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 control, 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 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 delivering repeated digital pulses that can be used to control image acquisition from a sCMOS camera. In another example, we used the Teensy interface to control temporally precise delivery of auditory and visual stimuli in a trace conditioning behavioral paradigm, while delivering repeated digital pulses to initiate camera image acquisition. These examples demonstrate that the Teensy interface, consisting of a Teensy 3.2 and custom software functions, provides a low-cost and flexible platform to integrate a sCMOS camera into behavioral experiments, 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 of neural network analysis during 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 studies, 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 or other PC-based programs can lead to undesired temporal delays, as the PC operating system needs to balance the demands of many system operations at once.

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, and capable of delivering digital outputs with microsecond time precision, while using user-friendly, open-source software functions. The Arduino was the first major microcontroller to gain substantial popularity. Recently, Teensy microcontrollers were developed, which have all the key features of Arduino microcontrollers, as well as the additional feature of delivering analog output. Teensy’s utilize the same open-source Arduino software environment, and thus they are easy to program (D'Ausilio, 2012). For example, Arduino devices have recently been integrated into two-photon imaging experiments (Wilms & Häusser, 2015; Takahashi, Oertner, Hegemann, & Larkum, 2016). In one study, an Arduino was used to generate a digital command to encode the duration of an image sequence (Micallef, Takahashi, Larkum, & Palmer, 2017). However, triggering the onset of an image sequence could lead to undesired temporal delays for individual image frames within the image sequence. One way to perform precisely timed image acquisition of each image frame is to trigger the acquisition of each frame independently, while simultaneously acquiring behavioral data. To do this, one can use the external trigger setting of the camera as demonstrated previously (Micallef, Takahashi, Larkum, & Palmer, 2017). Because of the simplicity of microcontrollers and their temporal precisions, microcontroller thus represent an attractive solution to precisely record digital data and monitor experimental progress.

Here, we demonstrate and characterize a flexible Teensy interface for temporally precise data acquisition and delivery of analog and digital signals, during a voluntary movement tracking experiment and a trace conditioning learning experiment. The Teensy interface can deliver digital pulses with microsecond precision to initiate individual image frame capture at a desired frequency, while simultaneously collecting animal behavioral data. We also demonstrate the ability of the Teensy interface to generate analog sound waveforms to drive a speaker for an auditory experiment. Together, these results demonstrate that the Teensy interface, consisting of a Teensy microcontroller and a set of custom software functions, offers a flexible, accurate, and user friendly environment for imaging experiments during behavior.

**Methods**

*Construction of Teensy boards*

The two experimental designs are shown in Figure 1. The specialty components required to build these designs are shown in Tables 1 and 2. In both experiments, a Teensy 3.2 (PJRC.COM, LLC, part #: TEENSY32) (Figure 1A), or a Teensy 3.2 soldered to a prop shield (PJRC.COM, part #: PROP\_SHIELD) (Figure 1B), is mounted on top of a printed circuit board via standard female headers (such as SparkFun Electronics, PRT-00115). Female headers were then soldered to the PCB for stability. Output from the Teensy was directed from pins on the female headers to standard SMA connectors (such as: Digi-Key, part # CON-SMA-EDGE-S-ND) via 22 gauge wires (for example: Digi-Key, part #1528-1743-ND). Coaxial cables were then attached the SMA connectors to connect the Teensy to external devices. The Teensy was connected to a computer via a standard USB-microUSB cable (for example: Digi-Key, part # AE11229-ND). To easily upload code to the Teensy, we used PlatformIO (<https://platformio.org/>), an add-on to the widely-used Atom text editor (<https://atom.io/>), instead of the default Arduino programming environment. To turn digital pins on and off, and also to change their modes to either “input” or “output”, we used the DigitalIO library (<https://github.com/greiman/DigitalIO>), which decreases the amount of time spent performing each of these actions. To easily set experiment specific parameters for the Teensy, such as the sampling frequency, the trial number and the length of an experiment, we developed a simple MATLAB graphical user interface.

