A Teensy microcontroller based interface for data acquisition and behavioral monitoring in systems neuroscience research

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

Abstract

Systems neuroscience research aimed at understanding neural mechanisms of behavior requires precisely timed data acquisition and behavioral monitoring. While many commercial systems have been designed to meet these needs, they often fail to offer flexibility in experimental designs or to allow integration of novel neuroscience technologies. We here describe a Teensy microcontroller-based interface capable of fast and precisely timed digital signal readout for data acquisition and analog output to control behavioral experiments. We demonstrate the efficacy and temporal precision of the Teensy based interface in two experimental settings with different demands: reliable, high-accuracy motion sensing and analog output delivered synchronously with digital pulses for image capture. We then test the theoretical specifications and show the temporal accuracy and precision of this device in both of these settings. Both setups are accurate to the order of tens of microseconds per sample, and are precise to tens or hundreds of nanoseconds per sample. We conclude that the Teensy 3.2, in conjunction with specific sensors or shields, provides an optimal form of experimental control, particularly for those interested in neuronal imaging.

Introduction

Neuronal imaging is a burgeoning technique that demands high temporal fidelity. For example, new voltage imaging techniques utilize sampling rates up to 1 kHz (Yoav, et al., 2018). Strict alignment of neuronal signals with experimental inputs or outputs is essential (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018). A new challenge is to find flexible, intuitive, and accurate devices that allow for concomitant execution and recording of experimental paradigms in a way that is synchronized with imaging, particularly with imaging devices that utilize new, higher-frequency acquisition rates. Of recent interest include examining the motor output of mice while imaging (Klaus, et al., 2017; Barbera, et al., 2016), and imaging during trace conditioning (Muhammad et al, 2015).

Imaging experiments that examine the neural basis of behavior typically require precisely timed data acquisition and command signals, as noted previously for electrophysiology recordings (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018). Behavioral data must be precisely collected with respect to imaging data. For example, a recent study in the striatum finds additional neurological structure with respect to motor activity on very short timescales, suggesting that poor timing resolution could potentially lead to incorrect inferences (Markowitz, et al., 2018). A common imaging technique is to set up an imaging device to utilize an “external trigger”, where the rising phase of a digital pulse or TTL pulse either initiates a sequence of internally clocked image captures (Micallef, Takahashi, Larkum, & Palmer, 2017) or initiates frame capture once for every pulse. Microcontrollers such as the Teensy 3.2 or Arduino UNO are capable of delivering such pulses and thereby controlling the timing of image capture. Initiating the start of an imaging sequence via an Arduino device has been previously shown (Micallef, Takahashi, Larkum, & Palmer, 2017). However, a limitation of this approach is that it is necessary to synchronize frame timing with behavioral data after the experiment is complete, which is inexact and may necessitate interpolation. Alternatively, if one were to trigger each frame based on a different digital pulse, substantial jitter in digital pulse delivery can cause frame loss and can also necessitate interpolation for many statistical analyses. Thus, we need a device capable of delivering continuous, precisely timed digital pulses that can synchronize other experimental events with camera onsets. The Teensy 3.2 fulfills these requirements.

In addition to 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. Initiating experimental events 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. Further, with concomitant imaging, one must also align tasks to imaging data after the fact, or face substantial variability in frame spacing. As explained previously (D'Ausilio, 2012; Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018), using a microcontroller such as an Arduino or Teensy 3.2 circumvents the issue of imprecise timing of behavioral events. We note that in addition, synchronizing camera triggers with experimental events circumvents the need of post-hoc image alignment.

Finally, interfaces for managing simultaneous behavior and imaging must be flexible and allow for changes in experimental design. The Teensy 3.2 is easy to program for any particular need. The Arduino programming environment, which Teensy utilizes, is simple to learn for anyone with any programming background, as explained in depth previously (D'Ausilio, 2012). In addition to the standard features that the Arduino UNO, for example, offers, the Teensy 3.2 delivers true analog output. A library available only for the Teensy, the Audio library, makes use of this by providing a simple way to create and/or play sounds directly from the Teensy. Therefore, operant conditioning experiments that utilize sound don't necessitate additional equipment, aside from an inexpensive amplifier for microcontrollers and a speaker.

Here, we demonstrate in two simple experimental paradigms that the Teensy 3.2 is indeed a simple, flexible device capable of coordinating highly accurate data acquisition, and sound and stimulus delivery with image capture. It is capable of keeping highly accurate and low-bias timing that allow it to reliably instantiate frame capture with highly regular intervals while delivering stimuli or recording experimental data with microsecond-level precision. Further, as a universal experimental controller, it is capable of automatically aligning experimental data with image capture, removing the need for post-hoc interpolation or time-alignment between different external devices.

