2014, Goldsmiths, University of London, UK.

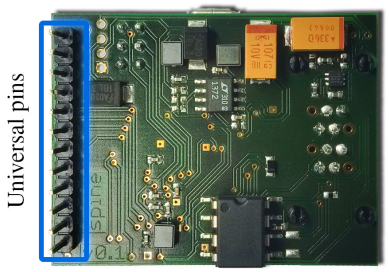


Figure 1: The SPINE with its 12 universal pins. The pins are spaced the standard 0.1" (2.54 mm) apart and can be configured to perform a variety of duties, thus making it possible to connect a large variety of sensor breakouts and modules.

TUI toolkits: TUI toolkits are non-programmable, integrated sensor solutions. They are typically composed of a master (or measurement) unit to which the user may connect a variety of sensor units. Both master and sensor units are provided by the same vendor, making it simple to integrate everything into an easy-to-use product. TUI toolkits are therefore well suited for inexperienced users, such as everyday composers and media artists. A drawback of TUI toolkits is that the users are constrained to a limited set of compatible sensors provided by a single vendor.

Contribution: We propose a concept for bridging physical computing and TUI toolkit communities, connecting everyday programmers and composers. The SPINE (see Figure 1) is our physical realization of a platform for that concept. It is based on the observation that today there are various manufacturers that produce breakout boards even for the latest sensor technologies available on the market. Most of those boards have an identical connector spacing of 0.1" (2.54 mm). While the physical pin layout is de facto standardized, little else is. With our idea of "universal pins", it is possible to connect a large variety of breakout boards to the SPINE. Our approach might help to alleviate the problems connected to current TUI toolkit solutions (especially quick obsolescence of sensors and limited choices) and even benefit experienced firmware developers by encouraging a higher degree of code reuse.

2. RELATED WORK

Related work generally falls into two categories: physical computing platforms and TUI toolkits. The SPINE, however, includes both those facets in one system.

2.1 Physical Computing Platforms

Physical computing platforms make hardware prototyping accessible to everyday programmers. To accomplish this, these platforms provide IDEs, standardized libraries, and

high-level language support that allow the programming experience to resemble "ordinary" application programming. The Arduino platform [13] is currently among the most popular physical computing platforms. It was preceded by the rather similar Wiring platform [2].

The popularity of the Arduino has led to a multitude of shields that can be attached to an Arduino and provide additional functionality (e.g., WiFi, connections to sensors and actuators, etc.). Furthermore, the Fritzing project has greatly reduced the "prototyping barrier" [10] and has simplified the process of designing such shields oneself [10]. There are specialized shield-and-connector solutions that allow connecting various devices to the Arduino. One example is the GROVE by Seeedstudio [15]. It features a significant number of modules that can be easily connected to an Arduino. The CUI32Stem, a physical computing platform based on the PIC microcontroller, also provides support for the GROVE prototyping system [14].

The Raspberry Pi is an ARM-processor based, full-fledged Linux system that can be used as a physical computing platform [16]. The Satellite software distribution has made the Raspberry Pi more accessible for musical physical computing applications, providing special support for interactive music creation, graphics, and media playback [3].

2.2 TUI Toolkits

TUI toolkits hide the low-level programming tasks completely, providing an easy-to-use black box implementation. Most systems are modular, composed of a master unit and several modules that can be attached to it. E.g., in the Phidgets system, most sensors are provided as modules, which output an analog voltage [7, 12]. This voltage is measured and converted to digital by the master unit while some advanced sensors are provided as standalone solutions [7, 12]. A similar approach is used in the d.tools system [8]. Here, the input/output units are all equipped with separate microcontrollers, communicating with the master via I^2C [8]. An overview of TUI systems and sensor interface platforms that have been used in the musical domain are listed in Table 1.

Table 1: Overview of TUI toolkits and sensor interface platforms used for musical applications

System	Description
Phidgets [7, 12]	Large collection of sensors and actuators, commercially available
TOASTER and KROONDE [5]	A/D conversion and transmission over WiFi
d.tools [8]	Communication via I^2C to input/output units
Hollinger & Wanderley [9]	Sensing of optical signals
Eobody [6]	Support of many different music-related protocols MIDI, DMX, etc.