Motion tracking *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 can be used to trigger a sCMOS camera for image capture every 50 ms. The overall design of this experiment is shown in Figure 1A. A mouse was 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. For the counts per inch setting of the sensor, which determines the sensitivity of the sensors to external movement, we used a value of 3400 counts per inch. Thus, the total distance travelled by the mouse at any 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). We also acquired readings in the “x” direction from both sensors, which can be used to calculate rotation. Velocity was computed as the distance divided by the time between two subsequent readings. These two sensors were connected to a Teensy via simple serial peripheral interface (SPI) connections with insulated 22 gauge wires as shown in Figure 2A.

To control experimental timing, we utilized the “IntervalTimer” function unique to the standard Teensy library, which can repeatedly call a function at specified intervals. Using IntervalTimer, we repeatedly called a function that send the accumulated displacement of the motion sensor readings to the attached PC. We acquired the x and y displacement readings from each sensor with freely available functions on Github (<https://github.com/markbucklin/NavigationSensor>), which read accumulated displacement from the “motion burst” register of each sensor, from the previous function call.. After reading the motion sensor, a digital “on” pulse that lasted for 1 ms was sent out of a digital pin designed to initiate an image frame capture from a sCMOS camera. To characterize the temporal precision of different digital pulses generated by the custom scripts using the “IntervalTimer” function, we recorded the digital outputs with a commercial system (Tucker Davis Technologies RZ5D (TDT RZ5D)) at 3051.76 Hz.

*Trace eye blink conditioning experiment*

In this experiment, a Teensy was programmed to deliver outputs capable of eliciting a sound, turning on an LED light and initiating an eye puff, while delivering digital pulses that can be used to trigger a sCMOS camera for image capture every 50 ms The overall design of this experiment is shown in Figure 1B. To deliver an audible sound through the Teensy, we used a Teensy prop shield module (PJRC.COM, LLC., part #: PROP\_SHIELD) to amplify analog output (shown in Figure 2B as pin A14). This add-on component can drive speakers with resistances up to 8 ohms. The prop shield was soldered to the bottom of a Teensy with 14x1 double insulator pins (PJRC.COM, LLC., part #: HEADER\_14x1\_D), and the output was connected to a speaker. The Teensy was then mounted onto the female headers separated by the prop shield, as shown in Figure 1B. The speaker, camera, and air valve for the eye puff were attached to the microcontroller through coaxial cables (Figure 1B).

We used the Teensy Audio library function “AudioSynthWaveformSine” to generate tones. This function continuously outputs a sine wave with a sampling rate of 44.1 kHz from the analog pin. We first initialized the tone, such as a 9500 Hz sine wave used here, at the beginning of each experiment, but set the amplitude to “0”, so that the tone was off. At desired time, we switched the amplitude to 0.05 or 0 (out of a maximum of 1) to generate an audible tone. , . The value of 0.05 generated a tone of approximately 75 dB with our amplifier and speaker settings.

We used the “ElapsedMicros” function to control the timing and the frequency of the experiment. ”ElapsedMicro” updated the status of the digital pins controlling the eye puff stimulus and the LED light, and the amplitude of the sine wave tone stimuli. Immediately following each update, a digital “on” pulse that lasted for 1 ms was sent out of a digital pin designed to initiate an image frame capture from a sCMOS camera. Then, a timer was allowed to elapse for 30s?, the intertrial interval, and after which the function was repeated through a while loop? At the completion of each trial, this single function also initiated the following trial or signaled to terminate the experiment.

To characterize the temporal precision of different digital pulses generated by the custom scripts using the “e….” function, we recorded the digital outputs with a commercial system (Tucker Davis Technologies RZ5D (TDT RZ5D)) at 3051.76 Hz, and the analog sinewave output at 24414.0625 Hz without additional amplification. To determine tamplified analog output from Teensy was first at 1kHz , zero-phaseW thenperformed the absolute at what freq???whererose above 0.005 onset of the analog signal, subsequent where it 0.005 offset.

