INNOVATIVE TEACHING

2013, Volume 2, Article 2
ISSN 2165-2236

DOI 10.2466/07.08.IT.2.2
© Christopher A. Varnon 2013
Attribution-NonCommercial-NoDerivs CC-BY-NC-ND

Received January 24, 2013
Accepted February 20, 2013
Published April 4, 2013

CITATION
Varnon, C. A., & Abramson, C. I. (2013) The propeller experiment controller: low-cost automation for classroom experiments in learning and behavior. *Innovative Teaching*, 2, 2.

The Propeller Experiment Controller: low-cost automation for classroom experiments in learning and behavior[1]

Christopher A. Varnon and Charles I. Abramson

Laboratory of Comparative Psychology and Behavioral Biology
Departments of Psychology and Zoology
Oklahoma State University

Summary

The use of the Propeller Experiment Controller to create inexpensive teaching laboratories in behavior is described. Pre-written programs are provided for unique classroom experiments in habituation and classical conditioning. Several programs are also provided to recreate the traditional student operant conditioning laboratory. Video tutorials are also available to guide the user to effectively use the Propeller Experiment Controller for behavioral experiments.

This article offers readers interested in the experimental analysis of behavior an inexpensive, versatile, and powerful experimental controller suitable for teaching laboratories. Our design is based on the Parallax Propeller microcontroller (Parallax Inc.; Rocklin, California). In contrast to experiment controllers available from companies such as Lafayette Instruments and Med Associates that cost thousands of dollars and are cumbersome, the Propeller costs fewer than one hundred dollars, is easily portable, powerful, and readily adaptable to a wide range of situations. A laboratory can literally be placed in the palm of your hand and carried from office to classroom in your pocket. See Fig. 1 for an image of the Propeller Experiment Controller.

For over two decades our laboratory has been fighting increasing use of computer simulations such as Cyber Rat ((AI)^2; Winter Park, Florida) and Sniffy the Rat (Graham, Alloway, & Krames, 1994) as a replacement for hands-on, inquiry-based learning experiences in behavior (Abramson, 1986, 1990; Abramson, Curb, Barber, & Sokolowski, 2011). Such products do not allow students to experience the nuances associated with the study of behavior and reduces the student to playing what amounts to a video game. In a study specifically designed to compare the experiences of working with live animals and computer stimulations, students found the live animal demonstration to be more realistic and more valuable as a learning experience than a computer simulation of classical conditioning (Abramson, Onstott, Edwards, & Bowe, 1996). The authors find it interesting that no one seriously questions the silliness of using a video game to train clinical psychologists, yet video games are routinely employed to replace animals in the teaching process.

We also believe that the lack of meaningful student experience with animals contributes to the decline of comparative psychology as a viable psychological enterprise. Cyber Rat and Sniffy the Rat are not suitable for a comparative psychology course because they only focus on emulating the behavior of a single species. What is needed to generate students' interest in comparative psychology are exercises that are truly comparative and experiential (Abramson, Hilker, Becker, Barber, & Miskovsky, 2011).

In this paper, we provide a brief introduction to programming and connecting the Propeller to various response devices such as a levers and photocells. We also describe how the Propeller can be used to activate lights and reinforcement dispensers. Our software that enables the Propeller to be used as an experiment controller is included, as

[1]Preparation of this Manuscript was supported in part by Sigma Xi Grants-in-Aid of Research G20120315160075, a Society for the Advancement of Behavior Analysis International Development Grant, a Provost Research Teaching Grant from Oklahoma State University, and National Science Foundation grants DBI 0552717 and OISE 1043057.

Address correspondence to Charles I. Abramson, Laboratory of Comparative Psychology and Behavioral Biology, Departments of Psychology and Zoology, Oklahoma State University, 116 N. Murray, Stillwater, OK 74078 or e-mail (charles.abramson@okstate.edu).

AmSci
Ammons Scientific
www.AmmonsScientific.com

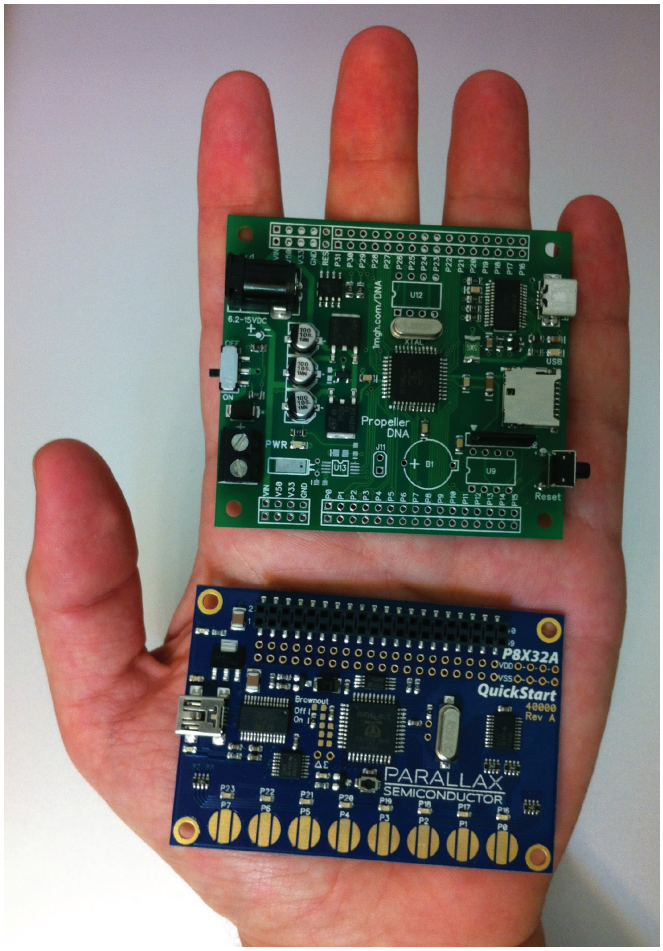

Fig 1 Two experiment controllers in one hand. Top: Propeller Platform DNA. Bottom: Parallax QuickStart.

well as pre-written programs for experiments in habituation, classical conditioning, and operant conditioning. The software package can be found by clicking here (http://www.amsciepub.com/doi/suppl/10.2466/07.08.IT.2.2). We discuss the use of these programs in a classroom setting. Video tutorials are also available to guide the user to effectively use the Propeller for behavior experiments. These tutorials can be found by clicking here (http://www.amsciepub.com/doi/suppl/10.2466/07.08.IT.2.2). To fully appreciate this paper some basic electrical skills are necessary. Experience with a breadboard, soldering iron, and voltmeter will be beneficial. Programming experience is not required to use the experiment programs we provide. However, some basic programming skills will allow the user to create novel experiments that go far beyond those described in this paper.

Overview of the Parallax Propeller

The Parallax Propeller microcontroller has 32 input/output (I/O) pins. Each I/O pin can be connected to some other device, either an input or an output. Input devices are typically used to record the activity of a subject and commands from the user and may include levers, photocells, and user interfaces. These devices are not limited to a simple digital on-off input; the I/O pins are also able to interface with a wide variety of devices that enhance the Propeller's functions. In addition to detecting simple events such as the number of times a lever was pressed, we have also connected the Propeller to other inexpensive devices to record temperature in an apparatus, use echolocation to detect the location of a subject in a runway, and to detect the faint glow from bioluminescent organisms.

Output devices are typically used to operate stimuli such as lights and food hoppers. The Propeller can also create more complex forms of output than the simple turning on and off of a device. The Propeller is powerful enough to generate audio and video signals. Audio generation is useful in any experiments related to auditory discrimination or conditioning. No separate tone generators are required; simply connect the Propeller to a speaker. Generating video is useful for tasks using visual stimuli without requiring the user to install an array of lights. A TV monitor or LCD screen can also be connected to the Propeller to allow the user to see data in real time. User interfaces can also be created through a combination of devices or through touch screens to provide control over the variables in an experiment without additional programming. These types of interfaces are great for student devices and may be used to allow the student to select specific values of experimental variables such as the inter-trial interval, and reinforcement schedule.

The Propeller is also very easy to use. We have experience with several programming languages, almost always related to experiments or data analysis, and have found the Propeller's language, "Spin", to be one of the easiest. Another feature that makes the Propeller easy to use is its unique multi-core design. Each of the Propeller's 8 processors, called cogs, can complete tasks independently of the others. This makes programming complex tasks easy, and even makes it possible for the Propeller to run several unrelated, independent experiments at the same time.

Complex tasks are made simple through the use of prewritten "objects." The Propeller is very well supported by a variety of users in hobby, industry, and robotics. Objects to deal with a variety of tasks are written and often made freely available. The objects function to perform a variety of tasks from writing to SD cards, to using infrared sensors to detect distance to an object, to generating video and detecting input on a touch screen. There is an object for everything. This allows users to write programs faster as they do not need write every function themselves. It also allows users to perform complicated functions that may be beyond their current level of programming skill.

The Propeller as an Experiment Controller

The previously mentioned characteristics make the Propeller an excellent experiment controller. We have used the Propeller in our laboratory to conduct investigations in the learning of bees, bioluminescent algae, pigeons, planarians, and rattlesnakes. To make experiments easy, we developed a set of objects dedicated to conducting experiments and collecting data. These objects provide a framework for future experiments. Our experience with the Propeller in the laboratory suggests that it would be an excellent device for classroom use. To further encourage classroom use we also developed 20 programs for classroom experiments.

Using the Propeller for Classroom Experiments

In this section, we describe turn-key programs for several laboratories in habituation, classical, and operant conditioning. These laboratories are suitable for a variety of courses including those in introductory psychology, psychology of learning, and comparative psychology. Moreover, the laboratories can be used by students in conducting their own independent investigations as part of, for example, a directed reading course.

