A low-cost, computer-controlled robotic flower system for behavioral experiments

Erno Kuusela & Juho Lämsä

Department of Ecology, University of Oulu, PO Box 3000, Oulu 90014, Finland

Keywords
Arduino, automatization, bumblebee, data acquisition, electronics, feeder, infrared sensor, microcontroller, pollination, servo.

Correspondence
Juho Lämsä, Botanical Garden, University of Oulu, PO Box 3000, Oulu, FI-90014, Finland. Tel: +358503440532; Fax: +358 8 553 1061; E-mail: juho.lamsa@oulu.fi

Funding Information
Oskar Öflunds Stiftelse, Oulu University Scholarship Foundation, Finnish Cultural Foundation.

Received: 8 December 2015; Revised: 29 January 2016; Accepted: 9 February 2016

Ecology and Evolution 2016; 6(8): 2594–2600
doi: 10.1002/ece3.2062

Abstract

Human observations during behavioral studies are expensive, time-consuming, and error prone. For this reason, automatization of experiments is highly desirable, as it reduces the risk of human errors and workload. The robotic system we developed is simple and cheap to build and handles feeding and data collection automatically. The system was built using mostly off-the-shelf components and has a novel feeding mechanism that uses servos to perform refill operations. We used the robotic system in two separate behavioral studies with bumblebees (Bombus terrestris); The system was used both for training of the bees and for the experimental data collection. The robotic system was reliable, with no flight in our studies failing due to a technical malfunction. The data recorded were easy to apply for further analysis. The software and the hardware design are open source. The development of cheap open-source prototyping platforms during the recent years has opened up many possibilities in designing of experiments. Automatization not only reduces workload, but also potentially allows experimental designs never done before, such as dynamic experiments, where the system responds to, for example, learning of the animal. We present a complete system with hardware and software, and it can be used as such in various experiments requiring feeders and collection of visitation data. Use of the system is not limited to any particular experimental setup or even species.

Introduction

Use of artificial flowers in pollination experiments has many advantages compared with field studies, including a comprehensive control of the environment. The easiest way of conducting such a laboratory experiment is to establish a group of artificial flowers in an experiment arena and refilling their nectar reservoirs by hand, and then making observations visually (e.g., Cartar and Real 1997; Lynn et al. 2005). However, such a system is labor-intensive and the experimenter is required to watch the animals constantly. In comparison, the benefits of an automated system are obvious: There is no need to refill nectar manually, and data collection about visitations to flowers is more reliable using sensors rather than visual observations by the researcher. This is especially the case when dealing with rapidly moving animals, such as most pollinators.

According to Essenberg (2015), the first automatic electronic flowers were developed in the 1970s (Grossmann 1973; Hartling and Plowright 1979). Subsequently, various systems have been developed to provide sugar solution either in continuous fashion (e.g., Leadbeater and Chittka 2008; Ohashi et al. 2010) or in discrete doses, such as designs using an electromagnet (e.g., Keasar 2000; Cnaani et al. 2006; Lihoreau et al. 2010). Regardless of the type of feeding mechanism, light sensors are commonly used to track the movements of pollinators (e.g., Keasar 2000; Winter and Stich 2005; Cnaani et al. 2006; Sokolowski and Abramson 2010).

Here, we describe an artificial flower system that is fully automatic in terms of its refill operation and data collection and that uses a novel feeding mechanism to deliver discrete doses of sugar solution. The system we designed is highly reliable, relatively easy and cheap to construct, and uses mostly off-the-shelf components. Although the original system was developed for bumblebees (Bombus terrestris, L.), a common model species in behavioral and pollination experiments, its hardware parts can be scaled up or down to suit other species.
System Description

The system includes a control unit, separate flowers, and a personal computer (Fig. 1). The function of the control unit is to handle the electronics of the flowers, collect data from them, and to send data to the computer. The flowers themselves contain an infrared (IR) sensor, their associated electronics, and an electromechanical device (servo) that offers a small, precise amount of sugar solution (referred from hereafter as “nectar”) from a reservoir (Fig. 2). The computer runs software that controls the refilling rate and data collection via the control unit (Fig. 3).