*Statistics*

Statistics were performed in Matlab. Linear models were constructed using the “fitlm” function in MATLAB 2017b.

**Results**

Microcontrollers such as Arduinos have gained popularity in neuroscience research due to their user-friendly interface, open-source software environment, and their flexibility of device integration (D'Ausilio, 2012; Chen & Li, 2017; Micallef, Takahashi, Larkum, & Palmer, 2017). Recently, Teensy 3.2 has been developed, which has analog outputs, a major improvement over the popular Arduinos. Teensy also has a comprehensive Audio library, as well as the IntervalTimer function, generating precisely timed events. Here, we present a Teensy-based interface to integrate frame-by-frame image capture with behavioral experimental control and data acquisition.

*Mice motion tracking experiment*

In this example experiment (Figure 3A), we recorded mouse running on the spherical treadmill for 10 minutes. Motion data was acquired at 20 Hz concomitantly with digital outputs that can be used to trigger individual image frame capture from a sCMOS camera. To measure locomotion from awake head fixed mice, we used the Teensy interface to record from two ADNS-9800 motion sensors (Figures 1A and 2A).

ADNS-9800 sensor boards are low cost, and can measure up to 8200 counts per inch, allowing for more sensitive measurement of mouse movement than a normal computer mouse. For example, a standard computer mouse, the Logitech M100 (Logitech, PN: 910-001601), measure up to 1000 counts per inch, making the ADNS-9800 sensor over 8 times more precise at its highest setting. ADNS-9800 sensors were affixed to a “spherical treadmill” setupand wired to the Teensy as demonstrated in Figure 2A.

We calculated the velocity of the mouse, which averaged 7.1 + 6.9 cm/s (mean + std) with a maximum velocity of 47.0 cm/s, in general agreement with previous studies (Dombeck, Khabbaz, Collman, Adelman, & Tank, 2007)).

To characterize the temporal precision of the teensy interface, we measured the timing of the Teensy digital output, and compared it to the theoretical 20 Hz signal. We found that digital outputs have a near-perfect linear relationship with the theoretical signal(Figure 3B). However, we noted a 28.9 µs per second positive drift, resulting in an actual frequency of 19.999 Hz instead of 20.000 Hz. To further examine whether this small timing drift depends upon the frequency of data acquisition or the timing of the digital outputs, we performed 5 minute long recording sessions without a live mouse at 20, 50, and 100 Hz, and a 0.5ms long digital pulse designed to trigger image capture from the camera. We found that the actual frequencies were 19.999, 49.999, and 99.997 Hz, respectively. These all correspond to an approximately 30 µs delay per second, suggesting that the timing drift is independent of the data acquisition rate.

Having assessed the timing of digital outputs, we next wanted to assess its variation in timing across different output samples. We calculated the root mean squared error (RMSE) of the difference between the recorded timing of each digital output sample and the theoretical timing of the 20Hz signal. The root mean squared error is 38.9 microseconds, indicating that the digital output has microsecond-level precision. Together, these results demonstrate that the Teensy interface using the IntervalTimer function can be used to generate digital pulses for precise and accurate image frame capture during behavioral experiments while maintaining alignment of imaging data with behavioral parameters.

*Trace eye blink conditioning behavioral experiment*

In this experiment, we designed the Teensy interface for a trace conditioning learning experiment (Figure 1B and 2B), where a mouse can be trained to associate a conditioned stimulus (700ms long tone and/or light) with a subsequent unconditioned stimulus (a 100ms long gentle eye puff), separated by a brief memory trace time window (250ms). In each session, 50 trials were performed with each trial lasting 20 seconds long.