The habituation and classical conditioning laboratories are unique. There are currently no automated classroom demonstrations for these two classes of conditioning. Our habituation laboratory allows the student to vary such training variables as the number of habituation trials and time between trials. The classical conditioning laboratory allows the student to explore not only the effect of training variables but also various arrangements of conditioned and unconditioned stimuli.

We have also included a series of operant conditioning exercises. These exercises repeat the classic demonstrations so familiar to a previous generation of students. In this laboratory, students learned to shape their animal's behavior and conduct simple demonstrations in acquisition, extinction, discrimination, chaining, and schedule effects.

For those readers unfamiliar with using microcontrollers or connecting input and output devices we suggest that you read the sections on preparing the Propeller for experiments and attaching devices in Appendix A. We also created several video tutorials that may be helpful.

Habituation

In this section, we will describe how to use the Propeller to create a student demonstration in one of the more simple types of learning, habituation. Habituation refers to the reduction of a response to some stimulus as that stimulus is repeatedly presented. Although habituation is a simple form of learning compared to classical and operant conditioning, it shares many characteristics with these higher-order forms of learning including discrimination, generalization, and spontaneous recovery. Therefore, habituation experiments are an easy and practical way to demonstrate complex learning characteristics. In the following paragraphs, we will describe how to present stimuli, select training variables such as stimulus duration and inter-trial interval, and record responses using habituation programs we created. The instructions in this section are more detailed than instructions related to classical and operant conditioning experiments and serve to familiarize the reader with the Propeller and the programming paradigm we use throughout all the provided experiments.

First, open the habituation program in Propeller Tool or BST. You will notice the large document is broken into color-coded blocks of text. Much of this text is code to run a variety of habituation experiments. Additional text found within brackets or following quotation marks contains descriptions of how the program works. The reader is free to read the code along with the comments to gain a better understanding of how to create other programs for classroom experiments. The general function of the program is to repeatedly present a stimulus, while recording responses from the subject. The program also allows the user to provide specific values of training variables.

Before conducting habituation experiments, a little information must be provided to the program. Focus on the CON block starting at line 21. This section contains constants that are used by the program to detect responses, activate stimuli, and implement specific training procedures. First, you will need to provide some information about the SD card that the Propeller Experiment Controller uses to save data. See Appendix A for instructions on installing an SD card. Provide the pin numbers of the connections to the SD card on lines 33–36. Each connection is named following SD card conventions. If you use the Propeller Platform DNA, or a QuickStart with a Human Interface Board, the pin connections will be as follows: DO = 0, CLK = 1, DI = 2, CS = 3.

Next, provide the pin numbers connected to the apparatus on lines 50–54. The response pin refers to the pin connected to the device that detects a subject's response. The stimulus pin refers to the pin connected to the stimulus device, such as a vibrating motor or a light. The house light pin is optional. It refers to a light traditionally built into an apparatus that turns on when the session starts and turns off when the session ends. The diagnostic LED pin refers to an optional connection to a diagnostic LED. In all our programs, this LED will start flashing after the experiment is complete to signal that it is safe to remove the SD card. Although this device is optional, we highly suggest using it so that the SD card is not removed prematurely. Finally, the escape pin refers to an optional connection to a button that allows the user to end the experiment if the subject does not habituate. If one of the optional devices is not desired, set the pin number to a pin that is not connected to another device.

By default, the program will name the stimulus presentations "Stimulus" and any activation of the response device "Response" in the data file. The user is free to change these names in the "SetVariables" method on lines 156 and 157. Simply change the word "Response" or "Stimulus" into whatever description is desired. Note that the quotation marks are important. Only change the text within the quotation marks.

Training variables for the habituation program can be set in the CON block on lines 41–44. As learning occurs as a function of experience, the subject will need multiple habituation trials. Too few trials will not allow for a demonstration of habituation. We suggest using at least 15 trials. It is possible that some individuals may not habituate within this number of trials. For this reason, we included an optional mastery criterion. The mastery criterion is the number of trials without any response that must occur before the experiment ends. This allows an experiment to continue until a subject demonstrates habituation. For example, if you set the mastery criterion to 5, then 5 consecutive trials must occur without the subject responding to the stimulus before the experiment ends. However, if you set the mastery criterion to 0, the experiment will end as soon as the initial trials are complete, without regard to the subject's responses. See Place and Abramson (2008) for an example of using a mastery criterion in habituation of the rattle response in rattlesnakes.

Set the stimulus duration on line 43. Note that the duration is listed in milliseconds, as this is the base unit of time used by all our programs. Multiply minutes by 60,000 to quickly translate minutes into milliseconds. Longer stimulus durations may produce faster habituation than shorter stimulus durations. If the stimulus is long enough, you may also observe that the subject begins responding at different times during the stimulus. Initially, the subject may respond as soon as the stimulus starts. As trials pass, it may begin to respond closer to the middle or the end of the stimulus before habituating. A quick analysis of the data.csv file may make these changes easier to notice than by direct observation alone.

Another important training variable is the time between stimulus presentations or inter-trial interval. Inter-trial interval can be set on line 44. Generally, the shorter the inter-trial interval, the faster habituation occurs (Thompson & Spencer, 1966). One interesting classroom experiment is to use short inter-trial intervals with one group of subjects, and long inter-trial intervals with another group of subjects. This type of simple demonstration allows students to prove for themselves that training variables, in this case inter-trial interval, greatly affect learning.

Variable stimulus durations and inter-trial intervals may produce different rates of habituation than fixed stimulus durations and inter-trial intervals. Variability is likely to cause slower habituation. However, you may notice that habituation generalizes better to novel stimulus durations and inter-trial intervals after receiving training using variable durations. To explore these factors, we created a second habituation program that will allow the stimulus duration and the inter-trial interval to be randomly selected from a user-specified range. Open the variable habituation program. Note that it is very similar to the habitation program, except that you will need to provide the minimum and maximum duration of the stimulus presentations and inter-trial interval. If the minimum and maximum duration are identical, no variability will occur. Also note that although the general format is the same, some line numbers of constants changed slightly. From the user's perspective, the programs are otherwise identical.

Intensity of a stimulus is also an important factor in habituation. Animals habituate to weaker stimuli faster than stronger stimuli (Thompson & Spencer, 1966). Additionally, an individual that has habituated to a weak stimulus, such as a low volume noise, may still react to a stronger stimulus, such as a loud noise. A third habituation program titled "Variable Intensity Habituation" allows the user to specify a variable level of stimulus intensity in addition to variable levels of stimulus duration and inter-trial interval. As with other variable factors, the variable levels of stimulus intensity are generated randomly from user provided minimums and maximums. Additionally, an optional second stimulus intensity level can be used. On lines 54 and 55, you will notice familiar constants regarding the number of trials and mastery criterion for habituation. These refer only to the first stimulus intensity level. After the number of trials has passed and any mastery criterion is met, the program will begin presenting the stimulus at a second intensity level. On lines 57 and 58, you can set the number of trials and mastery criterion for this second intensity level, or set both to 0 if you do not wish to use a second intensity level. Both levels of stimulus intensity are set on lines 65–68. The second level of intensity enables experiments such as habituating a response to a low intensity, then increasing the intensity to observe if the response remains habituated. For example, if an earthworm habituates to a weak vibration from a motor, will a strong vibration elicit a response?

One important note about this program is that it will not work with devices controlled by relays. The program uses a method called pulse-width modulation (PWM) to create varying levels of stimulus intensity. This involves rapidly turning a device on and off thousands of times per second, creating an illusion of changes in intensity such as dimmer light levels or slower motors. Relays cannot operate this quickly, so this program is only suitable for devices connected directly to the Propeller's

I/O pins or to devices connected through transistors. For relay-controlled devices we recommend the use of potentiometers placed between the device and its power source. This will allow students to control the intensity of the device manually.

A variety of habituation experiments with multiple species can be conducted using these programs. Planarians are inexpensive to procure and maintain and can be used to demonstrate a variety of learning principles. A brief air puff from a relay-controlled aquarium pump can elicit body contractions and extensions. Air puffs and vibrating motors can also be used to elicit the audible hissing response of Madagascar hissing cockroaches, body contractions in earthworms (Abramson, 1990), eye stalk retractions in crabs (Abramson & Feinman, 1988), and head-withdrawal responses in turtles. Acoustic stimuli can produce startle responses in a variety of species but may not be suitable to classroom demonstrations if multiple experiments are being conducted in a small space. Although these responses are easily observable, some are difficult to detect electronically. We suggest requiring students to press a pushbutton each time the subject responds. This form of manual recording maintains students' attention while still providing more detailed and accurate data than paper-and-pencil reports.

Simple habituation experiments with students can also be conducted. For example, a strong vibration from a vibrating motor held in the hand may cause a noticeable reaction in a student. A student can press a pushbutton to record a response each time they are startled. Students can also work in pairs, with one student acting as the subject and another student acting as the researcher. Have the researcher student press a pushbutton when they observe a response. The students can then alternate. This exercise is also useful because the students must carefully consider how they operationally define a response to the vibration so that they can accurately record responses.

When the experiments are completed, the program will generate a spreadsheet file titled "data.csv" that can be opened and analyzed in programs such as Microsoft Excel. Graphs can also be constructed for visual analysis of the data. See Fig. 2 for an example of the data file. All of our programs create this kind of data file. It is important to note that any file named "data.csv" will be overwritten when new data is saved. Be sure to remove the SD card, save the data to the computer, and then change the file name to something more descriptive before starting a new experiment. Advanced users may take advantage of some methods provided in Experimental Functions to save spreadsheets with custom names.

A maximum of 1,000 instances of each type of event can be saved to the spreadsheet. If more than 1,000 instances of an event occurred during the session, an error message will be saved to the SD card. To create a spreadsheet, you can use the Python program included in the Propeller Experiment Controller package. See Appendix C for a brief description of the Propeller Python Interface program.