Control unit

The main responsibilities of the control unit are as follows: (1) low-level handling of the electronics of each flower, (2) collection of visitation data from the sensors, and (3) interpretation of the commands from the computer.

The heart of the control unit is Arduino Mega 2560 single-board microcontroller platform (http://www.arduino.cc), which is based on ATmega2560 processor chip by Atmel (http://www.atmel.com/, San Jose, CA). The microcontroller is connected by ribbon cables to two separate connector cards which contain components needed to connect the flowers. The microcontroller is powered by USB connection to the computer, whereas flowers receive electricity from a separate power supply via connector card. Arduino Mega 2560 can simultaneously use up to 32 flowers.

We used a standard 300 W PC ATX power supply to provide +5 Volt direct current for all 32 flowers in the system. Current draw of a single flower can be up to 200 milliamperes (mA) or even more, depending upon the type of servos used. As the microcontroller platform is powered by computer’s USB port, its maximum current output is limited to 500 mA, which is not enough for all flowers, and therefore, a separate power supply unit is needed.

The program running on the microcontroller (hereafter referred to as the “firmware”) handles the electronics in the system: It reads infrared sensors at specific intervals and generates correct electrical pulses to drive the servomotors. The firmware is operated by sending specific commands to it via the USB cable, which describe some
simple operations, such as “refill flower #8”. The control unit sends back data collected from the sensors.

Infrared sensors are constantly read at 10 Hz frequency (i.e., 10 times per second). The control unit uses the following algorithm to determine the time and length of a visit: When an animal enters a flower, and sensor detects it, system time (seconds elapsed since the start of the experiment) is recorded. Simultaneously, a timer is started; this timer value is incremented at 100 ms intervals and zeroed after it reaches the value 10, that is, timer resets once per second. Positive readings (that is, when the animal is detected) from IR sensor are counted during this 1-sec time period, and if detection count is greater than a certain preset value (value 2 was used in our first study), a variable containing the visit length is incremented by one. Otherwise, if detection count fails to reach this preset value, visit to flower is determined to be ended. The minimum length of a visit is 1-sec; in reality, the actual visit may be shorter than this (for example, 0.3 sec), but the algorithm rounds value up to the next full second.

This algorithm was designed because bumblebee behavior may not be consistent during every visit; instead of going straight into the flower, for example, the animal may go in and back out quickly, and then go into the flower again. Without our algorithm, the system would detect each such erratic movement as a separate visit and interpreting the data would be much more difficult.

**Figure 3.** User interface of the control software.
During a refill operation, the servo arm turns 60° to position the nectar cup directly below the feeding hole. A lid covers the reservoir, and mounted on top of the lid is the servo and a small circuit board. A rectangular hole is cut to the lid, through which the servo arm operates. Nectar is available to the animal via a 3-mm feeding hole directly above the nectar cup. A short plastic tube is vertically attached to the lid, so that the feeding hole is at the bottom of a "pit" where animal has to crawl. Mounted on the sides of the tube are an infrared LED and a phototransistor, which detect the feeding visit. Peak spectral wavelength of the IR LED is 940 nm, and the phototransistor was selected to spectrally match the LED. When the animal crawls into the sensor tube, it blocks the passage of the IR light beam, causing the phototransistor to shut off; this in turn is sensed by the control unit.

On top of the lid is the top cover, which protects the circuit board and servo. Different kinds of symbols can be attached to the top cover to symbolize, for example, flower corolla. Another function of the top cover is to hide the moving parts of the refill mechanism (servo and servo arm) so that the animals cannot visually detect when the flower is refilling the nectar cup.

**The flowers**

Each flower in the system is a self-contained unit and operates under the control unit’s command, independently of the other flowers. Power and data connection to a flower is provided via a four-wire cable, which was 5–10 m long in our system. The flower itself has a nectar reservoir (approx. 300 mL volume), from which the nectar cup is refilled. In our system, the nectar cup was a small Phillips screw head, which had a volume of 1.7 μL. The cup is fixed to an end of a steel wire, which in turn is attached to servomotor’s shaft via a servo horn (this assembly with servo horn, steel wire, and nectar cup is referred to as “servo arm”). The empty weight of a single flower is 50 g without a cable.