Methods

In order to demonstrate the flexibility of the Teensy 3.2 in designing precisely timed imaging-based experiments, we designed two experimental setups: one that utilizes motion tracking on a floating Styrofoam ball while delivering regular digital pulses at a typical image-capturing frequency, and one that utilizes the Teensy's Audio library in a trace conditioning paradigm while again delivering regular digital pulses.

Motor acquisition experiment

The overall design for this experiment is shown in Figure 1A. Two ADNS-9800 gaming sensor boards (https://www.tindie.com/products/jkicklighter/adns-9800-laser-motion-sensor/) were attached at the equator of a 3D-printed half-sphere in which a large, buoyant Styrofoam ball is floated by house air. These sensors lay at an angle of approximately 75 degrees from one another. This setup is adapted from that of Dombeck, Khabbaz, Collman, Adelman, & Tank (2007). The wiring of the two ADNS-9800 sensors to a Teensy 3.2 is demonstrated in Figure 2A.

To compute linear velocity, we used the y-readings of both sensors, and the rotational velocity can be computed using the x-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. The Teensy was connected to a PC using a USB-microUSB cable.

Due to the complexity of extracting motion data from these sensors, we utilized simple classes and functions that are freely available on Github (https://github.com/markbucklin/NavigationSensor) and abstract the complexity of acquiring motion data to a user-friendly level. In particular, this repository contains the ADNS9800 library, which is a modified version of the stock ADNS-9800 library (https://github.com/mrjohnk/ADNS-9800).

We modified the specific-use case of the library available in this repository to acquire data and send digital pulses every 50 milliseconds. In order to precisely time these events, we utilized the "IntervalTimer" function available in the standard Teensy library. This allows for microsecond-level precision in calling different functions using interrupts. Here, we used it to call a main function that sends out a digital pulse to trigger a frame capture, collects data from the two ADNS-9800 sensors, and sends the motion data to a main computer. The Teensy also has the very useful "ellapsedMicros" and "ellapsedMillis" libraries built in to the Teensyduino library, which, to microsecond or millisecond accuracy, respectively, act as time accumulators. These can also be used for precisely timing events as well. Though these can be downloaded separately for the Arduino, they come preinstalled in the Teensyduino library.

Via the ADNS9800 library, we read from the "motion burst" register of each sensor. On every call to the main function, we acquired the accumulated displacement over the previous 50 milliseconds in both the x and y directions. For the counts per inch setting we used a value of 3400 counts per inch, the default setting. As previously mentioned, during this interrupt, a digital "on" pulse that lasts for approximately 1 ms is sent out of a digital pin using the DigitalIO library (https://github.com/greiman/DigitalIO). This library allows us to use the functions "fastPinMode" and "fastDigitalWrite', for example, which reduce the latency introduced by turning pins on, off, or setting their "mode" (to INPUT or OUTPUT, for example). 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.

In order to begin experiments with the Teensy, after the main script was uploaded to the Teensy, we wrote a simple MATLAB-based graphical user interface that can be used on a desktop or laptop connected via a USB to the Teensy. In principle, however, this graphical user interface could be written in Python or any other programming language. Using this interface, the user enters the length of the experiment and the frequency of data acquisition. This frequency will determine the frequency with which digital pulses are sent to notify an external device such as a CMOS camera to capture an image, for example, and also determine the frequency with which accumulated motor information will be recorded by the PC. The PC or laptop sends this information over a serial connection to the Teensy utilizing a bidirectional microUSB-USB cable.

In a proof-of-concept experiment (Figure 3), we recorded a 10 minute long session of a mouse running on a 3 dimensional treadmill (Styrofoam ball floating on air), and data was acquired at 20 Hz concomitant with digital pulses that could be used to trigger a camera image capture or a different device. The mouse's speed was computed using the y-coordinates of each ADNS-9800 sensor, and the total distance travelled at any one time point was computed using the following equation:

$$dista\ \ ^{n} = \sqrt{\frac{y_R^2 + y_L^2 - 2y_L y_R \cos\ \theta}{sin^2\theta}}$$

Where y_R and y_L are the y readings from the left and right sensors, and $\boldsymbol{\theta}$ is the angle between the two sensors (75 degrees). Velocity was computed as the distance divided by the time between two adjacent frames. Times and distances travelled were recorded by the Teensy 3.2, and the timings of digital pulses were measured by an external device at 3051.76 Hz (Tucker-Davis Technologies RZ5D (TDT RZ5D)).