3. CONCEPT

Today there are many companies that produce and/or sell breakout boards and modules (Adafruit, Seeed, Sparkfun, etc.). Most of these boards can easily be connected to pins spaced of 0.1" (2.54 mm) apart, which is the same spacing used on prototyping breadboards all around the world. Based on the idea of universal pins (see Section 4), these modules can be connected to a common platform: the SPINE. In TUI toolkits, the vendor provides both the master (or measurement) unit and the sensor units. In our concept, the master unit is the SPINE and the sensor units are the sensor breakouts and modules, manufactured not by a single vendor but by different vendors all over the world. This avoids the vendor lock-in that is present in today's

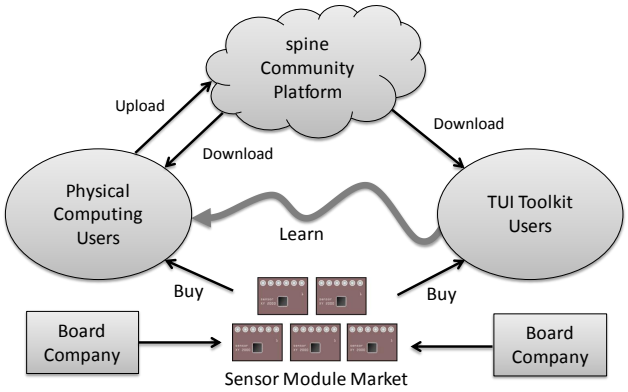


Figure 2: An overview of the concept

TUI toolkits. Due to the dynamic nature of the breakout board and sensor module market, obsolescence is not an issue when using the SPINE: Users will usually be able to find a recent version of their desired sensor type. Furthermore, the users are not restricted by the limited choices that the TUI toolkit vendor provides but can choose their preferred sensor freely.

Figure 2 provides an overview of the concept. TUI toolkit users and physical computing users are both using the SPINE. The TUI toolkit users buy sensor modules from various board manufacturing companies. They then download the documentation, software, and firmware from the (future) SPINE community platform and they are ready to go. Physical computing users may do the same with their SPINE (thus reusing someone else's code) but may also decide to write a new firmware and upload it to the SPINE community platform. Furthermore, since the SPINE can both be used as a TUI toolkit and a physical computing platform, there is a "natural" learning path connecting the two communities of users.

In the following we present two use cases: one where the SPINE is used as a TUI toolkit and another one where the SPINE is used as a physical computing platform.

3.1 SPINE as a TUI Toolkit

Alice is a modern music composer. For her live concerts, she likes to work with graphical programming environments such as Max/MSP. In her most recent project, Alice wants to use inertial movement sensing to control her sounds. She owns a SPINE, which can connect a large variety of sensor breakout boards to her programming environment. Scrolling through a list of supported boards, she finds an appropriate one and orders it from the manufacturer.

A few days later the breakout board arrives. Alice solders a small connector to her new sensor and attaches the device to the SPINE. After uploading the provided firmware to the SPINE with just a few clicks, a Max/MSP object immediately lets her use the sensor values.

3.2 SPINE as a Physical Computing Platform

Bob wants to use a particular sensor breakout board for his project. Unfortunately, this breakout board is currently not supported by the SPINE. Being experienced with the Arduino platform, Bob decides to write his own firmware. Bob likes the idea of an open community and submits the source code of his firmware to the SPINE community platform. Next time anyone wants to use this specific sensor, no firmware programming is necessary.

4. UNIVERSAL PINS

While the physical connector layout of sensor breakouts and modules is de facto standardized (0.1" spacing), little else

is. Breakout boards vary in the number of connectors, the function of each connector, which ones are used for power supply, what voltage they require, which ones provide analog output (if any), which ones are used for serial communication (I^2C, SPI, UART) with the sensor, etc. To ensure wide compatibility with sensor breakouts we need each of the 12 pins of the SPINE to be universal. A pin is "universal" if it can be configured to do any of the following things:

- It can provide supply voltage (VCC) or a connection to ground (GND).
- It can do analog/digital conversion.
- It can assume the role of any line in the most common microcontroller/sensor communication protocols: I^2C (SDA and SCL lines), SPI (MOSI, MISO, SCLK and SS lines), and UART (RX and TX lines).

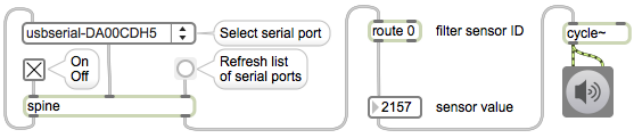
Configurable Power Supply: The microcontroller on the SPINE (the ATmega644PA) can switch each of the 12 pins to VCC or GND. Furthermore, we have provided a mechanism for switching between 5 and 3.3 V on the entire SPINE circuit for compatibility with 3.3 or 5 V breakout boards. The user controls this by closing or opening a connection through a jumper. 3.3 respectively 5 V are provided by the LT1372 configurable DC-DC converter [11].

A/D Conversion: The ATmega644P microcontroller offers 8 analog-to-digital converters inputs. Therefore, an additional multiplexer is used to enable A/D conversion on all 12 pins. The multiplexer configuration is handled by the firmware core library in the background so the programmer is provided with usual Arduino syntax. Short switching times allow measuring analog values on all 12 SPINE pins almost simultaneously.

