Efficient Acquisition of Force Data in Interactive Shoe Designs

Marco Civolani[1], Federico Fontana[2], and Stefano Papetti[1]

[1] Università di Verona, Department of Computer Science
strada Le Grazie, 15 – 37134 Verona, Italy
{marco.civolani,stefano.papetti}@univr.it
[2] Università di Udine, Department of Mathematics and Computer Science
via delle Scienze, 206 – 33100 Udine, Italy
federico.fontana@uniud.it

Abstract. A four-channel sensing system is proposed for the capture of force data from the feet during walking tasks. Developed for an instrumented shoe design prototype, the system solves general issues of latency of the response, accuracy of the data, and robustness of the transmission of digital signals to the host computer. Such issues are often left partially unanswered by solutions for which compactness, accessibility and cost are taken into primary consideration. By adopting widely used force sensing (Interlink) and analog-to-digital conversion and pre-processing (Arduino) components, the proposed system is expected to raise interest among interaction designers of interfaces, in which the reliable and sufficiently broadband acquisition of force signals is desired.

Keywords: Force sensing, closed-loop interfaces.

1 Introduction

Assessment of human gait has been a well known issue in biomechanics and biomedical engineering. Early experiments in measuring and evaluating forces under the foot date back to the end of the 19th century (Beely, 1882; Momburg, 1908). Since that time, various techniques and methods have been implemented with the same purpose, involving the use of different kinds of floor-based electromechanic transducers [1,2,3].

Within this specific research topic, we are currently developing an instrumented shoe design that virtually reproduces ground surfaces, by interactively augmenting otherwise neutral (i.e., flat and homogeneous) floors [16]. By means of appropriate force sensing, real time signal processing, and final displaying of multimodal (audio and tactile) cues through portable loudspeakers and haptic actuators underfoot, this design is expected to serve purposes such as navigation in functional spaces, support to physical rehabilitation, and entertainment.

While planning the design of a sensing system for such shoes, we wanted to reach an acceptable trade off between accuracy of the recorded data and accessibility and cost of the technology. Furthermore, the same system had to deliver a

R. Nordahl et al. (Eds.): HAID 2010, LNCS 6306, pp. 129–138, 2010.

continuous data flow affording tight close-loop interaction with walkers through the real time processing of such data. Recently, small and light force sensing devices have been developed, allowing for within the shoe, online force measurements. In parallel, the exploitation of powerful integrated electronic devices has led to the design of portable and wearable data acquisition systems [4,5,6,7].

Unfortunately, there is a general lack of simple yet reliable wearable force measurement systems. It is true that biomechanical and biomedical researchers and engineers can choose commercial solutions providing integrated hardware and software platforms for accurate underfoot force measurements and gait analysis (see for example the Tekscan product line). Besides their quality, hardly these solutions can be adapted for prototyping novel, flexible concepts of foot-floor interfaces. Furthermore, integrated technologies like these do not always guarantee low-level accessibility to raw data, nor they specify exact figures of latency for the transmission of the related signals.

For these reasons, we have worked on the in-depth optimization of an architecture based on popular hardware in the interaction design field, made by Interlink[1] and Arduino[2] and overall costing few tens of dollars. Its performance depends on the number and characteristics of the sensors and the specifications of the acquisition board: together, they set the dynamic range and the band of the acquired information. Besides its applicability to instrumented shoes, the same architecture is of potential interest in all situations where accurate force data must be acquired and processed in real time.

This paper describes in detail its design and operation inside our prototype.

2 Sensors

Two Force Sensing Resistors (FSR) have been inserted between a sandal and a removable sole, forming an additional layer on top of the sandal itself (Fig. 1). Interlink equipment was chosen, for its versatility and popularity among interaction designers [8]. In this sort of "sandwich" configuration, the two sensors detect forces respectively in correspondence of the toe and the heel.

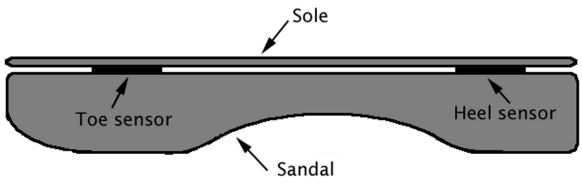


Fig. 1. Mechanical assembly of the shoe. For both toe and heel the Interlink FSR model 402 has been used.