Classical Conditioning

Classical conditioning is a more complex form of learning in which an animal learns an association between a conditioned stimulus (CS) and an unconditioned stimulus (US). The CS is an originally neutral stimulus that comes to elicit a response only after it is repeatedly associated with a US that already produces that response. In this section, we will describe how to use the Propeller to create classroom demonstrations and experiments in classical conditioning. For instructions regarding setting up the Propeller and related software, please refer to the instructions in the habituation section and Appendix A.

Open the classical conditioning program in Propeller Tool or BST. As with other programs you will need to provide pin numbers for the SD card, stimuli, and response devices. The general function of the program is to present the CS and US during a series of trials. See Fig. 3 for an example data spread sheet created by the classical conditioning program. Set the trial length on line 50. On lines 51–54, you can set the time within the trial that the CS and US will start and stop. For example, if you set a trial length of 10 seconds, a CS that starts at 0 and ends at 10 will occur during the entire trial. A more likely interval would be for the CS to start at 0 and end at 5, while the US starts at 2 and ends at 5. This would cause the CS to start as soon as the trial begins, the US to start 2 seconds later, and both stimuli to end 5 seconds into the trial. After the final 5 seconds of the trial elapses a new trial will start. See Fig. 4 for an illustration of these constants.

As you can specify the start and stop times of both stimuli within a trial, a wide variety of conditioning procedures can be created. Many classical conditioning procedures are considered forward conditioning procedures in which the CS precedes the US. Forward condition procedures are generally effective because the CS predicts US. One type of forward conditioning is delay conditioning. In delay conditioning, the CS starts, then the US starts after a small delay, and then both stimuli end simultaneously. To create a delay conditioning procedure make sure that CS_Start is less than US_Start and that CS_Stop is equal to US_Stop. For example, with a trial length of 10 seconds, set the CS_Start to 0 seconds, the US_Start to 2 seconds, and both the CS_Stop and US_Stop to 4 seconds.

Another form of forward conditioning is trace conditioning. In trace conditioning the CS starts first, then the US starts after the CS ends. The time between the

<>	A	B	C	D	E	F	G	H
1	Event	Instance	Onset	Offset	Duration	Inter-Event Interval	Total Duration	Total Occurrences
2	Stimulus	1	0	0.751	0.751	0	14.269	19
3	Response	1	0.334	0.464	0.13	0	1.49	10
4	Stimulus	2	5.751	6.502	0.751	5	14.269	19
5	Response	2	6.015	6.131	0.116	5.551	1.49	10
6	Stimulus	3	11.501	12.252	0.751	4.999	14.269	19
7	Response	3	11.772	11.903	0.131	5.641	1.49	10
8	Stimulus	4	17.251	18.002	0.751	4.999	14.269	19
9	Response	4	17.518	17.638	0.12	5.615	1.49	10
10	Stimulus	5	23.001	23.752	0.751	4.999	14.269	19
11	Response	5	23.299	23.439	0.14	5.661	1.49	10
12	Stimulus	6	28.751	29.502	0.751	4.999	14.269	19
13	Response	6	29.247	29.38	0.133	5.808	1.49	10
14	Stimulus	7	34.501	35.252	0.751	4.999	14.269	19
15	Response	7	34.796	34.959	0.163	5.416	1.49	10
16	Stimulus	8	40.251	41.002	0.751	4.999	14.269	19
17	Response	8	40.714	40.879	0.165	5.755	1.49	10
18	Stimulus	9	46.001	46.752	0.751	4.999	14.269	19
19	Stimulus	10	51.751	52.502	0.751	4.999	14.269	19
20	Response	9	52.436	52.658	0.222	11.557	1.49	10
21	Stimulus	11	57.501	58.252	0.751	4.999	14.269	19
22	Stimulus	12	63.251	64.002	0.751	4.999	14.269	19
23	Stimulus	13	69.001	69.752	0.751	4.999	14.269	19
24	Stimulus	14	74.751	75.502	0.751	4.999	14.269	19
25	Response	10	75.492	75.662	0.17	22.834	1.49	10
26	Stimulus	15	80.501	81.252	0.751	4.999	14.269	19
27	Stimulus	16	86.251	87.002	0.751	4.999	14.269	19
28	Stimulus	17	92.001	92.752	0.751	4.999	14.269	19
29	Stimulus	18	97.751	98.502	0.751	4.999	14.269	19
30	Stimulus	19	103.501	104.252	0.751	4.999	14.269	19

Fig 2 An example data file automatically generated by the habituation program. The instance column refers to the number of the event, such response instance 1, the first response. The onset and offset columns refer to the start and stop times of an event, respectively. Duration refers to the duration of that instance of an event, while total duration refers to the combined duration of all instances of an event. Inter-event interval refers to the time between events of the same type. Finally, total occurrences refers to the total number of times an event occurred in a session. When the file is produced, it is initially sorted by event type. In most spreadsheet programs, the data can be sorted by row. In this example, the data has been sorted by onset, so that the events appear in chronological order. The event, instance, and onset rows indicate that fewer responses were elicited by the stimulus as the experiment progressed. Also note that the inter-event interval of the response, or the time between responses, increases from approximately five seconds to eleven, then twenty-two seconds.

end of the CS and the start of the US is known as the trace interval. To create a trace conditioning procedure make sure that US_Start is greater than CS_Stop. The difference between these two constants is the trace interval. For example, with a trial length of 20 seconds, set the CS_Start to 0 seconds and the CS_Stop to 2 seconds. Then set the US_Start to 4 seconds and the US_Stop to 6 seconds. In this case, the trace interval is 2 seconds. Generally, the longer the trace interval, the more experience is required before the CS elicits a response. Other conditioning procedures such as simultaneous conditioning and backward conditioning can also be conducted by adjusting the start and stop times of the stimuli.

You will also notice several trial types listed on lines 56–58. Acquisition trials refer to the pairing of CS and US as in the previously mentioned procedures. During acquisition, the animal is acquiring or learning the association. The primary purpose of the classical conditioning program is to present the CS and US at the user-specified times during a user-specified number of acquisition trials. You can also use an optional number of inhibition trials, where the CS is presented by itself before the acquisition trials begin. As the CS is repeatedly demonstrated not to predict anything in inhibition trials, acquisition can take longer once the CS and US are paired. After the acquisition trials end, you can use an optional number of extinction trials where only the CS is presented. In extinction trials the association between the CS and US previously learned during acquisition becomes extinguished because the CS is no longer associated with the US.

Another method to study classical conditioning is discrimination training. An animal will learn to

<>	A	B	C	D	E	F	G	H
1	Event	Instance	Onset	Offset	Duration	Inter-Event Interval	Total Duration	Total Occurrences
2	CS	1	1	3	2	0	40.008	15
3	US	1	2	3.001	1.001	0	20.012	15
4	Response	1	2.385	2.559	0.174	0	3.194	15
5	CS	2	11	13.001	2.001	8	40.008	15
6	US	2	12	13	1	8.999	20.012	15
7	Response	2	12.682	12.865	0.183	10.123	3.194	15
8	CS	3	21	23.001	2.001	7.999	40.008	15
9	US	3	22	23	1	9	20.012	15
10	Response	3	22.574	22.748	0.174	9.709	3.194	15
11	CS	4	31	33	2	7.999	40.008	15
12	US	4	32	33.001	1.001	9	20.012	15
13	Response	4	32.767	32.934	0.167	10.019	3.194	15
14	CS	5	41	43	2	8	40.008	15
15	US	5	42	43.001	1.001	8.999	20.012	15
16	Response	5	42.333	42.515	0.182	9.399	3.194	15
17	CS	6	51	53	2	8	40.008	15
18	US	6	52	53.001	1.001	8.999	20.012	15
19	Response	6	52.28	52.421	0.141	9.765	3.194	15
20	CS	7	61	63	2	8	40.008	15
21	US	7	62	63.001	1.001	8.999	20.012	15
22	Response	7	62.194	62.357	0.163	9.773	3.194	15
23	CS	8	71	73	2	8	40.008	15
24	Response	8	71.933	72.086	0.153	9.576	3.194	15
25	US	8	72	73.001	1.001	8.999	20.012	15
26	CS	9	81	83	2	8	40.008	15
27	US	9	82	83.001	1.001	8.999	20.012	15
28	Response	9	82.174	82.314	0.14	10.088	3.194	15
29	CS	10	91	93	2	8	40.008	15
30	US	10	92	93.001	1.001	8.999	20.012	15
31	Response	10	92.068	92.209	0.141	9.754	3.194	15
32	CS	11	101	103.001	2.001	8	40.008	15
33	Response	11	101.816	101.957	0.141	9.607	3.194	15
34	US	11	102	103	1	8.999	20.012	15
35	CS	12	111	113.001	2.001	7.999	40.008	15
36	US	12	112	113	1	9	20.012	15
37	Response	12	112.263	112.391	0.128	10.306	3.194	15
38	CS	13	121	123	2	7.999	40.008	15
39	Response	13	121.614	121.758	0.144	9.223	3.194	15
40	US	13	122	123.001	1.001	9	20.012	15
41	CS	14	131	133.001	2.001	8	40.008	15
42	Response	14	131.519	131.683	0.164	9.761	3.194	15
43	US	14	132	133	1	8.999	20.012	15
44	CS	15	141	143	2	7.999	40.008	15
45	Response	15	141.075	142.282	9.392	10.392	3.194	15
46	US	15	142	143.001	1.001	9	20.012	15

Fig 3 An example data file automatically generated by the classical conditioning program. In this example, the data has been sorted by onset, so that the events appear in chronological order. Note that the response occurred after the US for the first 7 trials, and then began occurring after the CS but before the US on the remaining trials.

respond to a stimulus that is associated with the US, the CS+, and will learn not to respond to a stimulus that is not associated with the US, the CS-. A separate program titled "ClassicalConditioningDiscrimination" can conduct these types of experiments. The format of the program is similar to the previous classical conditioning program; you will need to set many of the same constants. This program is distinct in that there are no inhibition, acquisition, or extinction trials. Instead, the number of trials provided on line 58 reflects the number of CS+ and CS- trials. The total number of trials is double this number. At the start of each trial, the program randomly determines if the trial will be a CS+ trial, in which the CS+ and US are presented, or a CS- trial, in

