Tappingduino: A versatile shield for paced finger tapping experiments with Arduino

Paula Caral[1], Tomás Correa Morales[1], Paz García Lodi Fe[1], Ariel D. Silva[2,4], Ulises Bussi[2], Ignacio Spiousas[3,4], and Rodrigo Laje[2,4]

[1]Departamento de Física, FCEN, Universidad de Buenos Aires, Argentina
[2]Departamento de Ciencia y Tecnología, Universidad Nacional de Quilmes, Argentina
[3]Laboratorio del Tiempo y la Experiencia, Universidad de San Andrés, Argentina
[4]CONICET, Argentina

October 24, 2023

Abstract

Paced finger tapping is the simplest task to study sensorimotor synchronization (SMS), that is the mostly specifically human behavior of synchronizing motor actions to an external metronome [1–4].

In this work we show the design and construction of an Arduino shield for paced finger-tapping experiments. The pin footprint is compatible with Arduino Mega 2560 [5].

1 System specifications

The device is intended to be versatile and to allow many different types of experimental conditions, all related to the paced finger-tapping task.

Independent audio channels. Auditory stimuli are extensively used in finger tapping experiments. This device would allow experiments with dichotic auditory stimuli, not only for the study of time processing in healthy subjects [6] but also for the study of several perceptual and motor disorders [7], cognitive impairments, and diseases [8].

Sound timbre. Control of sound quality or timbre is important in some experimental designs, for example choosing the sound waveform to be a square wave, a sinusoidal wave, etc.

Noise masking. In order to better control the auditory feedback from the tap itself it is important to have good quality headphones, and use noise masking. White noise generation capabilities are thus desirable.

Virtual ground. Arduino is a single-supply device, that is it only uses 0 and 5 V supply rails. Audio signals, however, oscillate above and below a reference. A virtual ground is thus needed at a value of approximately 2.5 V to allow the audio signals to oscillate around it.

Force sensor. Responses are detected by a force-sensitive resistor (FSR). Some experimental designs need two of them [9] so this is also a requirement.

Visual stimuli. Experimental designs to study sensory and sensorimotor integration need the stimuli to have at least two different modalities. This is why we included two LEDs in the device.

Temporal alignment. Simultaneous recording of EEG data is now very usual in finger tapping

experiments. A connector is included to allow a one-way, 3-bit alignment signal between Arduino and a second recording device.

Servo driving. Some experimental designs include *spatial* perturbations in addition to the usual temporal ones [10]. This is why we include connectors to drive two servos.

2 Circuit design

We designed the circuit under the constraint that all op amps should be single-supply (LM324) so the shield is fully compatible with the voltage range of Arduino. There are several technical reports available with recommendations and design criteria for single-supply op amp circuits [11].

We used Eagle, a well-known electronic design automation software for schematic creation and editing, auto-routing, and PCB layout. The circuit was tested on protoboard before printing and soldering. Figure 1 shows the circuit schematic.

2.1 Virtual ground

Block 1 in Figure 1 is a standard design of a virtual ground: a voltage divider with an op amp in a follower configuration [11]. This allows us to match the impedance by decoupling the voltage divider from the rest of the circuit (ideal voltage divider). An additional capacitor helps filtering out high-frequency noise.

2.2 Independent audio channels

Block 2 in Figure 1 contains the two independent audio channels A and B (left and right, respectively). Each channel has an op amp in a variable-gain, summing amplifier configuration. Each summing amplifier has three inputs corresponding to stimulus, feedback, and noise. Each input signal comes from a different output pin in the Arduino and enters a voltage divider prior to the summing amplifier to prevent saturation. The variable gain is controlled by a dual potentiometer and allows the subject to adjust the sound volume. The reference voltage for the summing amplifier is virtual ground. Resistor values were chosen so that the impedance at each stage in the chain divider-summing-amplifier is an order of magnitude greater than the previous stage (10 kΩ, 100 kΩ, 1 MΩ), thereby reducing unwanted currents and voltage drops.

2.3 Alignment, visual stimuli, and servos

Block 3 in Figure 1 includes a 3-pin connector for sending a temporal alignment signal to other devices, two LEDs for the delivery of visual stimuli, and two 3-pin connectors for driving two servos.

2.4 Force sensors

Block 4 in Figure 1 shows the connectors for the two force sensors. Each sensor has a potentiometer (trimpot) to adjust the circuit's sensitivity so it can either detect a soft tap (i.e. a low-voltage signal) or prevent a strong tap from saturating (i.e. a high-voltage signal).