[1] http://www.interlinkelectronics.com/
[2] http://arduino.cc/

Each FSR is connected in series with a fixed resistor. Together, they realize a voltage divider. The four dividers, two for each shoe, are finally connected to the first four analog inputs of an Arduino Duemilanove board, another quite popular device. Fig. 2 illustrates this connection.

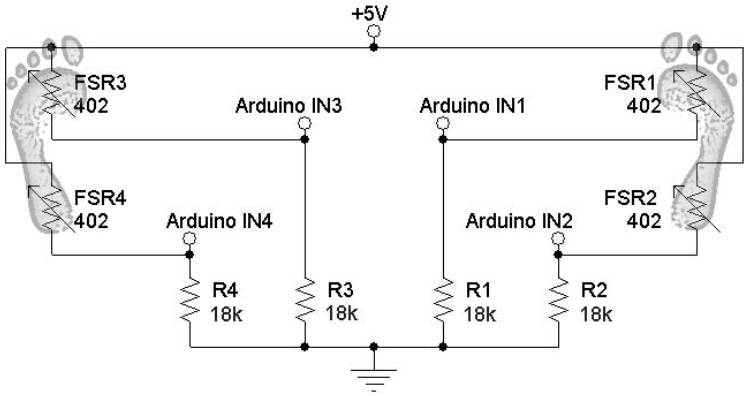


Fig. 2. Schematic of the conditioning circuit. The 5 V and ground pin are accessible from the Arduino board.

A previous version of the prototype used different sensor models and resistance values in the voltage divider [9]. In the following we will see that optimizing these two components leads to substantial improvements.

2.1 Characterization of the FSR

Interaction designers know that Interlink force sensors have good response in front of rapid and large changes in the applied force [10], and they are also especially robust [11]. Thus, they are suitable for detecting forces such as those typically exerted by the foot during gait. For their small size, the model 400 and model 402 are the only Interlink products that can find place inside a shoe without risk of breaking due to excessive mechanical deformation. Although quite similar, the model 400 overall exhibits a different behavior compared to the 402 [12]: in our previous prototype, a good balance between front and rear sensitivity had been achieved by placing a model 400 under the heel and a 402 under the toe.

While changing shoe model during the development of the current prototype, we found that the model 400 reaches mechanical saturation much faster than the 402. This overall unbalanced the acquisition of the data. Clearly, the foot pressure had a different distribution in the new sole.

Measurements have then been made to characterize their behavior. The same measurements furthermore proved useful to characterize the nonlinear behavior of the overall system, caused by the electrical coupling between the FSR and the

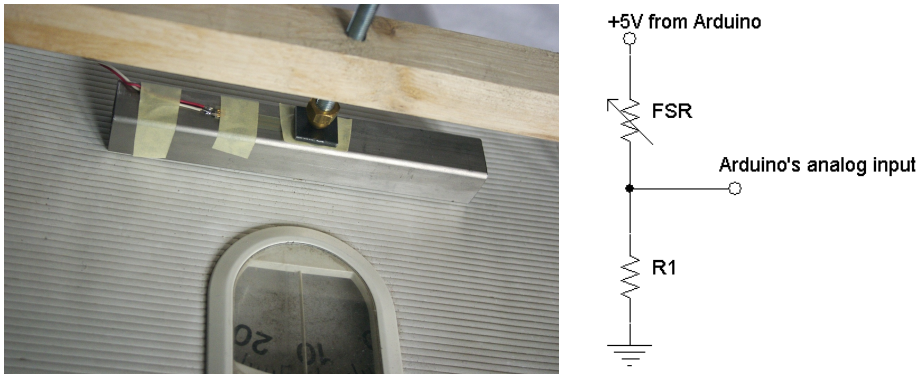


Fig. 3. Characterization of the FSR. (left) Mechanical setup. (right) Voltage divider for the acquisition of measured forces.

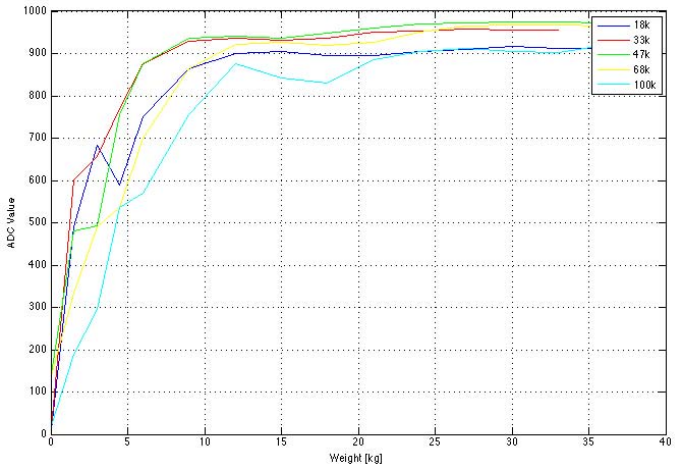
analog-to-digital converter (ADC) in the Atmel microcontroller on-board the Arduino [13].