When in its normal position, the arm is held horizontal to position the nectar cup directly below the feeding hole. During a refill operation, the servo arm turns 60° downwards and submerges the cup into the nectar reservoir to fill it; after this, the servo is returned to the normal position. The exact position of the nectar cup can be finetuned simply by bending the steel wire arm by hand. The refill cycle takes 14 sec; while a faster rate of refill is technically possible, we found during testing that this led to periodic formation of a small air bubble in the nectar cup we used, thereby preventing it from filling up completely.

A lid covers the reservoir, and mounted on top of the lid is the servo and a small circuit board. A rectangular hole is cut to the lid, through which the servo arm operates. Nectar is available to the animal via a 3-mm feeding hole directly above the nectar cup. A short plastic tube is vertically attached to the lid, so that the feeding hole is at the bottom of a “pit” where animal has to crawl. Mounted on the sides of the tube are an infrared LED and a phototransistor, which detect the feeding visit. Peak spectral wavelength of the IR LED is 940 nm, and the phototransistor was selected to spectrally match the LED. When the animal crawls into the sensor tube, it blocks the passage of the IR light beam, causing the phototransistor to shut off; this in turn is sensed by the control unit.

On top of the lid is the top cover, which protects the circuit board and servo. Different kinds of symbols can be attached to the top cover to symbolize, for example, flower corolla. Another function of the top cover is to hide the moving parts of the refill mechanism (servo and servo arm) so that the animals cannot visually detect when the flower is refilling the nectar cup.

**Computer and control software**

The control software running on the computer is responsible for sending commands to the control unit and receiving flower visitation data from it. The software was written in Java programming language and works stably at least on Windows XP and Windows 7 operating systems.

Before starting an experiment, the system is configured using the control software. Each flower’s refill rate can be individually configured, and if needed, the refill mechanism and/or sensor of specific flowers can be disabled. The refilling rate can be fixed (with the operation performed at steady intervals) or it can include random variation (within an adjustable range). Once an experiment is started, the software controls the experimental setup automatically, refilling flowers according to the predefined configuration, and gathering sensor data from them. When the user decides to stop the experiment (via the control software), the sensor data are formatted and saved automatically as a Microsoft Excel file.

When an experiment is not running, the flowers can be commanded to refill themselves manually. The servo arms can also be stopped and held in downward position, a feature that proved useful in some maintenance operations (such as cleaning of the nectar cups).

We also included a “deception mode” feature in the control software and firmware: In this mode, the servo arm is not brought to a fully upward position, so that the nectar cup is below the feeding hole but still inaccessible to the animal. The purpose of this feature is to simulate deceptive flowers, such as some orchids, which do not produce nectar. Independent flowers can be configured to be in deception mode; they respond to refill commands such as flowers in “normal” mode, but keep the nectar cup out of the reach of the animal.

**Price**

See Table 1.

**Discussion**

We used this robotic system in two separate behavioral studies during 2014 and 2015 (J. Lämsä, E. Kuusela, S. Juntunen, A. Jälkäniemi, P. Watts and J. Tuomi, unpubl.). A training phase was arranged in a separate nest cage with five flowers and an experiment phase

---

© 2016 The Authors. *Ecology and Evolution* published by John Wiley & Sons Ltd.
Table 1. Rough price estimate for the parts of the robotic system (with 32 flowers, not including the computer).

| Part                    | Cost  |
|-------------------------|-------|
| Flower                  |       |
| Cable (5 m)             | 3 US$ |
| Servo                   | 2 US$ |
| Electronics             | 3 US$ |
| Structural parts (reservoir, lid, etc.) | 2 US$ |
| Flower total            | 10 US$ (10 US$ x 32 = 320 US$) |
| Control unit            |       |
| Arduino Mega 2560       | 50 US$|
| Electronics             | 20 US$|
| Power supply            | 30 US$|
| Control unit total      | 100 US$|
| System total            | 420 US$|

Conducted in a flight arena that was connected to the nest cage via a gated tube. The flight arena had 27 flowers, and the same system operated flowers in both cages simultaneously. Bumblebees readily visited the flowers. Based on our observations, the bumblebees were unable to detect the refill operations before actually probing the feeding hole.