2.5 Low-pass filtering

We decided not to include a low-pass filter due to space constraints: two additional op amps would be needed (one for each audio channel), meaning a whole additional integrated circuit plus its connections, in addition to resistors and capacitors. As shown below (Figure 2), the PCB is already very crowded for hand soldering and a double-layer copper-clad laminate was necessary to accommodate all traces. However, noise capabilities are included as described above; in case noise is desired, an external low-pass filter should be added between the audio output of the shield and the headphones.

3 PCB design

Components were initially located based on a trade-off among several criteria: proximity to corresponding Arduino pins, proximity among related components, simplicity and accessibility for the user, and

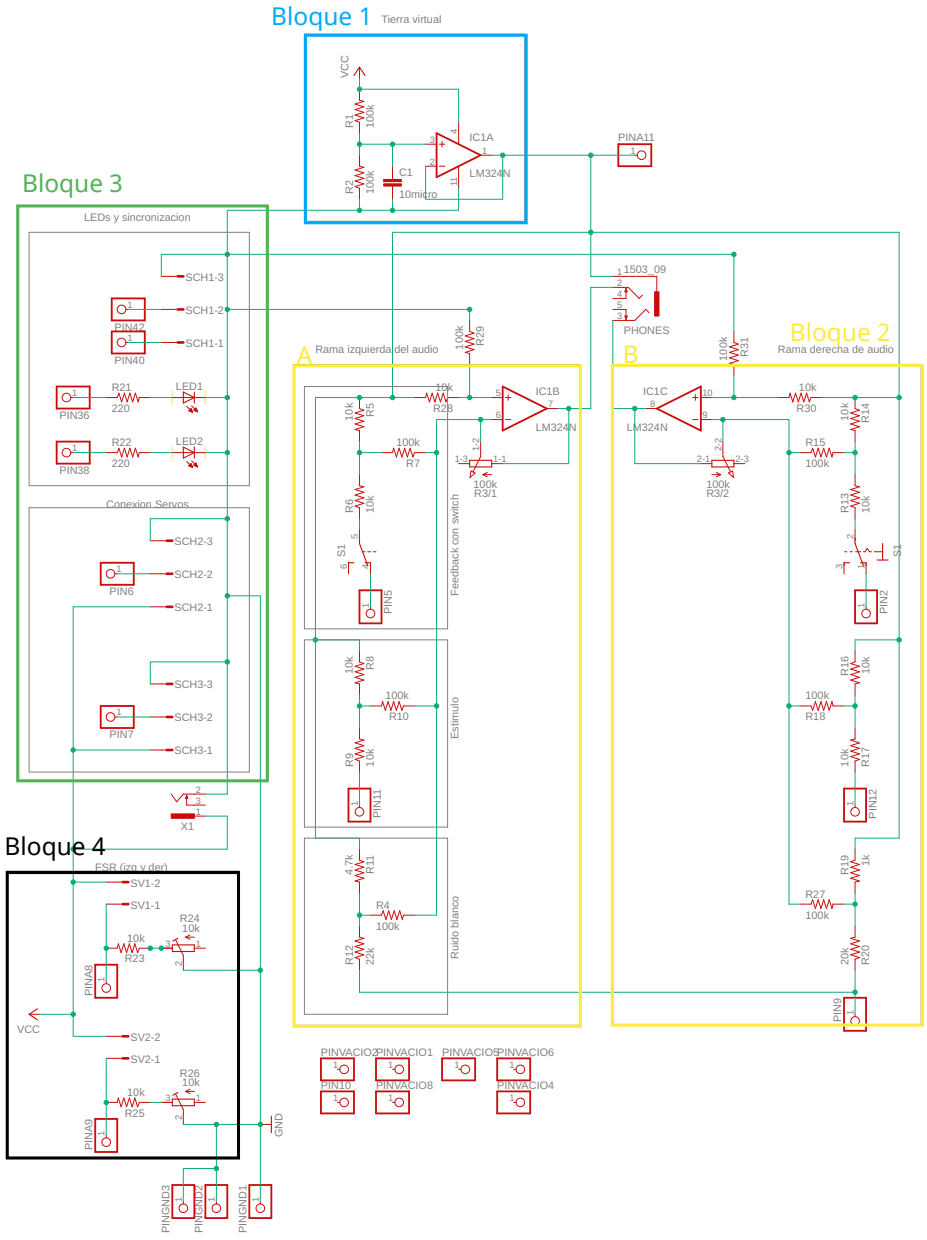

Figure 1: Circuit schematic. Block 1: virtual ground. Block 2: independent audio channels. Block 3: temporal alignment, visual stimuli, and servo driving. Block 4: force sensor connectors. Pin A11 is used as analog input to record the actual value of virtual ground and making it available to the code.