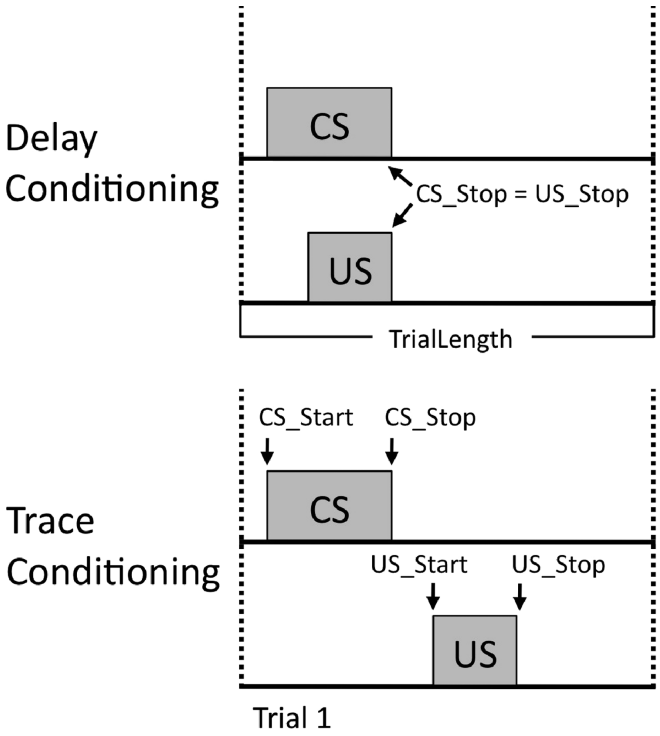

Fig 4 A diagram illustrating the constants in the classical conditioning program. The user provides a trial length, and the start and stop times of the CS and US. In the delay conditioning example, the CS_Start time occurs early within the trial and the US_Start occurs at a slight delay. Both stimuli end simultaneously as the CS_Stop and US_Stop times are identical. In the trace conditioning example, the CS_Start and CS_Stop times are the same as the previous example. Although the US has the same duration as the delay conditioning US, the US_Start and US_Stop times occur much later in the trial. The time between CS_Stop and US_Start can be considered the trace interval. Adjusting the start and stop times of both stimuli can create many other conditioning procedures.

which just the CS- is presented. Alternatively, the program can use a pseudorandom selection so that the same trial type does not randomly occur many times consecutively. The number entered on line 59 for ConsecutiveLimit refers to the maximum number of trials of the same type that can occur simultaneously. Increase ConsecutiveLimit so that it exceeds the number of trials for completely random selection.

With these programs, many classical conditioning experiments can be conducted. Often an apparatus used to study habituation can also investigate classical conditioning. For example, the body-contraction response of the earthworm can be explored in either a habituation or a classical conditioning procedure. In the habituation procedure, vibration is repeatedly presented. In the classical conditioning procedure, some other stimulus, such as a light, acts as a CS that predicts the vibration (Abramson, 1990). The eye retraction of crabs in response to a puff of air can also be studied in either a habituation or a classical conditioning procedure. For crabs, light vibrations can act as a CS (Abramson & Feinman, 1988). For these two experiments, aquarium air pumps and vibrating motors can easily be controlled by relays. The responses, however, are more difficult to detect automatically and we again suggest that students activate a pushbutton to record responses.

Honeybees are also exceptionally good subjects for classical conditioning procedures as they learn relatively quickly and can be collected in large numbers. One common area of study in honeybee learning is the proboscis extension reflex. When a bee's antenna contacts sugar water, the proboscis automatically extends to drink the solution. This reflex can then be conditioned to occur to a wide variety of odor CSs. The procedure is simple enough to be conducted manually (Abramson, 1990) but can also be fully automated (Abramson & Boyd, 2011).

Classical conditioning experiments can also be conducted with students. For example, the same vibrating motor used in student habituation experiments can also be used as a US in classical conditioning experiments. An LED can then act as a CS. As before, students can work individually or in pairs and press a pushbutton to record responses. With the correct classical conditioning procedure, the students may begin to startle when the LED turns on before the motor activates.

Operant Conditioning

One of the most popular teaching demonstrations widely used by the previous generation of psychology students is the standard operant conditioning pigeon or rat laboratory. In this laboratory, students learned to shape an animal's behavior and conduct simple demonstrations in acquisition, extinction, discrimination, chaining, and schedule effects. In this section, we will describe how to use the Propeller to recreate traditional operant demonstrations and experiments. For instructions regarding setting up the Propeller and related software, please refer to the instructions in the habituation section and Appendix A.

The first demonstration in many operant conditioning laboratories is shaping. Often the target behavior shaped by the students can then be used as a foundation for further explorations in learning. Open the shaping program in Propeller Tool or BST. The general function of the program is to provide a reinforcer any time the subject activates the response device, or anytime the student activates another device to reinforce an approximation to the target behavior. Although both actual responses and approximations can produce reinforcement, the data spreadsheet makes a distinction to allow comparisons between the two types of responses and reinforcement. As with other programs you will first need to provide pin numbers for the SD card, stimuli,

and response devices. The response pin refers to the pin connected to the response device activated by the subject, such as a lever for a rat. The shaping pin refers to the pin connected to a button or lever the student will activate to manually provide reinforcement for approximations.

Our operant experiments follow a free operant paradigm, where the subject is free to earn as many reinforcers as possible within a fixed session duration. Set the session length, in milliseconds, on line 42. When reinforcement is provided, our programs activate the reinforcement device for a fixed period of time. During this period of time, the subject can respond more, but cannot earn any other reinforcers. Set the reinforcement length on line 43. This duration can be adjusted to emulate traditional pigeon feeders, where food is presented for several seconds. Alternatively, you may need a brief reinforcement duration. Some reinforcement devices such as pellet dispensers can deliver a small amount of pellets when they receive only a few milliseconds of current. For this type of device, you can set the reinforcement length to be only a few milliseconds long.

The continuous reinforcement program functions identically to the shaping program except that there is no input to manually reinforce approximations. Once the student has established a response, you can switch from the shaping program to the continuous reinforcement program. After several sessions of continuous reinforcement, stable patterns in responding will likely emerge. Then you can switch to a program with a different contingency to demonstrate the effect of contingency on behavior.

The operant discrimination program provides reinforcement for a response only in the presence of an S^D (discriminative stimulus), but never in the presence of an S^Δ (S-delta). The program randomly alternates between S^D and S^Δ presentations. You can specify pins for the S^D and S^Δ devices on lines 62 and 63. If you want to use an absence of a stimulus as an S^D or S^Δ, provide the number of the pin that is not connected to another device. For example, if you want key pecks to be reinforced only in the presence of a red light (S^D), and never to be reinforced when no lights are on (S^Δ), connect the red light to the Propeller and provide the pin number to the program. Then provide an unused pin number for the S^Δ. The S^D and S^Δ can have distinct durations. Set these values on lines 47-48. An optional time out interval can be set on line 49. During this period neither the S^D nor the S^Δ will be presented. As with the previously described classical conditioning discrimination, the S^D and S^Δ are presented randomly. However, you can set a consecutive limit to use a pseudorandom selection on line 52. An alternate version of the operant discrimination program that functions identically with slightly different code structure is also provided for readers interested in learning to create experiment programs.

We created a chaining program that will provide reinforcement only if a series of responses occurs in a specific order. For this program, a chain of either two or three responses can be used. On line 42, set UseThree-Responses to true to enable a three-response chain. Set it to false to use a two-response chain. On lines 47–49, you can set the pins for all three-response devices. The order the devices are listed here reflects the order the devices must be activated to produce reinforcement. Even if you set the program to the two-response chain mode, it will still monitor the third response device. Set the third device pin number to an unused pin if you do not have three response devices. From the user's perspective, the program is otherwise set up as any other operant program.

Exploring schedules of reinforcement is another common exercise in student operant laboratories. In the fixed ratio schedule (FR) of reinforcement programs the subject must respond a fixed number of times before receiving reinforcement. Set the FR requirement on line 42. A variety of exercises can be created just from changes in FR schedule. For example, transitioning from continuous reinforcement to a low FR schedule will result in an increase in response rate. However, if the FR schedule is initially too high, the animal may stop responding. FR schedules can also be gradually increased to very high levels. See Fig. 5 for an example data sheet created by the fixed ratio program.

We also created a variable schedule (VR) of reinforcement program. The VR program is identical to the FR program except that the response requirement is chosen randomly after each response from a user-provided range. On lines 43 and 44, you can set the minimum response requirement, VRmin, and the maximum response requirement, VRmax. All other factors can be set as they were in the FR program. Typically VR schedules produce a higher rate of response with fewer pauses than FR schedules with similar response requirements.

Fixed interval (FI) schedules of reinforcement can be used to explore an animal's ability to sense time. In this program, reinforcement is only provided when a response occurs after a fixed time interval passes. Responses before this interval elapses have no effect. Set the FI interval on line 41. Typically, as an animal learns the time interval, responses will occur more at the end of the interval than the beginning or middle. This trend can be revealed by creating scatter plots of response onset on the x, versus response instance on the y. This type of plot is very similar to B. F. Skinner's cumulative record.

A variable interval (VI) program is also available. Like the VR program the VI program functions nearly identically to its fixed counter part. The user only needs to add a minimum and maximum interval on lines 41 and 42.