A manual wooden press has been built, and mounted onto a weighing scale. This press had an interchangeable termination, providing mechanical matching with the active area of the sensor. With this simple setup, we could read the value of the force exerted by the press over the sensor. Fig. 3 (left) shows the setup.

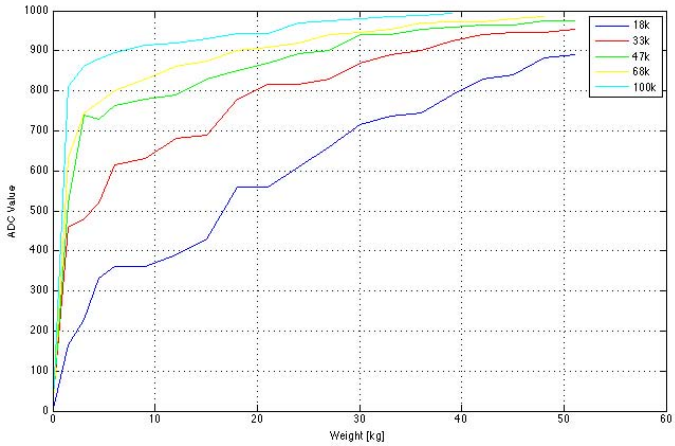
The voltage divider containing the FSR was connected to the Arduino analog input as shown in Fig. 3 (right), then pressed. As a result of this simple test, several sets of curves have been obtained for each sensor mapping force into ADC values, for changing values of the resistance R1 in the voltage divider. In practice, each curve represents the relationship between the applied force and the ADC output (ranging from 0 to 1023) for a specific value of R1. Fig. 4 displays the corresponding plots, respectively for sensors model 400 (a) and 402 (b). Their inspection shows that the system is more linear when using the model 402.

All measurements were made by choosing standard resistance values for R1 below 100 kΩ. Smaller values result in a smoother response that furthermore reduces the dynamic range, particularly with the model 402. This means that when the FSR reaches mechanical saturation, the voltage across R1 is below the maximum input voltage of the ADC (5 V). Conversely, values amounting to more than 100 kΩ introduce a strong nonlinearity. Choosing R1 = 18 kΩ results in a good compromise between range and linearity of the system.

In conclusion, the model 402 has been preferred for all sensing points and the resistance in the voltage dividers has been set to 18 kΩ: these choices have substantially improved the sensitivity of our instrumented shoes. Besides our specific application case, a simple measurement experience like the one described here can reward each time an FSR must be coupled to an ADC in an acquisition system of interest.



(a) Model 400.



(b) Model 402.

Fig. 4. Force/ADC maps for changing values of R1

3 Firmware

A good firmware program can enable the cheap hardware on-board the Arduino to perform as an effective front-end for data acquisition from the sensors. A reliable analog-to-digital conversion along with a low-latency transmission of the acquired data are in fact necessary to achieve this goal, when designing a real time interactive system providing instantaneous feedback. The essential tasks of the Arduino are illustrated in Fig. 5.

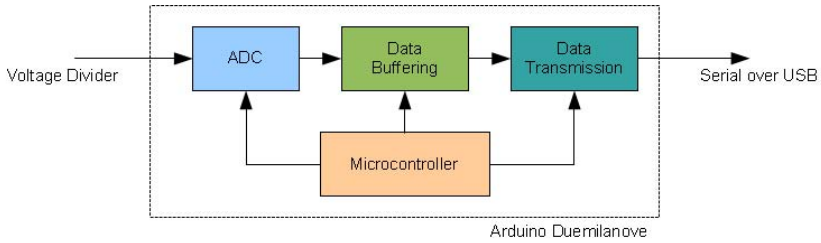


Fig. 5. Blocks of the data acquisition module

The firmware employed in the previous prototype of the interactive shoes included an application program, that repeatedly read the four values coming from the respective analog inputs through a traditional polling procedure nested inside a main loop [9]. Then, data were filtered in such a way that a packet containing the values was sent only if two subsequent samples differed at least by a given threshold. Concerning the serial transmission, we sent to the host packets made of four ADC values separated by a space character. Each packet ended with a separator (end of line) character.

