9

FUTURE DIRECTIONS AND CONCLUSION

Future Directions

The Propeller Experiment Controller (PEC) was developed to meet the needs of research in our laboratory, beginning with a master's thesis on sexual conditioning in pigeons (Varnon, 2013), and the development of an inexpensive conditioning and learning laboratory (Varnon and Abramson, 2013). As we continued to use the PEC in our laboratory, additional features were added to solve unique problems that we encountered. For example, the methods to record and save raw data events were developed specifically to help a student record light emitted by bioluminescent algae. Development of raw data events allowed us to investigate, not only if the algae illuminated, but also the brightness of the light. Many such features have been added to the PEC over the course of its development, and these additions have proved to be instrumental to using the Propeller as an experiment controller. In the following sections, we describe some potential areas of future development that may also be useful in many applications of the PEC.

Integration with Python for the Personal Computer

One potential area of growth for the PEC is to develop supporting software for the personal computer. To that end, we have begun developing an interface for the PEC using the Python programming language (Python Software Foundation, Python.org). Python is a useful language for this project because it is free, runs on any operating system, is very easy to use, and even aesthetically resembles Spin in some manners. The Propeller Python Interface (PPI) is still in its infancy but could provide many advantages.

First, use of the PPI can solve the maximum instances problem of the PEC. The PEC has a limit to the number of instances of a single event that can be saved to the data spreadsheet. By default, only 1,000 instances of an event can be saved. Although there is no practical limit to the number of events that can be recorded to the memory file during an experiment, processing that information and saving it to the data spreadsheet has additional requirements. The limit is encountered when the Propeller attempts to read the entire memory file for information about a specific event, such as a lever-press. The Propeller uses the very concise information recorded in the memory file to calculate the more detailed information that appears in the data spreadsheet. As the Propeller's RAM is limited, it simply cannot process the data for more than 1,000 instances of an event using the techniques currently implemented in Experimental Functions. Personal computers do not have the same memory limitations as the Propeller, and therefore use of a computer to create the data spreadsheet from the memory file can solve the maximum instances problem. Currently, the PPI does have a rudimentary ability to read the PEC's memory file, and then transform that information into a data spreadsheet. Unfortunately, the program is not user-friendly, nor is it well-tested. In principle, however, it shows great promise. Use of such a program on a personal computer raises the maximum instances limit substantially. For example, using a 64-bit computer, the maximum instances of a single event that can be processed is over nine quintillion (as determined by Python's sys.maxsize method).

Second, the PPI could provide additional data analysis and sorting techniques, such as counting the number or duration of events that occurred in specific time interval bins. Such "binning" techniques can be conducted by the PEC, but only if they are carefully programed in advance. A post hoc method to sort the data may be useful. The PPI could also provide integrated statistical analysis and graphing through the SciPy package (Jones et al., 2001), a free scientific analysis package for the Python. We have made use of SciPy techniques in many projects, but we have not formally integrated these techniques with the PPI.

A third benefit that could be provided by the PPI is to allow the Propeller to interface with the computer during an experiment. This could enable data to be displayed on a computer. It could also enable commands to be sent from the computer to the Propeller, controlling experimental parameters or some other aspect of the experiment. Interfacing the Propeller with a computer during an experiment could be accomplished in two ways. A modification of Experimental Functions's Record and RecordRawData method could send information to the computer through the USB programming cable using a serial protocol. This would enable a simple modification of existing programs to send data to a personal computer instead of an SD card. Alternatively, the Experimental Event and Experimental Functions objects could be translated into Python, allowing the computer to control the Propeller directly. Both techniques could be useful. While one major strength of the PEC is that it does not require a constant connection to a computer, some things would only be possible with such integration. For example, integration would allow graphs to be displayed on the computer monitor as the data are collected and updated in real time. A full translation of the PEC to the Python programing language may also make the PEC more accessible to individuals that are already familiar with Python.

Translating to C/C++

At the time the PEC was developed, the Propeller's custom language, Spin, was the most convenient option. Now, however, there are many easy options, including the popular C and C++ programming languages. With the advent of Simple IDE, Parallax began officially supporting C/C++ programing for the Propeller. Translating the PEC to C/C++ would likely make the PEC more accessible, as many programmers have some experience in these languages. The Arduino family of development boards (Arduino; New York, New York), in particular, has popularized programing microcontrollers with C/C++. Although the Propeller and the PEC offer many benefits compared to other systems, some users may be hesitant to make the switch because of the perceived difficulty of learning a new language.

Conclusion

The PEC provides a unique opportunity to implement many types of behavioral experiments at low cost without sacrificing technological power. In many cases, the PEC is only limited by the imagination of the user. Although programming skills are required to make the most of the PEC, this is true for many systems. Even behavioral control systems that advertise that programming is not required often use a visual form of programming that provides limited control. For those without programming experience, we offer a set of prewritten programs with modifiable experimental parameters (Varnon and Abramson, 2013; CAVarnon.com). For those with some programming experience, the PEC offers a world of opportunity.

The PEC also makes a very affordable foundation for teaching laboratories. Computer simulations, such as Cyber Rat((AI)^2; Winter Park, Florida) and Sniffy the Rat (Graham, Alloway, and Krames, 1994), have been frequently used as replacements for inquiry-based learning exercises with live animals, in part because the hardware and software to implement automated behavioral experiments is so expensive. Although potentially useful, we do not believe that exclusively relying on computer simulations provides the same benefits as interacting with a live animal. The PEC is a great alternative to computer simulations because it can be implemented with a reasonable budget. Depending on the exercises and species used, the laboratory can also be taken to existing classrooms, instead of requiring a dedicated room to contain computers and equipment too large and cumbersome to transport.

The PEC may also promote scientific development in graduate and undergraduate students. Based on our own personal experience, we believe that greater student involvement in the areas of apparatus design and automation leads to greater interest. Many of the programing skills learned while working with the PEC can also be of great use after graduation. Unfortunately, apparatus design is becoming a lost art. Interested students are often prevented from learning apparatus related skills due to the high cost of commercial equipment. As the PEC is so inexpensive, students are able to purchase or borrow a Propeller without much financial risk. This may encourage students to learn how to construct their own equipment and even permit them to conduct behavioral research and automated demonstrations with farm animals or household pets.

In conclusion, the PEC is a powerful and versatile device that presents many opportunities for anyone interested in a scientific study of behavior. From data-loggers for a single sensor, to devices to coordinate and control several experimental apparatuses, the PEC can do it all. We will continue to provide software, and instructional material related to the controller at CAVarnon.com with the goal of keeping projects related to the controller as open-source as possible. Much like open-access journals, we expect that open-source technologies such as the PEC will greatly increase research novelty and productivity. We also hope that making the PEC available and easily accessible will help encourage a new generation of behavioral researchers.