A Teensy microcontroller-based interface for wide-field optical imaging and behavioral experiments

Michael Romano, Mark Bucklin, Dev Mehrotra, Robb Kessel, Howard Gritton, Xue Han

**Abstract**

Systems neuroscience experiments often require the integration of precisely timed data acquisition and behavioral monitoring. While specialized commercial systems have been designed to meet various needs of data acquisition and device controls, they often fail to offer flexibility to interface with new instruments and variable behavioral experimental designs. For example, it has been difficult to integrate recently developed sCMOS cameras with various input and output devices for high speed, large scale calcium imaging experiments during behavior. We here developed a Teensy (version 3.2) microcontroller-based interface that offers high-speed, precisely timed digital behavioral data acquisition, and digital and analog outputs for controlling sCMOS cameras and other devices. We demonstrate the efficacy and the temporal precision of the Teensy interface in two experimental settings. In one example, we used the Teensy interface for reliable recordings of an animal’s directional movement on a spherical treadmill, while controlling image acquisition from a sCMOS camera. In another example, we used the Teensy interface for temporally precise delivery of auditory and visual signals in a trace conditioning learning behavioral paradigm, as well as image acquisition from a sCMOS camera. These examples demonstrate that Teensy provides an low-cost and flexible interface for integrating image acquisition into behavioral experimental designs, allowing high-speed and temporally precise imaging analysis during behavior.

**Introduction**

Recent advances in sCMOS camera technology and genetically encoded calcium sensors enable large scale fluorescence imaging of thousands of individual cells’ activity during behavior (Klaus, et al., 2017; Barbera, et al., 2016; Mohammed, et al., 2016; Markowitz, et al., 2018). One key technical aspect neural network analysis of behavior is the temporal precision, where neural activities need to be precisely aligned with behavioral features (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018). However, it has been difficult to integrate sCMOS cameras, deployed in large scale calcium imaging experiments, with devices needed to monitor and control behavioral experiments. Traditional Analog/Digital interfaces are often operated by programs, such as MATLAB, that offer a wide range of applications. However, using MATLAB can lead to undesired temporal delays if code is implemented inefficiently, because it utilize a full operating system that needs to balance the demand of many systems operations. For example, if using a MATLAB-based interface to time camera triggers to a sCMOS camera and to control other experimental tasks,it tends to result in undesired variability in experimental timing, including frame capture, data acquisition, or device control.

Over the last decade, microcontrollers marketed to hobbyists have gained popularity across a variety of scientific fields (Sanders & Kepecs, 2014; D'Ausilio, 2012; Chen & Li, 2017; Husain, Hadad, & Zainal Alam, 2016). Microcontrollers are small, low-cost, capable of delivering precisely timed digital outputs with microsecond time resolution, while using user-friendly, open-source software functions. Arduino was the first major microcontroller to gain substantial popularity. Recently, Teensy (most recent version is 3.2)was developed, which has all the key features of Arduino microcontrollers, as well as the additional feature of delivering analog output. Teensy utilizes the same open-source Arduino software environments, and thus is easy to program without the need of advanced programming experience (D'Ausilio, 2012). Microcontrollers can thus be easily adapted to various experimental needs by neuroscientists, including the integration of newly developed instruments.

Arduino (model Uno-Rev3) has recently been used to control a two-photon microscope through initiating image acquisition of an entire image sequence, where the timing of each image frame is not specified (Micallef, Takahashi, Larkum, & Palmer, 2017). Thus, this approach requires post-experimental data interpolation to re-align frame timing with behavioral data, reducing temporal precision. One way to precisely time sCMOS image acquisition is through timed capture of each individual frame via digital commands. Thus, there currently exists a need to engineer a simple interface for precise delivery of digital signals for camera control in biological imaging experiments while maintaining accurate alignment with behavioral control and data acquisition.

Here, we demonstrate a flexible Teensy interface for accurate digital data acquisition and synchronous delivery of analog and digital a signals, in two experimental paradigms. Teensy interface delivered digital pulses with microsecond precision to initiate frame capture at a desired speed, while simultaneously collecting movement data during voluntary movement or controlling/recording sensory stimuli for trace conditioning. We also demonstrate the ability of the Teensy interface to generate analog high frequency sound waveforms. Together, these results demonstrate that the Teensy interface, containing a Teensy microcontroller equipped with specific hardware modules and a set of custom software functions, offers a flexible, accurate, and simple to use environment for imaging experiments during behavior, analog with the option of analog output..