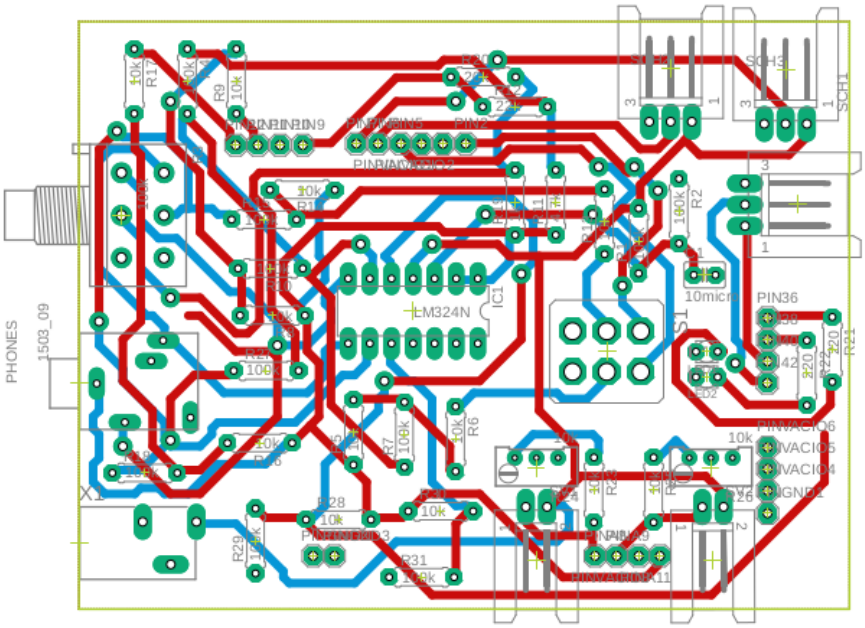

Figure 2: PCB layout. Red traces: top layer, blue traces: bottom layer. Ground planes not shown.

simplicity for soldering. We decided to use a double-sided copper-clad laminate. We took into account the following specific considerations:

Soldering to a specific layer. Some components have very short legs and are very difficult to solder to the layer where they sit due to potential damage to the component. For instance, in the top layer: switches, op amps sockets, potentiometers, female miniplug connectors, and trimpots; in the bottom layer: pins to connect to Arduino. That is why we constrained the autorouting to prevent a trace to reach any of these components on the same layer (polygons in tRestrict and bRestrict in Eagle).

Forbidden regions for vias. Vias are connections between the two copper layers. Components that sit close to the layer are difficult to solder when they have a via beneath them, so we defined polygons (vRestrict) to prevent the location of vias beneath components (other than regular resistors).

Central connector in Arduino. Arduino Mega 2560 has a six-pin connector in the middle, so any soldering in the bottom layer could potentially make electrical contact with it. To avoid this we placed a bRestrict polygon thus preventing any component or connection to be located there.

Extra pins for better structural stability. We added several pins with no electrical function in order to have a better union between Arduino and the shield.

Size of pads and vias. To make drilling and soldering easier we increased the size of pads and vias.

Ground planes. We covered empty spaces in both layers with polygons connected to ground for easier routing and reducing electrical noise and interference.

Isolation of vias. Vias connected to the ground

plane were termically isolated to prevent loss of heat during soldering.

Size of laminate. The size of the copper-clad laminate was increased beyond the size of the Arduino to give Eagle a little more room to route traces.

Figure 2 shows the final design of the PCB. We attempted several different initial distribution of components and several auto-routings each, and chose the one that maximized the completion percentage and minimized the number of vias to simplify soldering. Figure 3 shows the actual finished shield.

4 Software and flow control

4.1 Flow control

The experimental unit in a typical finger tapping experiment is the trial, where the subject is exposed to a sequence of consecutive 30-45 brief stimuli. Trials are separated by pauses, and are grouped in blocks. A full experiment is a few blocks long.