After analyzing the time stamps acquired by the TDT RZ5D system, we noticed that there was a very small timing drift (approximately 30 microseconds per second). To confirm that the frequency of data acquisition and timing of the corresponding digital pulses didn't affect this 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 the digital pulse ("delayMicroseconds(500)") instead of a 1 millisecond delay ("delay(1)").

Classical conditioning experiment

To illustrate another simple experimental design wherein the Teensy 3.2 can be used to control four devices simultaneously while reliably outputting a sound, we created a trace conditioning experimental design. The general setup is shown in Figure 1B. In a trace conditioning experiment utilizing this setup, a head-fixed mouse would theoretically be exposed to a 9500 Hz tone concomitantly with a light stimulus. After, the mouse would receive a puff of air in its eyes. The goal is to train the mouse to blink upon exposure to the unconditioned stimuli.

In order to amplify the sound to a suitable volume, we added a "prop shield" to the Teensy. The prop shield is a very affordable, easy-to-use add-on that is capable of amplifying the analog output signal (shown in Figure 2B as pin A14). If stereo output were desired, the manufacturer also offers a true audio shield (https://www.pjrc.com/store/teensy3_audio.html) that is capable of stereo output, as demonstrated previously (Solari, Sviatkó, Laszlovsky, Hegedüs, & Hangya, 2018).

To attach the prop shield to the Teensy 3.2, we used 14x1 double insulator pins, and then fed the output to a speaker, as demonstrated in Figure 2B. The prop shield can power speakers with resistances up to 8 ohms (https://www.pjrc.com/store/prop_shield.html). We also directed digital outputs from the Teensy to activate a light concomitant with the sound, and a puff as an aversive stimulus following each sound/light combination. Meanwhile, digital pulses were programmed to occur during every frame, which could be used to trigger image captures, for example.

Again, we utilized an "IntervalTimer" in order to reliably time all of the experimental events. Every 50 ms, this interval timer called a main function that updated the status of the digital pins associated with the "puff" and the light, and updated the amplitude of the 9500 Hz sine wave (amplitudes were set to 0.05 during audio stimulus time periods, and 0 elsewhere). Also, at the termination of a trial, this function incremented the trial number. Finally, a 1 ms digital pulse was delivered via another pin to instantiate a theoretical camera trigger. The speaker, camera, puff, and light source can be 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.

In order to begin experiments with the Teensy, we wrote in MATLAB a simple graphical user interface that can be used on a desktop or laptop. With this, a user enters the length of the each trial and the number of trials desired. The PC or laptop sends this information over a USB connection to the Teensy, which in turn reports information about the experiment, in particular the frames during which the tone is on, the puff is on, or the light is on, and the experimental and trial times (in milliseconds) at the beginning of each IntervalTimer function call.

In our proof-of-concept experiment (Figure 3), the puff, light, and camera trigger pins were all attached to and were recorded by the same external device (TDT RZ5D) at 3051.76 Hz. We performed a mock-recording consisting of 50 trials of 15 seconds length each, where sound and light output pins were turned on 11.1 seconds into each trial for 700 ms, and the pin used to generate the aversive puff stimulus was turned at 12.05 seconds into each trial for 100 ms. Output from the puff, light, and camera pins were recorded by an external device at 3051.76 Hz.

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/Discussion

Low cost microcontrollers such as Arduino UNOs, with their user friendly interface and low cost, have gained popularity in neuroscience research (D'Ausilio, 2012; Chen & Li, 2017; Micallef, Takahashi, Larkum, & Palmer, 2017). However, the Arduino UNO is somewhat limited, in that it does not have true analog output. Further, while the Arduino UNO has several useful timing libraries, it lacks the IntervalTimer function, which in particular is optimal for precise control of experiments and precise acquisition of experimental data. The Teensy 3.2 (https://www.pjrc.com/store/teensy32.html) is a newly developed microcontroller that not only has analog output and a comprehensive audio library, but also has the capability to use an IntervalTimer. Therefore, to maximize the flexibility of experimental design and maintain high accuracy, we utilized the Teensy 3.2 instead of the Arduino UNO.

To demonstrate the flexibility of this device for both experimental control and data acquisition, we constructed two separate and commonly utilized experimental setups both built upon a Teensy 3.2. In the first (Figure 1Ai and 1Aii), we constructed a device that monitors and records motor data at a fixed interval and is capable of simultaneously delivering highly regular, brief digital pulses to an external device such as a scientific CMOS camera. As shown in Table 2, the cost of specialty components for this experimental design is quite low, totaling less than $80 total. Other commonly used components such as wiring, solder and wire strippers and crimpers are also needed on a case-by-case basis and are listed in Table 3, but are widely available and in many cases such as a lab setting be available for use.