**Methods**

*General overview of construction of Teensy boards*

The two experimental designs used are shown in Figure 1, and the specialty components required to build these designs are shown in Tables 1 and 2. In both cases, a Teensy 3.2 (PJRC.COM, LLC, part #: TEENSY32) is mounted on top of a printed circuit board via standard female headers (like those found here: SparkFun Electronics, PRT-00115). Female headers were then soldered to the PCB. Output from the Teensy was directed from pins via the female headers to SMA connectors (for example: Digi-Key, part # CON-SMA-EDGE-S-ND) via 22 gauge wires (for example: Digi-Key, part #1528-1743-ND), and from SMA connectors toward external devices via SMA cables (for example, Digi-Key, part # 744-1429-ND). To easily set the sampling frequency and length of the motor acquisition experiment in the Teensy, we developed a simple MATLAB graphical user interface that directs the Teensy to initiate experiments via a standard USB-microUSB cable (for example: Digi-Key, part # AE11229-ND). A similar interface was designed for the trace conditioning experiment in which the length of each trial and total number of trials were specified.

*Motor acquisition experiment*

In this experiment, we performed motion tracking using two ADNS-9800 gaming sensors (<https://www.tindie.com/products/jkicklighter/adns-9800-laser-motion-sensor/>, Tindie, part: “[NS-9800 Laser Motion Sensor](https://www.tindie.com/products/jkicklighter/adns-9800-laser-motion-sensor/)”, see Table 1), while delivering digital pulses that could be used to trigger a sCMOS camera. The overall design for this experiment is shown in Figure 1A. Mice were positioned on top of a buoyant Styrofoam ball floated by house air as described previously (Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007). Two ADNS-9800 gaming sensors were positioned at the equator of the sphere, at an angle of approximately 75 degrees from one another, so that the total distance travelled at any one time point was computed using the following equation:

Where yR and yL are the y readings from the left and right sensors, and is the angle between the two sensors (75 degrees). Velocity was computed as the distance divided by the time between two adjacent readings.

These two sensors were connected to a Teensy 3.2 via simple serial peripheral interface (SPI) connections with insulated 22 gauge wires as shown in Figure 2A. A crimping tool (for example, SparkFun Electronics, product TOL-13193) was used to attach crimp pins and housing to the ends of the wires and in order to connect them to the Teensy and to the sensors (for example, female crimp pins: Pololu Corporation, part #: 1930; male crimp pins: Pololu Corporation, part # 1931; housing: Pololu Corporation, part #: 1903).

Readings from motion sensors were extracted with freely available functions on Github (<https://github.com/markbucklin/NavigationSensor>). These functions read from the “motion burst” register of each sensor, from which we acquired motion data. For the counts per inch setting, which determines the sensitivity of the sensors to external movement, we used a value of 3400 counts per inch. To control experimental timing with high precision, we utilized the “IntervalTimer” function unique to the standard Teensy library, which repeatedly calls a function at intervals specified in microseconds, which we specified as 50,000 (50 ms). On every call of the main function in our experiment, the accumulated displacement since the previous call in both the x and y directions were collected and sent to the attached PC, and then a digital “on” pulse that lasts for 1 ms was sent out of a digital pin using the DigitalIO library (<https://github.com/greiman/DigitalIO>) to simulate a mock frame capture. Instead of using the default Arduino programming environment to upload our code to the Teensy, we used PlatformIO (<https://platformio.org/>), an add-on to the widely-used Atom text editor (<https://atom.io/>). This allowed us to easily build and upload our multi-folder library to the Teensy.

The output from the Teensy, representing frame capture triggers, was measured by an external device at 3051.76 Hz (Tucker-Davis Technologies RZ5D (TDT RZ5D)).

*Trace eyeblink conditioning experiment*

In this experiment, we ran an experiment in which the Teensy was programmed to deliver a sound simultaneously with an LED and a puff in a trace conditioning behavioral paradigm, while delivering digital pulses to a sCMOS camera for image capture every 50 ms. The general setup is shown in Figure 1B, and the components required to build this design are shown in Table 2.