The general control of flow during the experiment is performed by an experiment controller written in Python, while the control during a trial switches to the Arduino code in C. The experiment controller and the trial controller communicate to each other with text messages via the Serial port: the experiment controller sends the current trial parameters (interstimulus interval, number of stimuli, perturbation size, etc.), the trial controller records a trial and sends back the trial data (e.g. response times).

4.2 Trial controller and Arduino timers

We based our Arduino software development on Taparduino [12].

One of the main functions of the Arduino-shield duo is producing sound. As the output pins in Arduino are only capable of having digital values (either HIGH or LOW), a technique called pulse width modulation (PWM) is needed, where the pin is switched between HIGH and LOW at a high frequency with a variable duty cycle. The averaging over many oscillations leads to a continuous mean value depending on the duty cycle (close to LOW for a small duty cycle, or close to HIGH for a large duty cycle).

PWM is performed via timers, which are dedicated internal clocks/counters that operate in parallel with the central processor and can execute small portions of code on an interrupt basis. Timers in all operation modes work by counting from 0 through $2^b - 1$, where b is the number of bits of the timer, at a frequency $f_{\text{timer}} = 16\,\text{MHz}/2^b$, where 16 MHz is the oscillation frequency of the crystal in Arduino. Once counting reaches its TOP value it overflows and starts over again. The counting frequency can be modified by setting prescalers according to eq. 1 [13]:

$$f_{\text{timer}} = \frac{16\,\text{MHz}}{N \cdot 256} \tag{1}$$

Prescaler N can take values $N = 1, 8, 16, \ldots$.

Every timer has two channels each associated with a different output pin, and can have different duty cycles. We used timers to generate sounds corresponding to stimuli and (optional) auditory feedback from the taps, and to generate masking noise. The appropiate timer configuration is known as FastPWM and is shown in Figure 4 [14].

We chose timers 1 and 3 for stimulus and feedback, respectively, and their channels A and B for left and right audio channels, respectively. Registers OCR1A and OCR1B determine the duty cycle of each channel in timer 1 (OCR3A and OCR3B for timer 3) and can be set independently. This allows us to send different audio signals to either ear. Timers 1 and 3 are 16-bit timers and can be set in 9-bit Fast PWM mode, which sets the interrupt frequency at approximately 31 kHz. This allows us to produce sound with an acceptable quality up to a fundamental frequency of around 2 kHz.

We used timer 2 to generate noise. Timer 2 has less resolution than timers 1 and 3 (it is an 8-bit timer) but quality is not as important in this case.

4.3 Sound production

An analogous sound wave can be produced by a digital output by means of PWM: the digital output