Both the polling sequence and the transmission routine have been written using functions and procedure calls provided by the Arduino SDK. The use of this framework allows to write code very easily, thanks to the good level of abstraction provided by the embedded C++ API. Unfortunately the compiled program is often not efficient enough, especially if running on applications requiring a constantly low latency during the acquisition of uniform data flows.

In our case, the unconditioned use of the API affected the reliability of the analog-to-digital conversion and the constancy of the transmission rate of the serial connection over USB from the Arduino to the host. In fact, the previous firmware used the `analogRead` function, which, as a normal practice in acquisition procedures of this kind, was called for every channel during polling. As a consequence, the sampling frequency could not be kept stable: the main loop in fact can be interrupted at any moment, and it takes an unpredictable time for the system to return to the application program.

The transmission was managed using the `Serial.print` function, which prints out data to the serial port in the form of ASCII characters. In particular, every number digit was printed using the respective character. Once again, the API-based solution is quite standard and easy to be coded, but poorly performing. For instance, transmitting four 10-bit values requires, in the worst case (i.e., all the four values greater than 1000), $8 \cdot 4 \cdot 4 + 8 \cdot 4 = 160$ bits.

Interrupts can introduce heavy jitter. In the Arduino this problem is a consequence of the weak control of the low level structures inside the microcontroller through the API. In essence, the Arduino is not provided with resources capable to handle both processes in parallel:

- the ATMega168 is a single-thread machine, thus when the CPU is handling the ADC interrupt it cannot (among other things) transmit data over the serial connection;
- the Arduino USB connection is controlled by a FTDI FT232RL chip, which *emulates* a serial RS-232 connection over a USB line.

A consequence of the first point is that sampling with an exceedingly high frequency is unsafe since some values would be inevitably lost. The second point deals with the buffers that are present on the transmitter (inside the FTDI chip) and the receiver (in the USB controller driver, run by the operating system of the host computer). These buffers are governed by independent schedulers and procedures [14], and a low-level debugging of their mutual activity is a hard task.

As it often happens in this kind of architectures, the best performance is achieved by writing part of the code in assembly language at the cost of decreased readability and portability. A solution in between, that can be adopted with AVR-compliant microcontrollers like the ATMega168, is to write programs encoding API calls and AVR Libc instructions[3] together.

In the current firmware, we have set the ADC to work in *free-running* mode. Under this mode, the ADC interrupts the program only when the acquisition on the selected channel is completed. The sampling frequency can be selected by setting the ADC *prescaler* bits (ADPS) in the ADCSRA register. In free-running mode, a single conversion takes 13.5 CPU cycles, thus the sampling frequency turns out to be equal to $Fs = (16/13.5)/P$ MHz, where P represents the value of the prescaler.

Furthermore, a custom transmission protocol has been implemented [15]: binary data are transmitted instead of ASCII values using the `Serial.write` function. Every packet is formed by two bytes, B_{MSB} and B_{LSB}, such that their juxtaposition is $B_{MSB}\,B_{LSB}$ = 1CCCCVVV 0VVVVVVV. In this structure, the four bits denoted with C encode the channel number (16 channels are allowed), whereas the ten bits denoted with V encode the measured value thus guaranteeing sufficient accuracy for the acquired force. In the end, each sensor is assigned to a different channel. Hence, sending four values (one for each channel) using this protocol occupies a constant packet size, equal to $8 \cdot 2 \cdot 4 = 64$ bits.

3.1 Uniform Sampling of Force Data

We opted for a reliable, although not necessarily optimal assignment of the system variables in the new firmware, by empirically determining the transmission rates of in absence of drop-outs.

An oscilloscope was connected to a digital pin of the board. The pin was set at the beginning of the ADC routine, and was reset at the end of the same routine. The ADC routine called the transmission procedure only when the transmission buffer was full.

[3] `http://www.nongnu.org/avr-libc/`

When sending a cluster of bytes (i.e., a buffer) with the `Serial.write` function, the Atmel performs a serial transmission of data to the FTDI chip's buffer. For what we said above, data are sent on the USB line according to specific handshaking and buffering policies that in principle vary with the transmitter/receiver protocol.