To deliver an audible sound through the Teensy, we used a prop shield module available for Teensy (PJRC.COM, LLC., part #: PROP\_SHIELD). This is a simple add-on component that amplifies analog output to drive speakers with resistances up to 8 ohms (shown in Figure 2B as pin A14). The prop shield was attached to Teensy with 14x1 double insulator pins (PJRC.COM, LLC., part #: HEADER\_14x1\_D), and the output was connected to a speaker.

Here we used the “elapsedMicros” function, which keeps time in microseconds, to time all of the experimental events. Every 50 ms, a main function was called that updated the status of the digital pins associated with the “puff” and the light stimulus, and updated the amplitude of the 9500 Hz sine wave (amplitudes were set to 0.05 during audio stimulus time periods, and 0 elsewhere). At the termination of a trial, this function incremented the trial number. Finally, a 1 ms digital pulse was delivered to instantiate a theoretical camera trigger. The speaker, camera, and puff source were attached to the microcontroller using simple coaxial cables with SMA connectors, as shown in Figure 1A. The same programming environment (PlatformIO on top of Atom) was utilized, and functions such as “fastPinMode” and “fastDigitalWrite” were utilized to decrease latency.

The digital outputs for puff and camera were recorded with a commercial system (TDT RZ5D) at 3051.76 Hz, and the analog output for the tone was recorded at 24414.0625 Hz by the TDT RZ5D without additional amplification.

*Statistics*

Linear models were constructed using the “fitlm” function in MATLAB 2017b. Theoretical timings, to which measured timings were compared, were each taken to be timings beginning at 0 seconds in equal increments of 50 milliseconds for both experiments.

**Results**

A widely utilized design in imaging experiments utilizes commercial data acquisition boards in conjunction with software written and executed by a PC. However, PCs run a multitude of processes simultaneously, and it is therefore challenging to reliably time experimental events using these devices alone. An even more challenging task is aligning camera frames with experimental events following the conclusion of an experiment. For example, initiating a recording session or trial with a PC poses two problems. First, if one uses the PC to generate digital pulses via a data acquisition board in order to initiate frame capture for every frame individually, it is likely that the actual rate of frame capture will exhibit some variability. As previously noted, this is due to the fact that PCs have to balance the execution of multiple tasks simultaneously. However, with this design, an experimenter would still be able to programmatically synchronize behavioral observations or triggers with specific frames. Second, an alternative design involves using a PC only to initiate the beginning of a trial, with the camera then imaging at a fixed frequency independently of the behavioral aspects of the experiment. This solves the issue of camera jitter, but introduces two other issues. First, behavioral events cannot be measured with respect to frame capture, necessitating some kind of interpolation to align behavioral and imaging data. Second, relative timings between the camera and the behavioral data acquisition could experience timing drift if their clocks run at slightly different rates. In this case, even interpolating data won’t fully recover accurate alignment between the two data sets. Thus, an ideal experimental design has one clock kept by a single experimental controller that provides precise timing of image capture while simultaneously controlling and monitoring behavioral data synchronously with the image capture. Microcontrollers fill this need.

As mentioned previously, microcontrollers such as Arduino’s have gained popularity in neuroscience research due to their user-friendly interface and their flexibility of device integration (D'Ausilio, 2012; Chen & Li, 2017; Micallef, Takahashi, Larkum, & Palmer, 2017). However, Arduino devices do not have direct analog output. Recently, the Teensy 3.2 has been developed, which has analog output, a comprehensive Audio library, and the capability to use the IntervalTimer function, which makes highly precise timing of repeated events simple to orchestrate. Here, we present a Teensy-based interface to integrate and synchronize sCMOS camera image acquisition with behavioral experimental control.

*Mice Motion tracking experiment*

To measure locomotion from awake head fixed mice, we implemented a novel design wherein we used a Teensy interface to record from two ADNS-9800 motion sensors. These sensors are affixed to a “spherical treadmill” setup, as described by Dombeck, Khabbaz, Collman, Adelman, & Tank (2007).

ADNS-9800 sensor boards are inexpensive, and are more sensitive than regular computer mice as used in previous designs. For example, they can measure up to 8200 counts per inch, allowing for a more precise measure of locomotion parameters. Additionally, ADNS-9800 sensors have a high maximum sampling rate of 12000 frames per second, so multiple readings per image capture are possible (<https://datasheet.octopart.com/ADNS-9800-Avago-datasheet-10666463.pdf>). This would give an even more precise account of motor information while maintaining camera-behavior synchronization. Further, accumulated displacements can be stored in the sensors between readings, because ADNS-9800 sensors store motion data in 16 bits instead of the more standard 8 bits. Therefore, sensor saturation is not a concern at moderate sampling rates. This makes using these sensors simpler. Proper wiring is also simple and is demonstrated in Figure 2A. The connections demonstrated using dotted lines can be replaced with jumper wires or sturdier, longer lasting wire, as detailed in the *Methods*.