We first characterize the temporal precision of the teensy interface similar to that described in the motion tracking experiment. We recorded the timings of the digital pulses generated to trigger each image frame capture (Figure 4A), and found a 33.4 microsecond delay per second. Thus, the teensy interface with the “ElapsedMicros” function has an actual frequency of 19.999 Hz instead of 20.000 Hz The RMSE of the teensy interface is 13.3 us, similar to that observed in the motion tracking experiment.

We then characterized the precision of multiple digital outputs, by calculating the time difference between the digital pulses generated to drive devices for eye puff and LED light (Figure 4Bii).We found that there is nearly no temporal difference between the onset of the digital output for the eye puff versus that for camera image frame capture (0.004 + 0.012ms (mean + std) ms, range=0.04 ms seconds). Similarly the duration of the puff digital pulse was 100.03+0.02 ms (mean + std), close to the commanded duration of 100ms .

We next characterized the analog output generated by Teensy. We measured the analog output waveforms of teensy with the commercial TDT RZ5D recording device. Since analog outputs were generated together with the onset of the digital outs used to trigger camera image frame capture, we calculated the time difference between the onset of the analog output and the onset of the digital pulse (Figure 4Bi, details see Methods).

We found that the analog output directed to the speaker lagged the digital output directed to the camera by 7.6 + 0.9 milliseconds (mean+/-STD???, Figure 4Bi). The duration of the tone was 700 + 1 ms, (mean+/-??? Figure 4Bii). Together, these results demonstrate that the Teensy interface capable of generating analog ouputs has microsecond temporal precision, and custom functions using the ElapsedMicros” function and the internal timer function? Are equally accurate.

**Conclusion and Discussion**

We demonstrate the temporal precision of a Teensy 3.2 interface in integrating a sCMOS camera during two experimental settings. In one setting, the Teensy interface simultaneously generates digital pulses that can be directed for individual frame capture from a sCMOS camera, while simultaneously recording animal’s motion information from recently developed high precision ADNS-9800 gaming sensorsThe easy integration of the sCMOS camera and the high precion ADNS-9800 sensor illustrates the flexibility of the Teensy interface in designing experiments that require novel instrumentation. In a second experiment, we demonstrate that the Teensy interface, in conjunction with a prop shield, is capable of controlling three devices with precise timing during a trace conditioning experiment, with an option of including a fourth (an LED light, not utilized here). This Teensy interface can be immediately adopted for the designed motion tracking and trace conditioning behavioral experiments, or customized for other types of behavioral experiments, where sCMOS camera-based imaging is desired.

In both experiments, the timings of camera-directed digital pulses sent by the Teensy interface were precise, as measured by the root mean square error of the model fits, to the level of microseconds, and were accurate to within approximately 30 µs per second. This small 0.003% drift of the Teensy processing clock is linear and can thus be calibrated if desired. This finding additionally underscores the necessity of having a highly precise central device for experimental control. Synchronizing different devices only with a single signal at the start of an experiment can lead to problems if these devices have different temporal drifts, particularly if the experiment is long in duration. Alternatively, initiating or measuring experimental events on a frame-by-frame basis using a PC for timing can introduce timing jitter due to the multitude of tasks that a PC must attend to at any given point in time. This jitter can have a significant impact depending on the study. 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.

Temporal accuracy is not only important for behavioral data acquisition. In our trace conditioning experiment, for example, precisely timed stimuli are desired as well. Therefore, in this experimental setting, we characterized the accuracy and precision of digital and analog outputs, which the Teensy interface had to balance with repeated digital pulses directed at a sCMOS camera. First, we show that our Teensy interface accurately and precisely delivers a 9500 Hz tone using the built-in Audio library. This Audio library can also be used for sound synthesis, reading, and mixing, all at 44.1 kHz, which is stereo quality. This provides a tool for many experimental conditions, especially those needing sound as a stimulus. Secondly, we show that our Teensy interface accurately and precisely delivers a longer digital pulse that can drive “puffs” while simultaneously producing camera-directed digital pulses. 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 simultaneously capture imaging data using our simple software design.