The observations made on the oscilloscope uncovered that the serial transmission from the Atmel to the FTDI chip requires a large amount of CPU time. The values coming from the ADC which fall into this bottleneck are lost. For this reason we set the prescaler to $P = 128$, even if lower values are allowed.

By setting the buffer size to 2 bytes we obtained a regular sequence of impulses from the probed pin, testifying uniform sampling. In practice, this buffer size ensured that the transmission time is shorter than the sampling time. At this point, a linear relationship between the transmission rate and sampling frequency can be figured out. Table 1 lists possible choices complying with this relationship.

The values showed in Table 1 must be divided by the number of input channels, if a polling procedure is implemented. In our application, involving four channels, the sampling frequency per channel is $F_s = 5882/4 \approx 1470$ Hz. Considering that the FSR's have a response time of about 2 ms [12], a latency that is certainly smaller than any human response to psychophysical cue changes, the obtained sampling frequency is well above twice as much the Nyquist limit of $1/0.002 = 500$ Hz.

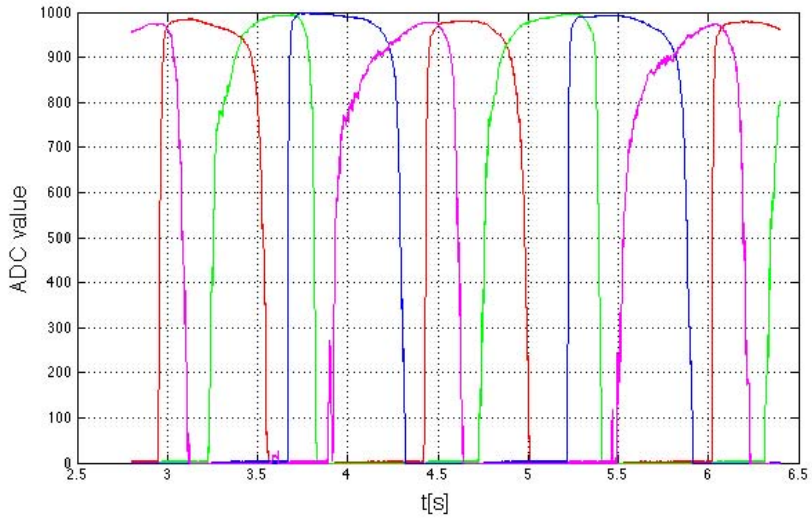
Table 1. Relationship between sampling frequency (F_s) and transmission rate (TXrate). Values per single channel. Prescaler set to 128.

TXrate [baud]	9600	19200	38400	57600	115200
T_s [ms]	2.1	1.04	0.52	0.36	0.17
F_s [Hz]	476	961	1923	2778	5882

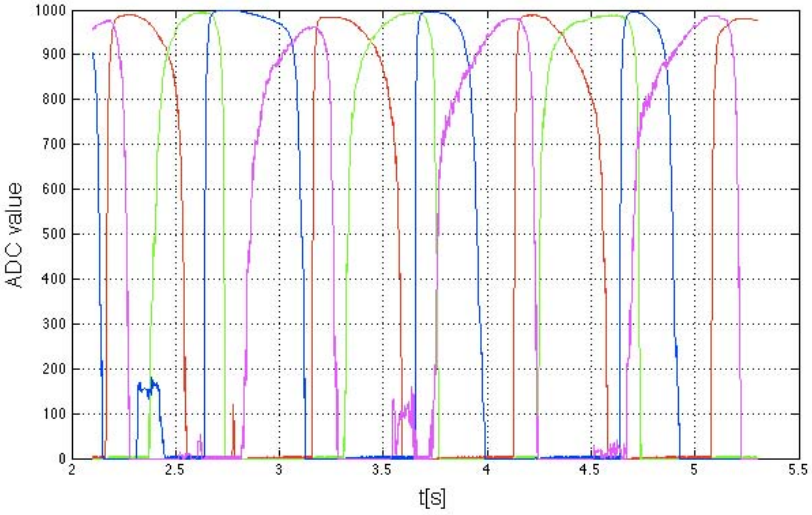
4 Results and Conclusions

Fig. 6 shows some plots of force signals recorded using the proposed system. These plots provide evidence of a substantial absence of noise and jitter in the data. If perhaps not accurate enough for applications where extremely high resolution is mandatory, such data is suitable for most interaction design and other applications.