In an example experiment (Figure 3), we recorded a 10 minute long session of a mouse running on the spherical treadmill, and data was acquired at 20 Hz concomitantly with digital outputs that could be used to trigger a sCMOS camera or another device. We calculated the velocity of the mouse, which averaged 7.1 + 6.9 cm/s (+ std) with a maximum velocity of 47.0 cm/s, which is in general agreement with previous studies (see, for example, (Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007)). When we measured the digital output timing with the TDT RZ5D, we found that digital outputs are precise, with a near-perfect linear relationship and a small, 28.9us per second positive timing drift. Thus, the actual frequency was 19.999 Hz instead of 20 Hz. The linear model fit demonstrated a root mean squared error of 38.9 microseconds, indicating that the camera trigger has at least microsecond-level precision.

To confirm that the frequency of data acquisition and timing of the corresponding digital pulses didn’t affect the small timing drift, we repeated 5 minute recording sessions without a live mouse at 20, 50, and 100 Hz. These recordings used an identical script, except we embedded a 500 microsecond delay between the start and end of each digital pulse instead of a 1 millisecond delay. We found that the actual frequencies were 19.999, 49.999, and 99.997 Hz, respectively. These all equate to approximate time delays of 30 um per second, and thus timing drift is independent of the sampling rate. Together, these results demonstrate the temporal precision and accuracy of the Teensy in conjunction with the IntervalTimer function and our simple software implementation. In addition, its simple design underscores the ease with which the Teensy can be used for triggering synchronous frame-capture during long recording experiments while maintaining precise alignment of neuronal data with behavior.

*Mice Trace conditioning Learning behavioral experiment*

In the second experiment (Figure 1B and 2B), we constructed a Teensy-based setup for a trace conditioning learning experiment, where a mouse is trained to associate conditioned stimuli (tone and/or light) with a subsequent unconditioned stimulus (an eye puff) separated by a brief time window.

We wrote a short script for the Teensy to deliver a tone while turning on an LED and then trigger a puff, and state changes were synchronized to frame capture. 50 trials of 20 seconds length each were recorded, where sound and light output pins were programmed to turned on 11.1 seconds into each trial for 700 ms, and the pin used to generate the aversive puff stimulus was turned on at 12.05 seconds into each trial for 100 ms.

We recorded the timings of each digital output and compared them to the theoretical timings with samples spaced at exactly 50ms apart, as shown in Figure 4A. Similar to the observation in the locomotion experimental design, the measured timings were close to the theoretical timings, with a 33.4 microsecond delay per sample. Notably, concomitant control of audio, light, and puff pins did not appear to greatly alter either the slight timing delay or precision of the digital pulses directed at the sCMOS camera compared with the motor setup, which experienced a similar delay per sample (28.9 us). The root mean squared error for the model fit was 13.3 us, consistent with the motor setup in its microsecond-level precision.

In order to measure latency (Figure 4Bi and iii), we acquired the timing of the camera digital pulse, measured by the TDT system, that corresponds to the exact imaging frame start at which time the audio signal or puff signal was turned on. We then acquired the timing of either the puff pin digital pulse or the timing of the onset of the audio signal. In order to measure the onset of the audio signal, we took the raw analog recording and high-pass filtered the signal using a 6th-order Butterworth filter, a bandpass frequency of 1000 Hz, and a “zero-phase digital filter” (MATLAB command “filtfilt”). Then, we took the absolute value of the Hilbert transform of the filtered signal to acquire an amplitude envelope. After finding the amplitude envelope, we found those values that exceeded a value of 0.005. The first time point that the amplitude crossed this threshold was considered the tone onset, and the next time point that dropped below this threshold was considered the tone termination.