Motion tracking using the ADNS-9800

We first introduced a system for imaging and simultaneous motion three-dimensional treadmill tracking that necessitates only a Teensy 3.2 microcontroller and two ADNS-9800 laser motion sensor boards. There are a number of ways in which people have attempted to observe motor output while imaging. In one particular technique, experimenters mount a fluorescence microscope on the head of a mouse, and allow the mouse to move freely while recording activity via video (Barbera et al. 2016) or via video in addition to an accelerometer (Klaus, et al., 2017). However, resting a microscope on the head of a mouse restricts its normal range of movement, limiting its peak velocity and introducing a confounding variable to the experiment, particularly when examining motion-related regions of the brain such as the striatum.

Another technique employs a "three-dimensional treadmill" setup, initially proposed by Dombeck, Khabbaz, Collman, Adelman, & Tank (2007) and utilized elsewhere (Aranov & Tank, 2014). In this setting, the mouse is fitted with a head plate and imaging window, and is suspended atop a Styrofoam ball

that is supported by compressed air (Figure 1). This type of imaging offers easily correctable in-plane jitter, as well as a setting in which mouse must apply similar forces to begin or to terminate a motor sequence as it would in a freely-moving setting (Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007). Generally, two computer mice are fit at the equator of the Styrofoam ball at an angle of 90 degrees, which provides the experimenter with linear movement in the X-Y plane, as well as rotational information. Such designs can obtain motor information from readings from the computer mice via LabView (Aranov & Tank, 2014; Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007) which, though a comprehensive piece of software, is expensive and proprietary.

We reconstructed this latter design utilizing a Teensy, a far less expensive alternative. Our approach also allowed us to interface directly with ADNS-9800 sensor boards. These sensor boards are inexpensive and the sensors themselves are an improvement in many aspects over the sensors in standard computer mice. For example, they are highly sensitive and have high maximum sampling rates, with a maximum read rate of 12000 frames per second (thus accommodating the temporal requirements of faster imaging environments), and maximum resolution of 8200 counts per inch (https://datasheet.octopart.com/ADNS-9800-Avago-datasheet-10666463.pdf). 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, one does not need to worry about sensor saturation.

In order to use these motion-sensors, we utilized a class-based ADNS-9800 library. We read displacements picked up by the sensors and converted them directly to micrometer displacements using the internal calibration of the sensors. Because of the simplicity of the ADNS-9800 library and example experimental design setup built alongside, building a usable design is particularly easy. This is particularly true if one is interested mostly in recording accurate x, y, and rotational displacements, which are already implemented directly into the code. Proper wiring is also simple and is demonstrated in Figure 2B. The connections demonstrated using dotted lines can be replaced with jumper wires or sturdier, longer lasting wire.

This system offers an inexpensive method of tracking mouse movement with high fidelity, temporal accuracy and without introducing confounding experimental variables. As can be seen in Figure 3A, the velocity that we calculate falls into the range of previously reported mouse velocity with similar setups (see, for example, (Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007)), and we are capable of seeing large variation in the mouse's motor output. In Figure 3B, we also see that digital pulses administered at 50 ms increments closely track the theoretical times, biased in slope by an exceedingly small amount (approximately 28.9 microseconds per sample).

To verify that this bias in slope was not due to the frequency of the IntervalTimer, we repeated recordings that were 5 minutes long each, each using the same script except with a 500 microsecond delay between the beginning of the digital pulse and end of the digital pulse. These all had very similar biases, at 28.3 microseconds per second for the 20 Hz recording, and 28.4 microseconds per second for the 50 Hz and 100 Hz recordings, respectively. A much similar bias in timing was previously reported previously using an Arduino UNO: with repeated sampling of single 900 ms long TTL pulses with 100 ms inter-pulse intervals, the average length of time between sequential pulses was 1000.6 milliseconds (D'Ausilio, 2012). Though the code utilized by that experiment differs from ours, it does illustrate the precision and low bias of the Teensy combined with the IntervalTimer function. In addition, it underscores the utility of the Teensy for continual frame-capture triggering instead of aligning a camera only to the beginning of a trial or experiment, particularly over the course of a longer recording session (Micallef, Takahashi, Larkum, & Palmer, 2017).