Bit Banging: To provide I^2C, SPI, and UART connectivity on any pin, we implemented (bit banged) the protocols in software. Based on this, any pin can assume the function of any line in those protocols. E.g., one can configure a certain pin to take to role of SDA in the I^2C protocol. To implement this efficiently, we needed to make sure that the microcontroller could receive logic level change interrupts on every pin. The ATmega644PA microcontroller provides such interrupts on each input. Our performance measurements show that handling such communication in software is unproblematic (see Section 6.1).

5. LANGUAGE INTEGRATION

The SPINE transmits data in packages. Each package contains the actual sensor value and a sensor stream ID (e.g., identifying the x-axis signal of a 3D accelerometer). Both values are integer numbers and are coded in a binary data format, which also unambiguously defines message boundaries in the byte stream that is transmitted over a serial interface (virtual RS-232 over USB).




```
~spine = Spine.new("/dev/tty.whatever");
~spine.register({
     arg ssensorID, data;
     if (sensorID == 3, {
          data.postln;
     });
});
```

Figure 3: Example for the integration of the SPINE with Max/MSP (top) and SuperCollider (bottom)

Arduino: For firmware programmers, the SPINE offers an Arduino-style programming experience. The user installs SPINE-specific extensions, which implement the universal pin functionality (see Section 4) as well as the encoding and serial communication functionality (see above). After the installation, the SPINE can be programmed with the Arduino IDE and language.

Max/MSP & SuperCollider: TUI toolkit users can access sensor values in Max/MSP in an easy manner without text-based programming skills (see Figure 3, top). An interface class for SuperCollider is provided as well (see Figure 3, bottom), making the SPINE utilizable for the two most commonly used musical programming environments.

6. EVALUATION

To evaluate the feasibility of our approach, we measured the performance that can be obtained with our bit banging approach (see Section 4) and performed a user study to test whether the SPINE would be usable as a TUI toolkit. Since the SPINE is a full-fledged Arduino programmed with the well-tested Arduino IDE and language, a quantitative evaluation of the SPINE's physical computing facet was deemed unnecessary.

6.1 Performance

We used a typical SPINE setup, attaching a breakout board that features the InvenSense MPU-9150 inertial sensor. The SPINE was programmed to ask the sensor chip continuously for new values via I^2C and forward all values to the computer. We measured the throughput in two conditions: (1) normal operation, where the actual values were read via software-I^2C from the sensors and forwarded, and (2) max. serial operation, where an arbitrary string was repeatedly sent out. In condition 2, we measured a throughput of 11,561 bytes/s (theoretical value = 11520 bytes/s).[1] There was no measurable difference between condition 2 and condition 1, showing that the RS-232 communication is the bottleneck and that the assembler-based implementation of software-I^2C is very efficient. In the experiment sensor values were packed in 4-byte sized messages, making it possible to transmit a sensor stream with a rate of 2890 Hz.

6.2 User Study

Design of the study: To evaluate whether the SPINE is usable as TUI toolkit, a user study was performed with six students (4m, 2f) of our university. To assess their level of experience in related areas, we prepared a questionnaire where the users would rate their own experience on a scale ranging from "no knowledge" (0 points) to "expert" (6 points) (see Figure 4, top).

The users had to perform the following tasks: install a driver to enable serial communication, install the SPINE software, upload the correct firmware to the SPINE using a provided GUI tool, attach the breakout board to the SPINE, open an example patch, and modify this patch so that it uses sensor data to control the frequency of a sine generator. The documentation provided consisted of general SPINE documentation and specific documentation for 10 different breakout boards. After the task, which took about 30 minutes, the participants filled out the System Usability Scale

[1]The 0.3% discrepancy between the theoretical and the actual value can be explained by inaccuracies in generating the baud rate signal from the crystal frequency.

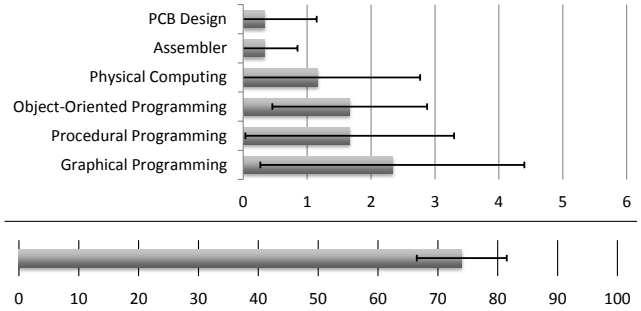


Figure 4: Experience levels of the users (top) and the overall SUS score (bottom) (avg. & std.-dev.)