<>	A	B	C	D	E	F	G	H
1	Event	Instance	Onset	Offset	Duration	Inter-Event Interval	Total Duration	Total Occurrences
2	Response	1	1.709	1.883	0.174	0	4.345	27
3	Response	2	3.815	4.024	0.209	1.932	4.345	27
4	Response	3	5.693	5.867	0.174	1.669	4.345	27
5	Reinforcement	1	5.694	7.695	2.001	0	18.009	9
6	Response	4	11.095	11.225	0.13	5.228	4.345	27
7	Response	5	12.687	12.874	0.187	1.462	4.345	27
8	Response	6	13.102	13.267	0.165	0.228	4.345	27
9	Reinforcement	2	13.103	15.104	2.001	5.408	18.009	9
10	Response	7	19.122	19.293	0.171	5.855	4.345	27
11	Response	8	19.64	19.804	0.164	0.347	4.345	27
12	Response	9	19.946	20.096	0.15	0.142	4.345	27
13	Reinforcement	3	19.947	21.948	2.001	4.843	18.009	9
14	Response	10	25.248	25.411	0.163	5.152	4.345	27
15	Response	11	25.498	25.643	0.145	0.087	4.345	27
16	Response	12	25.773	25.922	0.149	0.13	4.345	27
17	Reinforcement	4	25.774	27.775	2.001	3.826	18.009	9
18	Response	13	29.206	29.361	0.155	3.284	4.345	27
19	Response	14	30.073	30.218	0.145	0.712	4.345	27
20	Response	15	30.329	30.49	0.161	0.111	4.345	27
21	Reinforcement	5	30.33	32.331	2.001	2.555	18.009	9
22	Response	16	34.934	35.109	0.175	4.444	4.345	27
23	Response	17	35.22	35.379	0.159	0.111	4.345	27
24	Response	18	35.56	35.73	0.17	0.181	4.345	27
25	Reinforcement	6	35.561	37.562	2.001	3.23	18.009	9
26	Response	19	40.047	40.22	0.173	4.317	4.345	27
27	Response	20	40.504	40.657	0.153	0.284	4.345	27
28	Response	21	40.719	40.863	0.144	0.062	4.345	27
29	Reinforcement	7	40.72	42.721	2.001	3.158	18.009	9
30	Response	22	48.194	48.339	0.145	7.331	4.345	27
31	Response	23	48.515	48.663	0.148	0.176	4.345	27
32	Response	24	48.82	48.988	0.168	0.157	4.345	27
33	Reinforcement	8	48.821	50.822	2.001	6.1	18.009	9
34	Response	25	55.517	55.677	0.16	6.529	4.345	27
35	Response	26	55.784	55.936	0.152	0.107	4.345	27
36	Response	27	56.188	56.344	0.156	0.252	4.345	27
37	Reinforcement	9	56.189	58.19	2.001	5.367	18.009	9

Fig 5 An example data file automatically generated by the fixed ratio program. In this example, the data has been sorted by onset, so that the events appear in chronological order. Note that three responses must occur before reinforcement is produced, indicating a FR3 schedule. For traditional operant conditioning experiments, response rate can be derived by dividing the total occurrences of the response by the session length.

Fixed time (FT) and variable time (VT) also deal with time intervals. However, they are distinct from FI and VI schedules in that the subject is not required to make a response. In the FT schedule, a reinforcer is always provided after a fixed interval of time. This can be specified on line 40 of the FT program. The VT schedule provides reinforcement at variable intervals, specified by a minimum and maximum interval on lines 41–24 of the VT program. These types of schedules are known for producing superstitious behavior, as the presentation of the reinforcer may strengthen some behavior that is completely unrelated to producing the reinforcer.

We also included fixed duration (FD) and variable duration (VD) schedule of reinforcement programs. These schedules have been less common in operant laboratories often because other experiment controllers make recording response duration difficult. Our software, however, records the duration of every event in the background, allowing for duration-related contingencies to be easily created. In our FD and VD programs, reinforcement is only provided when a single response lasts long enough to meet the duration requirement. Individual response durations do not sum to meet the requirements. For example, in an FD 2 second schedule, a rat must press and hold the lever for 2 seconds before receiving reinforcement; however, two responses of 1 second each will not produce reinforcement. A good way to explore FD and VD schedules is to run several sessions of continuous reinforcement, then have the students review the data spreadsheets to

find the average and maximum response durations. Set the duration requirement (line 42 for the FD program, and lines 43–44 for the VD program) to the maximum response duration previously recorded. How does this change the response duration? Does response topography also change? See how long a response duration the students can train.

While FR and VR schedules require an increase in number of responses compared to continuous reinforcement, often resulting in an increase in response rate, differential reinforcement of high rate (DRH) schedules explicitly require an increase in response rate. In a DRH schedule, the subject is required to make a fixed number of responses within a fixed interval of time. In this sense a DRH can be considered an FR schedule with an additional time limit requirement. On line 46 of the DRH program, you can set the response requirement, or the number of responses that must occur to produce reinforcement. On line 47, you can set the interval requirement in milliseconds. In order for reinforcement to be produced, all responses must occur within the specified interval. For example, if the user requires 10 responses within 5 seconds, reinforcement will only be provided if they all occurred within the last 5 seconds. One interesting classroom exercise would be to compare performance on FR, VR, and DRH schedules to see which schedules are most successful at generating high response rate.

Differential reinforcement of low rate (DRL) schedules are designed to produce very low rates of response. Like FI schedules, in DRL schedules reinforcement is provided when a response occurs after a fixed time interval elapses. Unlike FI schedules, any response that occurs before the interval elapses resets the interval. For example, in a DRL 5-minute schedule, reinforcement will be provided for a response that occurs 6 minutes after the last reinforcer. However, a response that occurs 4 minutes after the last reinforcer will cause the interval to reset. The subject will now have to wait an additional 5 minutes before responding will produce reinforcement. You can specify the length of the response interval on line 42 of the DRL program.

Our final program is a simple extinction program. The program monitors any responses, but reinforcement is never provided. The extinction program can be used after stable behavior is established in another program to reveal additional effects of that contingency. For example, responding on VR schedules are more resistant to extinction than continuous reinforcement.

Although these experiments were created with traditional pigeon and rat laboratories in mind, they can be conducted with a much greater variety of stimuli, reinforcers, responses, and species. Invertebrates again make convenient and capable subjects. Free-flying honeybees will respond for sucrose solutions as reinforcers. Part or all of these procedures can be automated. See Abramson (1990) for some un-automated exercises and see Sokolowski and Abramson (2010) for a description of a fully automated apparatus. Crabs also respond well in operant contingencies and can learn to press a microswitch lever for a liquid food reinforcer (Abramson, 1990; Abramson & Feinman, 1990). The food can either be pumped to the crab manually to maintain students' attention, or provided automatically through an inexpensive peristaltic dosing pump for an aquarium.

Non-traditional vertebrates also make good subjects. Many species of fish can be trained using a similar method of reinforcement delivery as the crab. Fish also respond well to both LEDs and vibrating motors as stimuli (Miskovksy, Becker, Hilker, & Abramson, 2010). Some reptile species may also be suitable for classroom use. Snakes have been conditioned to press microswitches for a small amount of water reinforcement dispensed through an electronically controlled valve (Kleinginna, 1970; Kleinginna & Currie, 1979). Several designs exist for inexpensive apparatuses to study learning in aquatic turtles by providing food reinforcement (Bitterman, 1964; Pritz, Bass, & Northcutt, 1973). Heat may also be a good reinforcer for many reptile species (Kemp, 1969) and can easily be provided using a relay to turn on and off a basking lamp.

For student operant exercises, a pushbutton as a response device and an LED as a reinforcer can allow students to experience first-hand the power of schedules of reinforcement. If the students are instructed that the LEDs are reinforcers, they will typically try to earn as many LED reinforcers as possible. Students may be surprised to find that graphs of their responses closely resemble the classic graphs produced by pigeons and rats. See the video tutorials for an example of setting up a simple operant experiment with a student.

Discussion

We have found the controller to be reliable, easy to use, and extremely portable. The controller allows both the instructor and student to conduct fully or partially automated experiments depending on the need of the assignment. For example, students in our laboratory have used this controller to conduct research on habituation of the glowing response of bio-luminescent algae, the ability of planarians to seek water, and the social preferences of pigeons. These types of projects would be extremely difficult if performed manually. Many would not be possible with commercial experiment controllers. Instead of restricting the kind of research that can be conducted, as do many commercial devices, the Propeller Experiment Controller allows for a much greater range of studies to be conducted, encouraging a comparative approach.

For colleges that have no animal teaching laboratories, the Propeller Experiment Controller allows the possibility of a variety of in-home projects. Students

can use their own pets to explore principles of behavior analysis and comparative psychology. For example, with just a pedal and a feeder device attached to the controller, a dog can easily be trained to press a pedal under various schedules of reinforcement. A slightly modified fish stick (Miskovksy, *et al*., 2010) can also be easily attached to the controller for fully automated experiments with fish or other aquatic species.

An additional benefit of this device is that students become re-acquainted with apparatus design. The ability to interface an apparatus with a controller is becoming a lost skill at both the undergraduate and graduate level. Our experiment controller simplifies many of the technical aspects of creating programs for experiments, and provides an easy introduction into electronics and programming skills. Additionally, the controller is so inexpensive that any student can afford to purchase one themselves, or a school-owned device can be lent to them with little financial risk. Neither option is possible with expensive commercial equipment, further preventing students from acquiring these useful skills. The knowledge that doctoral students gain working with the Propeller Experiment Controller is also of great benefit after graduation, as these skills and the device itself can be extremely useful for a new faculty starting a laboratory with a tight budget.