The measured tone latency was both precise and predictable: it averaged 7.6 + 0.9 milliseconds. Because of the consistency of the timing latency, it would be easy to adjust for this latency within the code, in this case by instantiating a change in signal amplitude 7.6 milliseconds earlier than the corresponding frame capture. The value that we observed was similar to the value of 6.9 + 0.9 milliseconds observed in a similar design utilizing a Teensy 3.2, where the Teensy was used to play a pre-recorded sound after stimulation by a Bpod behavioral control system (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018).

We also measured the length of the tone, which was highly accurate. It lasted for 700 + 1 ms, with a range of 2.9 ms, which is not different from the programmed tone length (700 ms). Other implementations of the Audio library could potentially offer even more precision, if so desired. For example, if one needed to utilize a precise sound sequence in an experiment, they could upload the sound sequence as a .wav file and utilize the Teensy to play the pre-recorded sound (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018). However, our design can be implemented very simply, utilizing code only within a single main script.

We further quantified sound onset latency, sound duration, puff latency, and puff duration. As shown in Figure 4Biii, the digital output for eye puff showed no delay from the theoretical time (mean latency = -0.004 + 0.012 (+ std) ms, range=0.04 ms seconds). The duration of the puff digital pulse was also both highly accurate and consistent, as shown in Figure 4Biv, and lasted 100.03+0.02 (+ std) ms over the course of the 50 trials, hardly differing from the expected duration of precisely 100ms.

Ultimately, the precisions of both our puff and sound output are comparable to expensive, available systems such as the Habitest Modular system in conjunction with Coulbourn Graphic State 4 software, which itself offers 1 ms precision (<http://www.coulbourn.com/v/vspfiles/assets/manuals/Graphic%20State%204%20Users%20Manual.pdf>) , making the Teensy a viable, inexpensive alternative that is also able to capture synchronous imaging data using our simple software design.

**Conclusion and Discussion**

We demonstrate the use of Teensy 3.2 microcontroller in integrating synchronous sCMOS camera image capture for two different behavioral experiments. In one novel experimental design, we utilized recently developed ADNS-9800 gaming sensors for precise and high speed locomotion tracking, along with simultaneous camera commands. In a second experiment, we commanded four devices with precise timing during a trace conditioning experiment. In both experiments, the timing of the Teensy interface was accurate to within approximately 30 us, and precise, as measured by the root mean square error of the model fits, to the level of microseconds. Software designs were straightforward. We developed two user-friendly software interfaces, experimental scripts and simple hardware designs for both experiments. Together, these software and hardware configurations can be immediately adopted for the designed behavioral experiments, or customized for other types of behavioral experiments, where camera-based imaging is desired during behavior. This platform is additionally low cost and can be easily scaled for parallel experiments across many animals, or further customized for various types of behavioral experiments.

We also demonstrate an implementation of a trace conditioning paradigm. In addition to requiring accurate alignment of imaging with behavior, operant conditioning paradigms need reliable stimulus timing. In this setting, repetition of stimulus and response must occur in a highly regular temporal fashion in order for a mouse to learn and in order for the neuronal response to be consistent. This illustrates the ability of the Teensy to orchestrate different classes of output—analog and digital, both long and short pulses—simultaneously and with high temporal accuracy while simultaneously and synchronously sending out regular digital pulses to control an sCMOS camera. As previously stated, a major advantage of the Teensy 3.2 over other microcontrollers such as the Arduino UNO is the fact that it can output a true analog signal. This opens a venue for many experimental additions, particularly the addition of sound, without the need of extra devices such as resistors and capacitors to create an analog-like signal. In addition, it has a built-in “Audio” library that simplifies sound synthesis, reading, and mixing, all at 44.1 kHz, which is stereo quality.

An important discovery during development of this system is the realization of a slight drift of the Teensy processing clock. This drift is linear, which makes it simple to calibrate out if actual microsecond accuracy compared with real world timing is essential. Further, it underscores the necessity of a central controller for precise acquisition and total experimental control. Synchronizing different devices only by a single pulse at the start of an experiment can lead to problems when trying to acquire motor output or deliver some experimental stimulus and examine cellular behavior with high temporal accuracy. Alternatively, initiating experimental events directly from a high-level source, such as directly from a PC, can introduce timing jitter due to the multitude of tasks that a PC must attend to at any given point in time. For example, a recent calcium imaging study in the striatum finds additional neurological structure related to motor activity on very short timescales, but finds only velocity correlated with neural activity on longer timescales (Markowitz, et al., 2018). This suggests that with sufficient timing jitter, correlations and information on short time scales could be missed, yielding erroneous conclusions.