(SUS) questionnaire [4]. Empirical evaluation has shown that the SUS is a "highly robust and versatile tool" [1] for usability testing. The users were encouraged to bring their own computers to have a more realistic experience.

Quantitative results:[2] The results of our questionnaire are shown in Figure 4. Most users had some background in programming but in general the self-reported programming experience was relatively low, with the averages ranging from 0 to 2.3 on a 7-point scale ranging from 0 to 6. This corresponds well with the intended user group for the SPINE, which are users with some previous experience but lacking the skills for full-scale hardware hacking. The average of the user SUS score ratings is 74. According to the interpretation by Bangor et al. [1], this makes the SPINE a system with "good usability".

Interview results: In general the users were enthusiastic about the system. Suggestions for improvement were due to (sometimes significant) usability problems that were caused by the use of RS-232 via USB. This is a very common design, which is also used throughout the Arduino platform [13]. However, our target users were very uncomfortable using serial interfaces. Problems were the cryptic naming of the serial ports and communication problems that occurred when opening the same port more than once.

7. CONCLUSION AND FUTURE WORK

The SPINE is a hybrid between a physical computing platform and a TUI toolkit. Its 12 universal pins make it possible to connect a large variety of sensor breakouts and modules as long as they can be physically mounted. Since the connectors to most breakout boards and sensor modules are spaced 0.1" (2.54 mm) apart, this is usually not an issue. The flexibility of the SPINE allows the user to connect to a multitude of devices. The SPINE can even "cannibalize" sensor modules from existing TUI toolkits (such as a large portion of the sensor units from Phidgets [7, 12]) and connector solutions (such as a large portion of the GROVE sensor units [15]). For the physical computing toolkit user, the SPINE is usable as a full-fledged Arduino system, making it possible to use the Arduino IDE and language.

SPINE users do not experience vendor lock-in present in other TUI toolkit solutions. Therefore, sensor obsolescence is not an issue and the user is not limited to the relatively few choices that the vendor provides. The advantage for the physical computing user is that it gets easier to reuse existing firmware code since the code does not depend on a particular wiring of the sensor component with the SPINE, just sticking it to the SPINE in the SPINE-standard way is sufficient to get the breakout up and running.

We plan to make the SPINE a HID in order to improve its usability (see Section 6.2). This would also remove the driver installation step since HIDs are natively supported in modern operating systems. Furthermore, we want to provide XBee-based wireless transmission support to enable co-located interactive performances, e.g., a performance with dancers, wearable sensors, and live electronics.

[2]In one instance a technical problem, presumedly a failed serial driver installation, made it impossible for the user to connect to the SPINE and use the system. Thus we report quantitative results only for the five other users.

8. REFERENCES

[1] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[2] H. Barragán. Wiring: Prototyping physical interaction design. Master's thesis, Interaction Design Institute Ivrea, 2003.

[3] E. Berdahl, S. Salazar, and M. Borins. Embedded networking and hardware-accelerated graphics with Satellite CCRMA. In *NIME*, 2013.

[4] J. Brooke. SUS: A quick and dirty usability scale. *Usability evaluation in industry*, vol. 189, 1996.

[5] T. Coduys, C. Henry, and A. Cont. TOASTER and KROONDE: high-resolution and high-speed real-time sensor interfaces. In *NIME*, 2004.

[6] E. Gallin and M. Sirguy. Eobody3: a ready-to-use pre-mapped and multi-protocol sensor interface. In *NIME*, 2011.

[7] S. Greenberg and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Symposium on User interface software and technology (UIST)*, 2001.

[8] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Symposium on user interface software and technology (UIST)*, 2006.

[9] A. Hollinger and M. M. Wanderley. Optoelectronic acquisition and control board for musical applications. In *NIME*, 2012.

[10] A. Knörig, R. Wettach, and J. Cohen. Fritzing: A tool for advancing electronic prototyping for designers. In *Conference on Tangible, Embedded and Embodied Interaction (TEI)*, 2009.

[11] Linear Technology. LT1372/LT1377 500 kHz and 1 MHz high efficiency 1.5 A switching regulators, 1995.

[12] Phidgets Inc. Phidgets Inc. – unique and easy to use USB interfaces. `http://www.phidgets.com`, 2014. [Online; accessed 03-February-2014].

[13] D. Mellis, M. Banzi, D. Cuartielles, and T. Igoe. Arduino: An open electronic prototyping platform. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2007.

[14] D. Overholt. Musical interaction design with the CUI32Stem: wireless options and the GROVE system for prototyping new interfaces. In *NIME*, 2012.

[15] SeeedStudio. Grove system – wiki. `http://www.seeedstudio.com/wiki/GROVE_System`, 2014. [Online; accessed 03-February-2014].

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