A major advantage of the Teensy over other microcontrollers is its ability to generate a true, 12 bit analog signal, which has been, to this point, illustrated only insofar as it can be used to produce a sound. While Arduino microcontrollers can generate an analog-like signal, they need extra devices such as resistors and capacitors in order to do so. This makes any task that requires some type of analog output easier to implement with a Teensy, which makes it highly flexible for diverse experimental designs than benefit from any type of high resolution analog output. In sum, the precision and flexibility of our Teensy 3.2 microcontroller interface makes this a user-friendly, easily adaptable, accurate, and precise tool for use with simultaneous image capture, behavioral control and data acquisition.

Other implementations of the Audio library could potentially offer even more precision and accuracy. 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 only a few lines of code within a single script.

**Figures**

**Figure 1.** Diagrams of the two complete experimental device setups using a Teensy interface. **A** A floating, spherical treadmill setup for locomotion recording. This experimental design consists of a Teensy 3.2 connected to two ADNS-9800 sensors via serial-peripheral interfaces, and a CMOS camera through a coaxial cable. Every 50 milliseconds, a digital pulse was sent to the CMOS camera to elicit an image frame capture. Simultaneously, the Teensy acquired motion data from both ADNS sensors and sent them to a PC via a USB. The PC initiates each experiment by sending serial data consisting of the length of the experiment and the digital output frequency to the Teensy. **B** A trace eye-blink conditioning setup. In order to generate a sound through the speaker, the Teensy is soldered to a prop-shield, which contains an amplifier. The length and the number of experimental trials were specified in MATLAB. The Teensy 3.2 sends time stamps, trial, and stimulus information (audio, LED, and puff pin states) via the USB back to the PC. In each trial, the Teensy generates a 9500 Hz tone at a sampling rate of 44.1 kHz. The tone stimuli, delivered optionally along with turning on an LED light (not implemented in our example experiment), is followed by an eye puff.

**Figure 2.** Detailed electrical wiring schematics for both the locomotion recording and trace eye-blink conditioning experiments **A.** A schematic demonstrating the wiring of a Teensy 3.2 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. **B** A schematic demonstrating the electrical connections between a Teensy 3.2, a prop shield, and an external speaker. Dotted lines indicate 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 constructed using 22 gauge wire and a coaxial cable. Some extraneous and unused pins on the Teensy and the prop shield were not included in this diagram.

**Figure 3.** Example recording using the locomotion recording experimental design **A** Part of a sample 10 minute recording session during which a head-fixed mouse was allowed to run on the three-dimensional treadmill shown in Figure 1A. **B** Times of digital pulses sent by the Teensy as measured by an external device, vs theoretical times of the digital pulses at exactly 20 Hz. Red indicates linear model prediction, and in black are experimental data, down-sampled by a factor of 200. The linear model estimates a slope of 1.000028927 + 0.000000005 (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 trace conditioning tone-puff setup **A** Timing of the sCMOS camera-directed digital pulses as measured by an external device vs theoretical timing assuming a frequency of precisely 20 Hz. 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 of the analog output corresponding to tone delivery (i-ii) and digital output corresponding to puff delivery (iii-iv), both measured by an external device over the course of fifty trials; (i) shows the latency between the programmed onset of the tone and the timing of the tone onset as measured by the TDT device (mean=7.6 + 0.9 ms, range=2.9 ms); (ii) shows the accuracy and precision of the length of tone intervals across all trials (mean=700 + 1 ms, range=2.9 ms); (iii) shows the latency of the puff digital pulse, as measured by the TDT system (mean= -0.004 + 0.012 ms, range=0.04 ms); (iv) shows the accuracy and precision of the length of the puff digital pulse 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