As explained previously (D'Ausilio, 2012; Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018), using a microcontroller such as an Arduino or the Teensy 3.2 circumvents the issue of imprecise timing of behavioral recordings. In addition, synchronizing camera triggers with experimental events circumvents the need of post-hoc image alignment. In conclusion, the Teensy 3.2 enables a user to flexibly orchestrate experiments with synchronous behavioral monitoring and control with image capture. Additional timing functions, such as the “IntervalTimer”, make the Teensy 3.2 easier to program to deliver equally spaced, regular digital pulses needed for triggering image capture. Further, the Teensy 3.2 allows a user to generate analog signals. Last, the Teensy 3.2 utilizes the simple, user-friendly Arduino programming language. The precision and flexibility of the Teensy 3.2 microcontroller make this a user-friendly, easily adaptable, accurate, and precise tool for to utilize in different experimental designs that benefit from synchronous image capture and behavioral control and data acquisition.

**Figures**

**Figure 1.** Diagrams of the two experimental device setups using Teensy interface. A, a floating, spherical treadmill setup for locomotion recording (A), **A** This experimental design consists of a Teensy 3.2 connected to two ADNS-9800 sensors and a CMOS camera, via serial-peripheral interfaces and a coaxial cable via SMA connectors, respectively. Every 50 milliseconds, a digital pulse triggers the CMOS camera to capture an image while simultaneously acquiring motor data from both ADNS sensors and sending them via a USB to a PC. The PC initiates each experiment by sending serial data consisting of the length of the experiment and imaging frequency to the Teensy. **B** A tone/light and puff trace conditioning setup. This experimental design constitutes a classic classical-conditioning paradigm. The user specifies via MATLAB or via a different interface the length and number of experimental trials. This information is sent via a USB to the Teensy 3.2, which initiates the experiment. In each trial, the Teensy initiates a 9500 Hz tone at 44.1 kHz. These stimuli are followed by an air puff, also delivered via the Teensy. In order to generate a sound loud enough for the speaker, the Teensy is soldered to a prop-shield, which contains an amplifier. The Teensy 3.2 sends time stamps, trial, and stimulus information via the USB back to the PC.

**Figure 2.** Detailed electrical schematics for both the motor-control and tone-puff systems **A.** A schematic demonstrating the wiring connections between a Teensy 3.2, prop shield, and an external speaker. Dotted lines indicate solid connections. All connections between the Teensy 3.2 and prop shield were made using 14x1 double insulated pins, and the output to the speaker from the prop shield was made using regular wire and a coaxial cable. Some extraneous and unused pins on the Teensy and the prop shield were not included in this diagram. **B.** A schematic demonstrating the wiring of Teensy to two ADNS-9800 sensors via serial peripheral interface connections (SPIs). Solid dots at intersections between lines indicate connections. Some unused pins on the Teensy 3.2 were not included in this schematic.

**Figure 3.** Example recording using the motor control setup **A.** Part of a sample 10 minute recording session during which a head-fixed animal was allowed to run on the three-dimensional treadmill. Shown in **Figure 1A**. **B.** Times of digital pulses sent by the Teensy 3.2 as measured internally by the Teensy, vs times of the digital pulses as measured by an external device. Red indicates linear model, and in black are experimental data, down-sampled by a factor of 200. The linear model estimates a slope of 1.000028927 + .000000005 (t(11997)= 2.0381e+08, p < 0.001, R2=1; intercept = 0.000107 + 0.000002, t(11997) = 63.243, p < 0.001), indicating an excellent fit and very nearly a 1:1 correspondence of time stamps.

**Figure 4.** Example recording using the tone-puff setup **A.** Timing of the digital pulses as measured by the Teensy 3.2 in the tone/light-puff setup versus timing as measured by an external device. Linear model fit is shown in red, and in black are experimental data down-sampled by a factor of 200 for visualization. These measurements have a correspondence near 1:1 (R2=1, slope: 1.0000334 + 0 (to machine precision), t(14998)=infinite, p<0.001). **B.** Timing measured by the Teensy for (i) and by the TDT system for (ii-iv) over the course of fifty trials; (i) shows the latency between the theoretical onset of the tone and the measured timing of the tone as measured by the TDT device (mean=7.6 + 0.9 ms, range=2.9 ms); (ii) shows the consistency of the length of tone intervals across all trials (mean=700 + 1 ms, range=2.9 ms); (iii) shows the consistency of the latency of the puff stimulus, as measured by the TDT sytem (mean= -0.004 + 0.012 ms, range=0.04 ms); (iv) shows the consistency of the length of the puff across all trials (mean = 100.03+0.02 ms). (all + std).