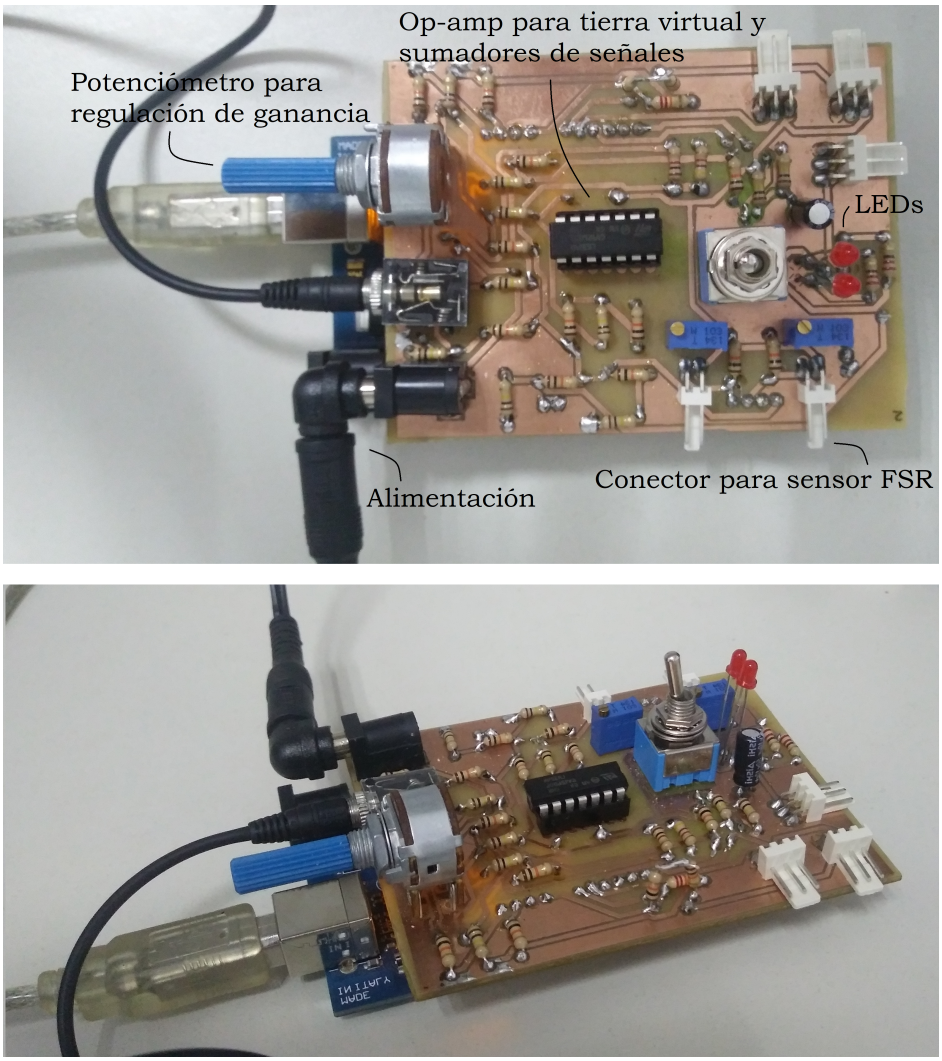

Figure 3: Actual finished shield, plugged on top of Arduino.

is switched between HIGH and LOW at a high frequency with a variable duty cycle, and then a low-pass filter with an appropriate cutoff frequency leads to a smoothed version of the signal whose mean value tracks the duty cycle [15]. Figure 5 shows how a square wave with a time-varying duty cycle leads to a smooth low-pass filtered sinusoidal curve.

The frequency of the sinusoidal is determined by the frequency of the duty cycle, and this in turn is set in the following way. Every time the timer overflows we set a new OCRnX value according to a lookup table with values that uniformly sample one period of the desired waveform, such that the duty cycle follows it. Figure 6 shows the contents of the lookup table for a sinusoidal waveform (orange curve). The actual frequency of the duty cycle (and thus of the sound signal) can be chosen by down- or up-sampling the lookup table. Down-sampling (by skipping values)

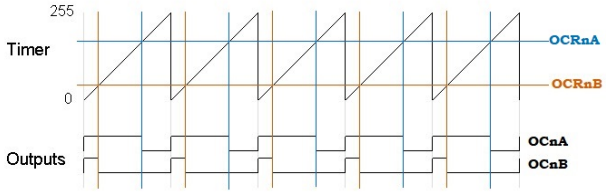

Figure 4: Timing diagram of a timer in 8-bit Fast PWM mode. The timer counts from 0 through 255 where it overflows and executes an interrupt. Registers OCRnA and OCRnB determine the duty cycle of each channel.

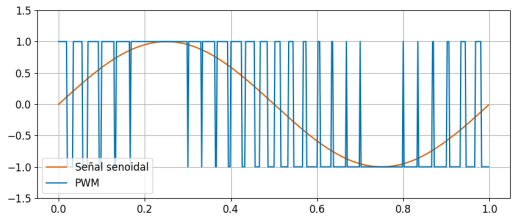

Figure 5: PWM with a relatively slowly varying duty cycle can produce a smooth sinusoidal signal with a well-defined frequency.

leads to higher frequencies, while up-sampling (by repeating values) leads to lower frequencies. Either way can be detrimental to sound quality if excessive.

The blue curve in Figure 6 corresponds to down-sampling the lookup table to get a 440 Hz sinusoidal waveform. The lookup table is down-sampled by selecting values separated by a fixed quantity called the *frequency tuning word* (FTW):

$$FTW = \frac{f_{\text{out}} \times 2^{32}}{f_{\text{timer}}} \tag{2}$$

where f_{out} is the desired output frequency in Hz, 2^{32} is the resolution of the phase accumulator, and f_{timer} is the timer frequency in Hz [16].