Trace conditioning

In the second experiment (Figure 1B and 2B), we constructed a device capable of running a simple trace conditioning experiment, where we can train a mouse to blink in response to simultaneous tone and light exposure by using a puff of air as an unconditioned aversive stimulus. Our design of a trace conditioning experiment mimics a setup previously reported by our lab (Mohammed, et al., 2016). Typically in this experimental setup, a mouse is gradually trained to blink after seeing a light and hearing a sound, via a "puff" that is consistently delivered following exposure to both light and a 9500 Hz tone. Here, we set up the Teensy to perform such an experiment, and recorded from the relevant pins. In addition to the Teensy 3.2, we needed only 2 additional specialty components in addition to a speaker, as shown in Table 1: a prop shield to amplify the analog output from the Teensy 3.2, which can then drive speakers of both 4 and 8 ohms, and a few sets of 14x1 double insulated pins for connecting the Teensy to the prop shield. In total, this setup costs approximately $40, excluding general equipment.

Imaging can be performed simultaneously by turning on and off a given pin during each frame. In this mimic experiment, we recorded the timings of each of these triggers and compared them to the theoretical timings with samples spaced at exactly 50ms apart, as shown in Figure 4A. Like the motion experimental design, the measured timings were very similar to the theoretical timings, biased by approximately 30 microseconds per sample. We looked at light onset timing, light length, interstimulus length, and puff length in Figure 4B as well. All were very consistent over the 50 trials, with standard deviations well under 1 milliseconds, showing that including a continuous audio output doesn't alter the accuracy or increase the bias of experimentation with a Teensy.

Conclusion

We introduce two inexpensive and highly accurate experimental paradigms both constructed around a Teensy 3.2 microcontroller. In the first, we utilize ADNS-9800 gaming sensors, which obviate the need for external calibration, and for which exists a user-friendly library and example implementation of this library. The Teensy is capable of performing this task while sending temporally regular and precise digital pulses out of another digital pin. This would be particularly useful in an imaging paradigm, where one could set a camera to capture and send motor output simultaneously with accurate camera triggers.

We also demonstrate a setup built to implement a trace conditioning paradigm. 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 sending out regular digital pulses to control an image capturing device. It also highlights the ability of this device to simultaneously produce an analog output, in particular to generate a sound, while performing other actions. As previously stated, a major advantage of the Teensy 3.2 over other microcontrollers such as the Arduino is the fact that it can output a true analog signal, whereas the Arduino Uno, for example, is capable only of outputting pulse-width modulated signals. 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. Rather, the Teensy 3.2 simply needs to be soldered on to a prop shield, and less in-depth knowledge about electronic circuits is necessary. In addition, it has a built-in "Audio" library that simplifies sound synthesis, reading, and mixing, all at 44.1 kHz.

A potential limitation of this system that we saw was the slight timing drift of the Teensy. This drift is linear in nature, however, which makes it simple to calibrate out. Further, it underscores the desirability of using a Teensy for 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. We note as well that the standard errors of our measurements across both linear models were very small: on the order of tens of nanoseconds. In conclusion, the precision and utility of the Teensy microcontroller, in conjunction with the ADNS-9800 sensors and available audio library and IntervalTimer functions, make this a user-friendly, easily adaptable, accurate, and precise tool for different experimental designs in neuroscience in general, and particularly for imaging studies.

Figures

Figure 1. Diagrams of the two experimental device setups, a floating, 3D treadmill with two sensors for recording motor output (A) and a tone/light and puff classical conditioning setup. **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** 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 while turning on a light. 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. 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. 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**. The mouse's average speed was 7.1 $\pm$ 6.9 cm/s, with a maximum velocity of 47.0 cm/s, within ranges reported elsewhere. **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. Green 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 $\pm$ .000000005 ($t(11997)= 2.0381e+08$, $p < 0.001$, $R^2=1$; intercept = 0.000107 $\pm$ 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. 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. These measurements have a correspondence near 1:1 ($R^2=1$, slope: 1.0000336 $\pm$ 4e-10, $t(14998)=2.35e+09$, $p<0.001$). **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 consistency of light onsets across all trials (mean=11.0999930±0.0000009 seconds); (ii) shows the consistency of the length of "light on" intervals across all trials (mean= 0.700046±0.000006 seconds); (iii) shows the consistency of the length of the conditioned-unconditioned stimulus interval (mean= 0.24999 ±0.00002 seconds); (iv) shows the consistency of the length of the puff across all trials (mean= 0.10003±0.00002 seconds). (all ± std).

Tables

Table 1. Specialty components necessary to build a tone/light-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

Table 2. 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/	None	$27.50

Table 3. General components necessary for work with both setups

Miscellaneous parts
SMA connectors
SMA coaxial cables
Solid wire (22 gauge used)
Soldering iron
Solder
Crimping tool
Wire stripper
Dupont connectors and housing
Male and female pin headers

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., Sviatk\o, K., Laszlovsky, T., Heged\us, 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