Finally, the need for flexible, inexpensive equipment is especially notable in countries with less developed behavioral psychology disciplines. Behavioral teaching laboratories are noticeably absent in developing countries. This is due in part because of the high cost of teaching-related equipment (Abramson & Bartoszeck, 2006). A lack of affordable equipment for researchers and classroom demonstrations means that the study of behavior is not able to thrive as a science in these areas. For example, previous research has found that many students in northeastern Brazil do not consider psychology to be a true science, in part because few universities contain laboratories devoted to research or teaching of psychological principles (Morales, Abramson, Nain, Junior, & Bartoszeck, 2005). The Propeller Experiment Controller presents a unique solution to this problem as both research and teaching laboratories can be created with minimal financial investment. Free software, such as Google Translate (Google Inc.; Mountain View, California), makes it possible to instantly translate portions of this paper and comments in the programs into other languages. Although these translations are not perfect, they offer an easy first step toward developing teaching laboratories in other countries.

References

Abramson, C. I. (1986) Invertebrates in the classroom. *Teaching of Psychology*, 13, 24-29.

Abramson, C. I. (1990) *Invertebrate learning: a laboratory manual and source book*. Washington, D.C.: American Psychological Association.

Abramson, C. I., & Bartoszeck, A. (2006) Improving the psychology undergraduate curriculum in developing countries: a personal note with illustrations from Brazil. *Journal of Social Sciences*, 2, 108-112. DOI: 10.3844/jssp.2006.108.112.

Abramson, C. I., & Boyd, J. (2001) An automated apparatus for conditioning proboscis extension in honey bees, *Apis mellifera* L. *Journal of Entomological Science,* 36, 78-92.

Abramson, C. I., Curb, L. A., Barber, K. R., & Sokolowski, M. B. C. (2011) The use of invertebrates and other animals to demonstrate principles of learning: activities developed by The Laboratory of Comparative Psychology and Behavioral Biology. *Journal of Behavioral and Neuroscience Research*, 9, 1-6.

Abramson, C. I., & Feinman, R. D. (1988) Classical conditioning of the eye withdrawal reflex in the green crab. *Journal of Neuroscience,* 8, 2907-2912.

Abramson, C. I., & Feinman, R. D. (1990) Lever-press conditioning in the crab. *Physiology and Behavior,* 48, 267-272.

Abramson, C. I., Hilker, A. C., Becker, B., Barber, K. R., & Miskovsky, C. (2011) Cost-effective laboratory exercises to teach principles in the comparative analysis of behavior. *Journal of Behavioral and Neuroscience Research*, 9, 7-15.

Abramson, C. I., Onstott, T., Edwards, S., & Bowe, K. (1996) Classical-conditioning demonstrations for elementary and advanced courses. *Teaching of Psychology*, 23, 26-30.

Bitterman, M. E. (1964) An instrumental technique for the turtle. *Journal of the Experimental Analysis of Behavior,* 7, 189-190. DOI: 10.1901/jeab.1964.7-189.

Graham, J., Alloway, T., & Krames, L. (1994) Sniffy, the virtual rat: simulated operant conditioning. *Behavior Research Methods, Instruments, & Computers,* 26, 134-141.

Kemp, F. D. (1969) Thermal reinforcement and thermoregulatory behaviour in the lizard *Dipsosaurus dorsalis*: an operant technique. *Animal Behaviour*, 17, 466-451. DOI: 10.1016/0003-3472(69)90145-6.

Kleinginna, P. R. (1970) Operant conditioning in the indigo snake. *Psychonomic Science,* 18, 53-55.

Kleinginna, P. R., & Currie, J. A. (1979) Effects of intermittent reinforcement in the Florida kingsnake. *The Journal of Biological Psychology,* 21, 14-16.

Miskovksy, C., Becker, B., Hilker, A., & Abramson, C. I. (2010) The "Fish Stick": an easy-to-use student training apparatus for fish. *Psychological Reports,* 106, 135-146. DOI: 10.2466/pr0.106.1.135-146.

Morales, B. L., Abramson, C. I., Nain, S., Junior, N. A., & Bartoszeck, A. B. (2005) The status of psychology as a science in northeast Brazil: undergraduate student perceptions. *Psychological Reports*, 96, 109-114. DOI: 10.2466/pr0.96.1.109-114.

Place, A. J., & Abramson, C. I. (2008) Habituation of the rattle response in Western Diamondback rattlesnakes, *Crotalus atrox*. *Copeia,* 4, 836-844. DOI: 10.1643/CE-06-246.

Pritz, M. B., Bass, A. H., & Northcutt, R. G. (1973) A simple apparatus and training techniques for teaching turtles to perform a visual discrimination task. *Copeia,* 1, 181-183. Retrieved from http://www.jstor.org/stable/1442390.

Sokolowski, M. B. C., & Abramson, C. I. (2010) From foraging to operant conditioning: a new computer-controlled Skinner box to study free-flying nectar gathering behavior in bees. *Journal of Neuroscience Methods,* 188, 235-242. DOI:10.1016/j.jneumeth.2010.02.013.

Thompson, R. F., & Spencer, W. A. (1966) A model phenomenon for the study of neuronal substrates of behavior. *Psychological Review,* 73,16-43. DOI: 10.1037/h0022681.

APPENDIX A

Preparing the Propeller for Experiments

Selecting a Propeller Board

A wide variety of general-purpose Propeller development boards are available to facilitate the development of new devices without requiring the creation of unique circuit boards. These development boards are used by the engineering community to accomplish a wide variety of tasks. We recommend several development boards that have both pin headers and microSD (µSD) card sockets to permit the use of the Propeller as an experiment controller. The pin headers allow permanent or temporary connections between a device and a pin on the Propeller chip, while the µSD card socket enables the creation of data spreadsheets on removable µSD cards that can be viewed on the computer.

The Propeller Platform DNA (MGH Designs; Knoxville, Tennessee) is a general-purpose platform that includes a built-in µSD card socket. It requires minimal assembly; only two pin headers need to be soldered in place. The pin headers can be installed in several positions to facilitate permanent or temporary attachment to an apparatus. The Propeller Platform DNA contains both a barrel jack and a screw terminal for connecting power supplies. It can be powered from either connector, or by a connection to the VIN and VSS pin headers. The USB mini-B port used to program the Propeller can also provide power from a computer or USB DC adapter. Dual voltage regulators allow the Propeller Platform DNA to power both 3.3-volt and 5-volt devices. The only drawback to this platform is that it does not contain any built-in LEDs or buttons; external devices will need to be connected to test programs when the platform is not attached to an apparatus. We recommend this device for both permanent and temporary installations. The almost non-existent assembly time and high versatility make this platform excel in every context other than providing programming demonstrations in the absence of an apparatus. Note that two versions of the DNA board are currently available. The basic DNA board without the real-time clock circuit (RTC) is suitable for most experiments.

The P8X32A QuickStart is one of Parallax's directly supported development boards. It includes 8 LEDs and 8 touch pads to make programming demonstrations very easy. The QuickStart provides access to all I/O pins through a single pin header and permits the optional installation of additional pin headers. It can be powered by a connection to the VIN and VSS pin headers, or through a computer connection to the USB mini-B programming port. An additional connection between pin header #30 and VSS will allow the QuickStart to be powered from a USB DC adapter.

Unfortunately, the QuickStart does not contain a µSD card socket. If data export to the computer is required, a µSD socket will need to be installed. A number of add-on boards present an easy solution. Parallax's Human Interface Board attaches directly to the QuickStart's pin header with no soldering required. It includes a µSD socket, an infrared transmitter and receiver, a TRRS connector for stereo audio and video, a VGA connector, dual PS/2 connectors for keyboards, mice and other devices, and a barrel jack power connector. The wide variety of connectors makes the Human Interface Board very versatile, but only leaves 4 I/O pins easily usable. For many experiments 4 I/O pins may be sufficient and could be used for connections to a response device, a reinforcement device, and two stimuli. The add-on board also restricts access to the QuickStart's touch pads.

An alternative is to use Parallax's QuickStart Proto Board Kit to manually attach a µSD card socket to the QuickStart. The protoboard can be configured to attach to the QuickStart's pin header in several ways, potentially leaving the LEDs and touch pads unobscured. After installing the µSD card socket, the remaining space on the protoboard is free to use for other devices. See the video tutorials for suggestions on installing the protoboard and µSD card socket.

Once the µSD card socket is installed, a µSD card and reader are needed. Both are very common and can be purchased online or in electronic stores. Note that more expensive µSD cards with very large amounts of storage space, up to 64 gigabytes, are available. These are not necessary for the Propeller Experiment Controller; small 1 gigabyte µSD cards are sufficient. Card readers also come in a variety of shapes and sizes. The Propeller has no specific requirement, but be sure the device is able to read µSD cards, as many other SD card sizes are available.

Finally, the Propeller requires USB A to mini B programming cable to communicate with the computer as well as a power supply. The development boards we recommended function well with the 7.5 volt DC, 1 amp power supply available from Parallax's website. Several of the boards we suggested are also able to draw power from USB DC adapters. In this case, you can purchase an adapter from an electronics store and provide power to the Propeller through the USB programming cable. USB adapters uniformly use 5-volt DC outputs. Look for a device that can output at least 1,000 milliamps (1 amp).

Installing Software

Although you can use almost any computer with the Propeller, the computer does not interact with the Propeller directly. Instead, free software to download programs to the Propeller is available for PC and Macintosh at http://www.parallax.com. Click on the "Propeller" tab on the main website, then click on "Propeller Downloads." If you want to use the Propeller with a PC, click the folder icon next to "Propeller/Spin Tool

Software" to download the software. By default, the installer program places Propeller Tool at C:\Program Files\Parallax Inc\Propeller Tool. Open the Propeller Tool folder. You will notice the Propeller Tool application, a Library folder, and some other documents and folders. The Library folder contains all the supporting objects that other programs might use. If these objects are not in the Library folder, Propeller Tool will not be able to find them. Download the Propeller Experiment Controller package. The files Experimental_Event.spin, Experimental_Functions.spin, and all files beginning with "EXP_" are all objects that need to be placed in the library. Replace any existing versions files in the library with the files from the Propeller Experiment Controller package. The remaining contents of the Propeller Experiment Controller folder do not need to be placed in a specific location.