4.4 Noise generation

We generate noise by randomly switching the PWM duty cycle such that the low-pass filtered signal ran-

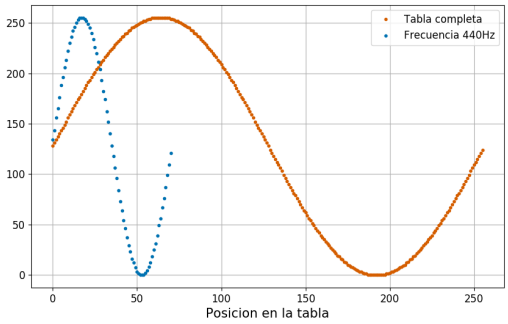

Figure 6: Contents of the lookup table for a sinusoidal waveform (orange), and down-sampling to a frequency of 440 Hz (blue).

domly oscillates around virtual ground. A noise amplitude is defined with value up to half the TOP value of the timer, and a number of overflow periods is pseudorandomly chosen by a lightweight linear-feedback shift register [17] (range 2-4). The duty cycle is set at a value equal to virtual ground + amplitude for the chosen number of periods, then reset to virtual ground - amplitude for the same number of periods, and then the number of periods is newly chosen to start over. In this way the noise power spectrum is relatively independent of the chosen amplitude.

5 Open software and hardware

Code for the experiment controller in Python and the trial controller in C, and .sch and .brd Eagle files for the circuit schematic and PSB are available in the following GitHub repository: `https://github.com/SMDynamicsLab/Tappingduino`.

References

[1] M. L. Bavassi, E. Tagliazucchi, and R. Laje. “Small perturbations in a finger-tapping task reveal inherent nonlinearities of the underlying

error correction mechanism". In: *Hum Mov Sci* 32.1 (2013), pp. 21–47.

[2] L. Bavassi et al. "Sensorimotor synchronization: neurophysiological markers of the asynchrony in a finger-tapping task". In: *Psychol Res* 81.1 (2017), pp. 143–156.

[3] B. H. Repp. "Sensorimotor synchronization: a review of the tapping literature". In: *Psychon Bull Rev* 12.6 (2005), pp. 969–992.

[4] B. H. Repp and Y. H. Su. "Sensorimotor synchronization: a review of recent research (2006-2012)". In: *Psychon Bull Rev* 20.3 (2013), pp. 403–452.

[5] Arduino. *Mega 2560 Rev3*. `https://docs.arduino.cc/hardware/mega-2560`.

[6] D. M. Piazza. "Cerebral lateralization in young children as measured by dichotic listening and finger tapping tasks". In: *Neuropsychologia* 15.3 (1977), pp. 417–425.

[7] O. Bo, K. Hugdahl, and E. Marklund. "Dichotic listening in children with serious language problems". In: *Percept Mot Skills* 68.3 (1989), pp. 1291–1301.

[8] M. Aoun Sebaiti et al. "Systematic review and meta-analysis of cognitive impairment in myalgic encephalomyelitis/chronic fatigue syndrome (MECFS)". In: *Sci Rep* 12.1 (2022), p. 2157.

[9] L. Versaci and R. Laje. "Time-oriented attention improves accuracy in a paced finger-tapping task". In: *Eur J Neurosci* (2021).

[10] S. L. López and R. Laje. "Spatiotemporal perturbations in paced finger tapping suggest a common mechanism for the processing of time errors". In: *Sci Rep* 9.1 (2019), p. 17814.

[11] B. Carter. *A Single-Supply Op-Amp Circuit Collection - SLOA058*. Tech. rep. Texas Instruments, 2000.

[12] B. G. Schultz and F. T. van Vugt. "Tap Arduino: An Arduino microcontroller for low-latency auditory feedback in sensorimotor synchronization experiments". In: *Behav Res Methods* 48.4 (2016), pp. 1591–1607.

[13] Atmel Corporation / Microchip Technology Incorporated. *ATMega2560 Manual*. `https://www.microchip.com/wwwproducts/productds/ATmega2560`. 2023.

[14] Ken Shirriff. *Secrets of Arduino PWM / Timers mode: Overflow - Fast PWM*. `http://www.righto.com/2009/07/secrets-of-arduino-pwm.html`.

[15] Arduino Tutorials. *Fading: Use an analog output (PWM pin) to fade an LED*. `https://www.arduino.cc/en/Tutorial/Fading`.

[16] Gamry Instruments. *Waveform Generation and Frequency Resolution*. `https://www.gamry.com/application-notes/EIS/waveform-generation-and-frequency-resolution`.

[17] Wikipedia. *Galois LFSRs*. `https://en.wikipedia.org/wiki/Linear-feedback_shift_register#Galois_LFSRs`.