In our case the force signals are sent to the host computer for further processing [16]. Still, for many other acquisition processes requiring similar performance and cost, we think that designers and others can make profitable use of the solutions and tests described in this work.



(a) Slow walk.



(b) Fast walk.

Fig. 6. Force plots. Red = left heel, green = left toe, blue = right heel, magenta = right toe.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme under FET-Open grant agreement 222107 NIW - Natural Interactive Walking.

References

1. Stott, J.R.R., Hutton, W.C., Stokes, I.A.F.: Forces Under the Foot. Journal of Bone and Joint Surgery 55-B, 335–345 (1973)
2. Manley, M.T., Solomon, E.: The Clinical Assessment of the Normal and Abnormal Foot During Locomotion. Journal of Prosthetics and Orthotics 3, 10–110 (1979)
3. Soames, R.W., Blake, C.D., Stott, J.R.R., Goodbody, A., Brewerton, D.A.: Measurement of pressure under the foot during function. Journal of Medical and Biological Engineering and Computing 20, 489–495 (1982)
4. Pollard, J.P., Quesne, L.P.L., Tappin, J.W.: Forces Under the Foot. Journal of Biomedical Engineering 5, 37–40 (1983)
5. Zhu, H., Maalej, N., Webster, J.G., Tompkings, W.J., Bach-Y-Rita, P., Wertsch, J.J.: An Umbilical Data-Acquisition System for Measuring Pressures Between the Foot and Shoe. IEEE Transactions on Biomedical Engineering 37, 908–911 (1990)
6. Faivre, A., Dahan, M., Parratte, B., Monnier, G.: Instrumented shoes for pathological gait assessment. Mechanics Research Communications 31, 627–632 (2004)
7. Morris, S.J.: A Shoe-Integrated Sensor System for Wireless Gait Analysis and Real-Time Therapeutic Feedback. PhD thesis, Massachusetts Institute of Technology (2004)
8. Miranda, E., Wanderley, M.: New Digital Musical Instruments: Control and Interaction Beyond the Keyboard. AR Editions (2006)
9. Papetti, S., Fontana, F., Civolani, M.: A shoe-based interface for ecological ground augmentation. In: Proc. 4th Int. Haptic and Auditory Interaction Design Workshop, Dresden, Germany, vol. 2 (2009)
10. Hollinger, A., Wanderley, M.M.: Evaluation of commercial force-sensing resistors. Unpublished report 1, Input Devices and Music Interaction Laboratory (IDMIL), Music Technology Schulich - School of Music, McGill University Montreal, QC, Canada (2006), http://www.idmil.org/publications
11. Vecchi, F., Freschi, C., Micera, S., Sabatini, A., Dario, P., Sacchetti, R.: Experimental evaluation of two commercial force sensors for applications in biomechanics and motor control. In: Proceedings of the 5th Annual Conference of the International Functional Electrical Stimulation Society, Aalborg, Denmark, p. 44 (2000)
12. Interlink: Force sensing resistor integration guide and evaluation parts catalog. Datasheet 90-45632 Rev. D, Interlink Electronics, 546 Flynn Road, Camarillo, CA 93012, USA (2009), http://www.interlinkelectronics.com/library/media/papers/pdf/fsrguide.pdf
13. Atmel: Atmega168 datasheet. Datasheet 2545RAVR07/09, Atmel Corporation, 2325 Orchard Parkway, San Jose, CA 95131, USA (2009), http://www.atmel.com/dyn/products/datasheets_v2.asp?family_id=607
14. FTDI: Data throughput, latency and handshaking. Application note AN232B-04 Rev. 1.1, Future Technology Devices International Ltd., Seaward Place, Centurion Business Park, Glasgow, G41 1HH, UK (2006), http://www.ftdichip.com/Documents/AppNotes/AN232B-04_DataLatencyFlow.pdf
15. Visell, Y., Law, A., Ip, J., Rajalingham, R., Smith, S., Cooperstock, J.R., Borin, G., Civolani, M., Fontana, F., Polotti, P., Nordahl, R., Serafin, S., Turchet, L.: Contact-based sensing methods for walking interactions. Deliverable 3.1, NIW project (2009), http://www.niwproject.eu
16. Papetti, S., Fontana, F., Civolani, M., Berrezag, A., Hayward, V.: Audio-tactile display of ground properties using interactive shoes. In: Proc. 5th Int. Haptic and Auditory Interaction Design Workshop (2010); Elsewhere in these proceedings