**Tables**

Table 1. Specialty components necessary to build a motor output system

|  |  |  |  |
| --- | --- | --- | --- |
| **Part name** | **Website** | **Part number** | **Cost per unit** |
| Teensy 3.2 | https://www.pjrc.com/store/teensy32.html | TEENSY32 | $19.80 |
| ADNS-9800 sensors | https://www.tindie.com/products/jkicklighter/adns-9800-laser-motion-sensor/ | ADNS-9800 Laser Motion Sensor | $27.50 |

Table 2. Specialty components necessary to build a tone-puff system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Part name** | **Website** | **Part number** | **Cost per unit** |
| Teensy 3.2 | https://www.pjrc.com/store/teensy32.html | TEENSY32 | $19.80 |
| 14x1 Double insulator pins | https://www.pjrc.com/store/header\_14x1\_d.html | HEADER\_14x1\_D | $0.85 |
| Prop shield | https://www.pjrc.com/store/prop\_shield.html | PROP\_SHIELD | $19.50 |

# References

Aranov, D., & Tank, D. W. (2014). Engagement of Neural Circuits Underlying 2D Spatial Navigation in a Rodent Virtual Reality System. *Neuron, 84*(2), 442-456.

Barbera, G., Liang, B., Zhang, L., Gerfen, C. R., Culurciello, E., Chen, R., . . . Lin, D.-T. (2016, October 5). Spatially Compact Neural Clusters in the Dorsal Striatum Encode Locomotion Relevant Information. *Neuron, 92*(1), 202-213.

Chen, X., & Li, H. (2017, December). ArControl: An Arduino-Based Comprehensive Behavioral Platform with Real-Time Performance. *Frontiers in Behavioral Neuroscience, 11*, 1-9.

D'Ausilio, A. (2012). Arduino: A Low-Cost Multipurpose Lab Equipment. *Behavior Research Methods, 44*(2), 305-313.

Dombeck, D., Khabbaz, A. N., Collman, F., Adelman, T. L., & Tank, D. W. (2007). Imaging Large-Scale Neural Activity with Cellular Resolution in Awake, Mobile Mice. *Neuron, 56*(1), 43-57.

Klaus, A., Martins, G. J., Paixao, V. B., Zhou, P., Paninski, L., & Costa, R. M. (2017). The Spatiotemporal Organization of the Striatum Encodes Action Space. *Neuron, 95*(5), 1171-1180.e7.

Markowitz, J. E., Gillis, W. F., Beron, C. C., Neufeld, S. Q., Robertson, K., Bhagat, N. D., . . . Datta, S. R. (2018). The Striatum Organizes 3D Behavior via Moment-to-Moment Action Selection. *Cell, 174*(1), 44-58.e17.

Micallef, A. H., Takahashi, N., Larkum, M. E., & Palmer, L. M. (2017, May). A Reward-Based Behavioral Platform to Measure Neural Activity during Head-Fixed Behavior. *Frontiers in Cellular Neuroscience*, 1-8.

Mohammed, A. I., Gritton, H. J., Tseng, H.-a., Bucklin, M. E., Yao, Z., & Han, X. (2016). An Integrative Approach for Analyzing Hundreds of Neurons in Task Performing Mice Using Wide-Field Calcium Imaging. *Scientific Reports, 6*, 20986.

Solari, N., Sviatkó, K., Laszlovsky, T., Hegedüs, P., & Hangya, B. (2018). Open Source Tools for Temporally Controlled Rodent Behavior Suitable for Electrophysiology and Optogenetic Manipulations. *Frontiers in Systems Neuroscience, 12*(May).

Yoav, A., Kim, J. J., Brinks, D., Lou, S., Wu, H., Mostajo-Radji, M. A., . . . Cohen, A. E. (2018). All-Optical Electrophysiology Reveals Brain-State Dependent Changes in Hippocampal Subthreshold Dynamics and Excitability. *bioRxiv*. doi:https://doi.org/10.1101/281618