If you want to use the Propeller with a Macintosh, click on the "i" icon next to "Brad's Spin Tool (BST) for Propeller" on the downloads page. This will take you to a page with detailed instructions and links to download the software. Before installing the BST application, follow the instructions on the page to update your VCP/FDTI drivers. After downloading BST, create a folder titled "Propeller" and put the BST application in that folder. Then, download the Propeller Experiment Controller package and place it in the Propeller folder with BST. There are a few important preferences to set in BST before using the Propeller. Open BST, click the "BST" menu, then choose "IDE Preferences." Click on the "Editor Paths" tab in the newly opened "IDE Preferences" window. Click "Add" and choose the Propeller folder you created earlier to let BST know where to save programs. Now close the "IDE Preferences" window and select "Compiler Preferences" from the "BST" menu. In the "Search Paths" tab, click "Add" and choose the Propeller Experiment Controller folder to let BST know where to find supporting object programs. Finally, in the "Optimizations" tab, check the "Eliminate Unused SPIN Methods" box. This makes programs load faster and use less space.

The software for PC and Macintosh functions nearly identically. The left pane of the window shows the files you can open, while the right pane shows any programs that are currently open. Find the "Sample Experiment Codes" folder from the "Propeller Experiment Controller" folder inside the left pane of Propeller Tool or BST. In this folder, you will find our experiment programs. Open the PropellerTest program. The code should appear in the right pane of Propeller Tool or BST. This program is designed to test the QuickStart and the connection to the SD card. Connect the Propeller to the computer with the USB programming cable, and insert the SD card into Propeller. To run the program from Propeller Tool on a PC, select "Compile Current" then "Load RAM" from the "Run" menu. To perform the same task in BST on a Macintosh, select "Compile and Load to Ram" from the "Compile" menu. When the PropellerTest program starts, all of the QuickStart's LEDs will flash twice. Then a test file will be created and the LEDs will flash twice again. Once the LEDs stop flashing, you can remove the SD card. If you are not using a QuickStart board, start the program and wait about 10 seconds after loading the program before removing the SD card. Load the SD card on the computer. If all connections are correct, a file called "TESTFILE.TXT" will be created containing the text: The SD card is connected properly! The Propeller is reading and writing files properly!

Although loading the program to RAM was adequate for testing the µSD card, programs loaded to RAM will disappear after Propeller loses power. Loading programs to RAM is useful when testing programs and equipment near a computer, but it will not enable you to save a program to the Propeller and use it away from the computer easily. To permanently save a program to the Propeller, choose "Load to EEPROM" instead of "Load to RAM." Programs loaded to EEPROM will start running as soon as the Propeller turns on, and will remain on the Propeller until another program is loaded to EEPROM. Turning off the power will reset the program. Loading experiment programs to EEPROM will allow you to conduct experiments in the classroom while away from your office or laboratory computer.

Attaching Devices to the Propeller

The Propeller can interface with a wide variety of input and output devices through its DC voltage I/O pins. The development boards we recommend use pin headers to connect to various devices through jumper cables or mating pin headers. For classroom experiments, we recommend building simple circuits on solderless breadboards. Circuits are made using removable jumper cables, allowing devices to be quickly constructed and altered. The Propeller can be connected to the breadboard and attached to devices through connections to the pin headers.

Input Devices

Many types of input devices can be used with the Propeller. Microswitches are one such device that can be found in a variety of experimental equipment. Levers for animals such as pigeons and rats can be made from lever actuated microswitches with little to no modification. Other devices such as weight sensitive plates can easily be made from a combination of several microswitches. Slightly more complex devices often are designed to activate internal microswitches and can be purchased from industrial electronics suppliers. See Fig. 6 for an example of a device made from microswitches in our laboratory.

Microswitches are also very easy to use, requiring only a few parts and connections. They often have mounting holes for attaching to apparatuses. To use a microswitch with the Propeller, connect the common terminal, labeled with a "C" to the Propeller's 3.3 volt pin. You will likely want to solder a wire to the microswitch for a secure connection. Note that you can indirectly connect the common terminal to the 3.3 volt pin through the power rails on a breadboard. This is useful so that you connect multiple devices to the 3.3 volt pin. Next connect either the normally open, NO, terminal or the normally closed, NC, terminal to the Propeller's ground pin through a 10k resistor. As before, you can indirectly connect the terminal to ground through a breadboard's power rails. Do not exclude the resistor. It reduces electrical noise and ensures the Propeller does not detect false activations of the microswitch. Make another connection from NO or NC terminal to the desired I/O pin. This connection can be made anywhere before the resistor. Use the NO terminal if you want the Propeller to detect an input when the switch is pressed. When the microswitch is activated, current will flow from the 3.3 volt connection to the Propeller's I/O pin. If you use the NC terminal, current flows when the switch is not activated.

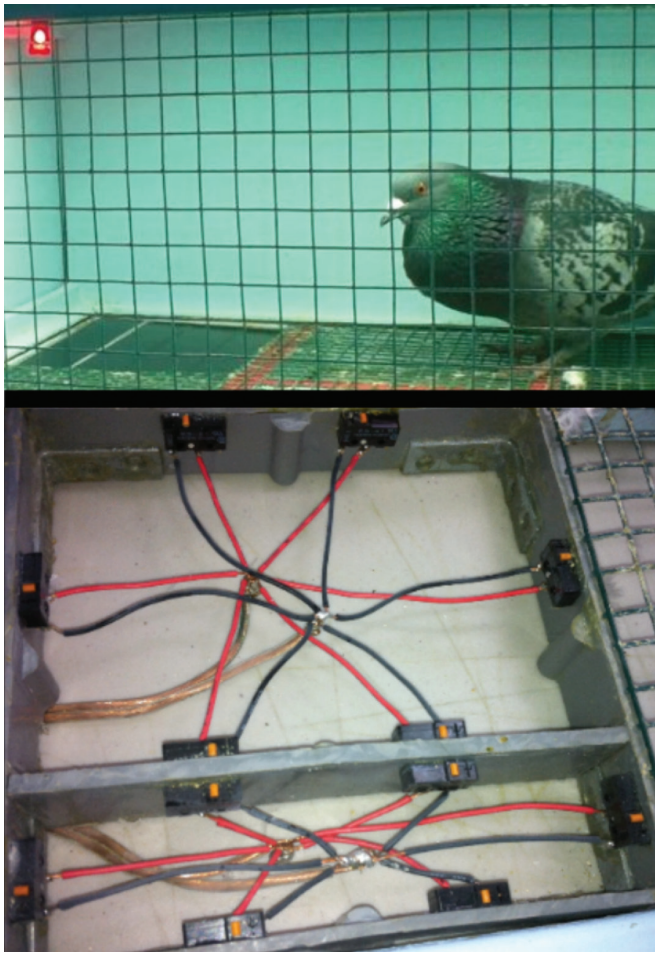

Fig 6 An example microswitch device. Top: The Propeller Experiment Controller records when the pigeon steps on the two black acrylic plates about 5 cm and 20 cm from the red light. Each plate sends a separate signal to the Propeller to indicate the general proximity to the light. Bottom: An inside view of the device reveals it is created from two sets of microswitches. In each group of switches, all common and normally open contacts are connected so that the closure of any switch in the group sends a signal to the Propeller Experiment Controller. When in use, the black acrylic plates rest directly on the microswitches.

Another method to detect an animal's behavior is through the use of infrared (IR) beams. Typically, an IR emitter and receiver pair is used to create an invisible beam of light. Any activity of the subject that breaks the beam causes a response to be recorded. Infrared beams are useful because they are invisible to many species, and because most lights do not emit a disruptive amount of infrared light. One simple way to create an IR beam is to use a dedicated IR LED and IR phototransistor pair. We recommend the LTE-302 LED and the LTR-301 phototransistor. These can be found as a pair at sparkfun.com (SparkFun Electronics; Boulder, Colorado). The IR LED can be connected just as any other LED. Connect the longer leg of the LED to the Propeller's 3.3 volt pin through a small resistor, such as 220 ohms. Connect the shorter leg of the LED to ground. The IR phototransistor will have two or three legs. Consult the phototransistor's documentation to identify the legs. Connect the leg labeled "emitter" to ground. Connect the leg labeled "collector" to the 3.3 volt pin through a 10 kiloohms resistor. Also connect the collector to the Propeller's desired I/O pin. The base pin, if available, does not need to be connected for this circuit. When infrared light from the IR LED hits the phototransistor, current will flow from the collector to the emitter, activating the Propeller's I/O pin in the process. Keep in mind that placement of the IR LEDs and IR phototransistors is key, as multiple IR beams in close proximity or direct sunlight may interfere with proper detection of responses. Positioning the components correctly can be made easier through digital cameras, as many, including those found in cellular phones, can detect the IR light.

One-piece reflective IR sensors also exist. These parts can be used to detect the subject's activity in a similar manner, except that only one part needs to mounted on the apparatus. Often they can be 'tuned' to detect objects only within a certain range. Other reflective IR sensors can measure the distance between the subject or some other object and the sensor. These types of IR sensors require some changes in the experiment programs. Please contact us if you are interested in using reflective IR sensors.

For some experiments, it is useful for students to observe and record the subject's behavior manually. We

suggest using pushbuttons to record behavior because actively monitoring the subject maintains student attention, and recording behavior is easier and more accurate than when using traditional pencil and paper scoring. Using a pushbutton to manually record behavior also allows the students to continue observing the animal while recording a response. Microswitches can also be used for this purpose but take a little more time to install.

Pushbuttons are easy to use with the Propeller. On the bottom of a pushbutton you will notice a line splitting four legs into two groups. The legs in the same group are always connected. The legs in opposite groups are only connected when the button is pushed. To use the pushbutton as an input device, place it in the center of the breadboard. Connect one set of legs to the Propeller's 3.3 volt pin. Connect another set of legs to the ground pin through a 10 kiloohms resistor, and connect this set of legs to the desired I/O pin. When the button is not pushed, no current will reach the I/O pin. When the button is pushed, current will flow from the 3.3 volt connection to the Propeller's I/O pin.

These simple inputs will suffice for most experiments. The Propeller is also capable of more complex forms of detection. Several input boards, such as Parallax's Digital I/O Board Kit or Futerlec's (New York City, New York) input boards, can make connecting the Propeller to an apparatus much easier. Other specialty input devices can be found at parallax.com, adafruit.com (Adafruit Industries; New York City, New York) and sparkfun.com, and include finger print sensors, joysticks, light sensors, methane sensors, microphones, tilt sensors and thermometers.

It is important to note that any input over 3.3 volts will damage the Propeller. If you have some input device that produces more than 3.3 volts, you can reduce the voltage to a safe level by using a pair of resistors to create a voltage divider. Voltage divider calculators found online can provide the values of the resistors needed to reduce the voltage level of your input down to 3.3 volts. Many of the useful input devices found at adafruit.com, futerlec.com, and sparkfun.com produce 5 volt signals. In addition to reading the documentation for a device, it is also often useful the check the voltage level with a multimeter.

Output Devices

The Propeller can also interface with a variety of output devices. LEDs are one type of output that can be used for a variety of purposes. They make great stimuli for classical and operant conditioning procedures. In classroom demonstrations LEDs make great reinforcers for students. One good exercise is to have students compete to see who can earn the most LED reinforcers. This keeps the students motivated during their own performance as well as interested in their classmates' performances. LEDs can be powered directly by the Propeller's I/O pins. Simply connect the longer leg of the LED, the anode, to the I/O pin through a small resistor, 220 ohms should be fine. Connect the shorter leg, the cathode, of the LED to ground. LEDs can also be dimmed using a programming technique called pulse-width modulation (PWM). Experimental Functions provides this technique. See Appendix B for a simple, 4-step modification of any of our experiment codes to use PWM with an LED.

Piezo speakers present a simple way to play audio stimuli. As with the LED, connect the longer leg of the speaker to the I/O pin and connect the shorter leg to ground. Occasionally, the legs are the same length and a plus symbol will denote the end to connect to the I/O pin. If you used a QuickStart with Human Interface Board, headphones or speakers can be connected directly to the Propeller. Note that tones are generated with rapid oscillations in current. Turning the speaker on in the same manner as an LED will not produce a tone. Experimental Functions provides a frequency generation technique that can produce tones. See Appendix B for a simple, 4-step modification of any of our experiment codes to use a frequency generator to produce sounds.

Some devices need more power than can be provided by the Propeller's 3.3 volts DC I/O pins. For example, many pellet dispensers are powered by motors that require more than 3.3 volts. Vibrating motors also make excellent stimuli in habituation experiments and can also be used as noisemakers when attached to hollow objects. Cooling fans can act as reinforcers for hot animals. Puffs of air from common aquarium pumps can be used to elicit eye-blinks and other reactions. Finally, many lights used for stimuli or to illuminate an experimental apparatus also require more power than the Propeller can directly provide. Fortunately, there are several ways the Propeller can indirectly provide ample power for these types of devices.

Transistors are one simple option to allow the Propeller to indirectly provide more DC voltage to a device. The general function of a transistor is to provide a higher voltage to one device when it receives a lower voltage signal from another device. We recommend the common 2N2222A or 2N3904 transistors for most purposes. Transistors are three-legged devices. Consult the documentation to determine which legs are the base, emitter, and collector. Connect the Propeller's I/O pin to the base through a 10 kiloohms resistor. Then connect the emitter to the ground of the higher voltage supply, and the collector to the power supply of the higher voltage. Insert your device in series with the power supply and the collector. When the I/O pin turns on and powers the base, the transistor connects the collector and emitter to power the device. Convenient integrated circuits, such

as the ULN2803A Darlington array, contain a series of transistors in one small package. The ULN2803A also contains internal resistors before the base of each transistor, so an additional resistor is not required.

Devices that produce electromagnetic fields, such as DC motors and DC fans require a slight addition to the circuit to protect the Propeller. In this case, connect a diode in parallel with the device with the cathode connected to the power source and the anode connected to the collector. The ULN2803A already contains this protective diode. Relays can be used for devices that require even more power than a transistor can provide. Many relays can also switch AC current. Relays can be connected to the Propeller using the same transistor and diode circuit used for motors and fans. For a quickly assembled relay kit, see Parallax's Dual Relay Board Kit. For a more versatile relay board, see Parallax's Digital I/O Board Kit, or Futerlec's several relay boards.

The speed of motors and fans, as well as the brightness of many lights can be controlled with pulse-width modulation (PWM) techniques. See Appendix B for a simple, 4-step modification of any of our experiment codes to use PWM with a motor or fan or light. Note that PWM only works for DC devices connected directly to an I/O pin or indirectly through a transistor.

APPENDIX B

Modifications to Enable Frequency Generation and Pulse-Width Modulation

The following instructions will allow you to modify any of our programs so that an output device such as a speaker can be turned on and off at certain frequencies to generate a tone, or other devices such as lights and motors can be pulse-width modulated (PWM) to produce a change in brightness or speed of the device. Note that for these instructions, we refer to variables such as "eventpin" or "event.record" to refer to the event you wish to modify. In the programs, the word event will be replaced by a more specific term such as "reinforcer" or "CS." Substitute the appropriate term for "event" when modifying the code. A modification of the classical conditioning program titled "ClassicalConditioningTone" is available as an example of a program modified to generate a tone as a CS instead of activating a standard on/off device.

Step 1

Every experiment has a method called "SetVariables." In this section, replace the code "event.declareoutput(eventpin,exp.clockID)" with "event.declaremanualevent(exp.clockID)." Remember that the word "event" in these instructions is a placeholder for the actual variable in the program. Make sure to use the appropriate term for the program you are modifying. For example, if you are modifying the CS event in the classical conditioning program, replace "event" with "CS." The resulting modified code would be "CS.declaremanualevent(exp.clockID)."

Step 2

In the "SetVariables" method, you will need to add one additional line. If you want to enable frequency generation, add the line "exp.StartFrequencyGenerator-(eventpin,-1)." If you want to use pulse-width modulation, add the line "exp.StartPWM(eventpin)" instead.

Step 3

Every experiment has a method related to starting a stimulus such as "StartReinforcement" or "StartCS." In this method, replace the code "exp.record(event.turnon, event.ID, exp.time(start))" to "exp.record(event.startmanualevent, event.ID, exp.time(start))." You will also need to add one line to this method. Add "exp.setfrequency(event.pin, frequency)" for frequency generation, or add "exp.setPWM(event.pin, percent)" for pulse-width modulation. In this statement replace the variables frequency and percent with the desired values. Frequency is provided in hertz. A list of musical frequencies can be found in Experimental Functions. For example, the note A4 is 440 Hz. The percent variable for pulse-width modulation is a value from 0 to 100 that refers to the apparent intensity of the device. For example, lights normally appear at 100 percent brightness, but you can dim them with PWM to 50 percent brightness.

Step 4

Every experiment also has a method related to the stopping of a stimulus such as "StopReinforcement" or "StopCS." In this method, change the line "exp.record(event.turnoff, event.ID, exp.time(start))" to "exp.record(event.stopmanualevent event.ID,exp.time(start))." You will also need to add the line "exp.setfrequency(event.pin, 0)" for frequency generation, or add "exp.setPWM(event.pin, percent)" for pulse-width modulation.

APPENDIX C

Propeller Python Interface

We also included a Python program that can be used with the Propeller Experiment Controller. The purpose of this program is to process memory files created during the experiment and create data spreadsheets in the rare scenario that the Propeller Experiment Controller cannot do this automatically. This should only occur for two reasons. First, if more than 1,000 instances of an event occurs, such as 2,000 lever presses, the Propeller Experiment Controller will not be able to process that event. Second, if the user turns off the Propeller Experiment Controller before the experiment is complete or removes the SD card prematurely, the controller will not

be able to create the data spreadsheets. In either case, the memory file will still be available on the SD card. Even if the experiment was accidentally terminated, the memory file has a record of all events that occurred up to termination. The Propeller Python Interface program can then create a data spreadsheet from those events.

To use the Propeller Python Interface program, you will need to install the Python programming language from python.org. Python is freely available and is already factory installed on many computers. We recommend installing version 3.1.1. Once Python is installed, simply open the file titled "Python Propeller Interface" in Python's IDE. Then choose "Run Module" from the "Run" menu. Instructions will be presented on screen. You can type "instructions()" for details on different methods to process a memory.txt file.

APPENDIX D

Additional Resources

CAVarnon.com (CAVarnon.com/experiment-controller)	Home of the Propeller Experiment Controller software. Website includes the most recent version of the software as well as instructional material.
Parallax Inc (http://www.parallax.com/)	Producer and distributor of Propeller products. Website includes tutorials, a discussion forum, and a program database.
Parallax Semiconductor (http://www.parallaxsemiconductor.com/)	Additional information on the Propeller, specifically the QuickStart.
Propeller Powered (http://propellerpowered.us/)	Propeller products and add-on boards for the QuickStart.
Gadget Gangster (http://www.gadgetgangster.com/)	Propeller products and add-on boards for the QuickStart.
MGH Designs (http://mghdesigns.com/)	Propeller products including the Propeller Platform DNA.
Adafruit Industries (http://www.adafruit.com/)	Vendor of a wide variety of sensors and other devices for microcontrollers.
Allied Electronics (http://www.alliedelec.com/)	Industrial electronics supply.
Futurlec (http://www.futurlec.com/)	Vendor of a wide variety of components, hardware, boards and other devices for electronics and microcontrollers.
Sparkfun Electronics (http://www.sparkfun.com/)	Vendor of a wide variety of sensors and other devices for microcontrollers.