System maintenance consisted of periodic cleaning of the flowers and exchanging of old nectar with fresh. Some of the servos in nest cage flowers had occasional malfunctions after long continuous use (several weeks of 24 hours per day operation) and were replaced. Servos failed because their internal mechanical parts wore out, which probably could be avoided by purchasing higher quality servos. However, not a single experiment flight in either of our studies failed because of these malfunctions, so overall reliability of the system can be considered good. The servo replacement was also relatively quick and easy, requiring about ten minutes of work. This delay was avoided by using spare flowers.

The servos used in this system were “micro” size class analog servos commonly used in remote-controlled model aircrafts. These are nowadays widely available and cheap (as low as 2 US$ per piece, Table 1). Servos and all the other electronic parts can be purchased from well-equipped electronics stores or online. Structural parts were purchased from regular hardware stores and supermarkets.

The lack of critical values in the structure of the feeding mechanism is a strength of our design. The size and positioning of parts relative to each other can vary, and once adjusted, there is virtually no variation in the repeated movements of the servo arm. Also, unlike float-and-magnet type feeding mechanisms (e.g., Lihoreau et al. 2010), our design avoids the problems associated with variation of fluid level in the nectar reservoir. As the servo arm is mechanically attached to the servo, there is also no risk of misalignment of the nectar cup in relation to the feeding hole as may occur when using a float-and-magnet design. Note, however, that these conclusions are only based on our experience with our own prototype device that used a magnet and a float, which we discarded in favor of the design presented in this paper.

The refill mechanism also avoids overfilling, whereby excess nectar accumulates in the feeding area. Overfilling might be a risk in designs that use a pump-type delivery mechanism (e.g., Leadbeater and Chittka 2008), and while overfilling can be avoided to some degree using sensors (e.g., a new dose is delivered only after an animal has visited the flower), overfilling can still be problematic if part of the nectar dose is left in the feeding area. Also, we decided against a pump-type design because they require high-precision parts (e.g., syringes, reduction gears) that can be expensive.

The visitation detector is simple and reliable, and the general design has been used in other artificial flower systems (Keasar 2000; Sokolowski and Abramson 2010). By mounting the detector on the side walls of a tube where the animal has to crawl into, the risk of getting false readings is smaller than using a detector system where sensors are placed around the feeding hole on the surface of the flower.

This system can be used without modifications for a wide range of behavioral experiments with bumblebees and with minor modifications for many other pollinators. For example, the sensor tube can be scaled down to suite smaller members of the Apidae (such as honeybees). With larger structural changes to the flowers, the system could possibly be used to feed butterflies and maybe even with hummingbirds and flower bats. With major hardware changes, the system could be used as a general feeder to many other types of animals, such as mice, dogs, and pigeons, as the only critical mechanical parts are the standardized servos, which are widely available in different sizes. The flowers in the current design are lightweight and compact (90 mm diameter), offering flexibility in experimental designs.

Flowers can be scaled down by placing the parts (circuit board, sensor tube, and servo) more tightly on the lid, and using a smaller diameter nectar container. We estimate that this can reduce the flower diameter by 2–3 cm. We used “micro” class servos, but there are smaller ones available (“submicro” and “nano” class), which could be used to reduce the diameter even further (about 0.5–1 cm). The height of the flower (approx. 7 cm total in our system) is mostly limited by the length of the servo arm, which has to have enough vertical room to turn to the downward refill position. If the diameter is made smaller, the height can also be lower, as the servo...
arm is shorter in that case. The circuit boards of the flowers can be made more compact by fabricating them instead of using stripboard. The circuit board can actually be omitted altogether, by soldering components directly to the wires (the so-called “dead bug” design) if a very small flower is desired. Also note that the flower does not have to be round as in our case, as the only factors limiting the geometry of the nectar container are the height and width in the servo arm’s direction, which has to have enough space to allow the refill movements. Scaling up is more straightforward, simply using larger parts and if necessary, more powerful servos.

Both the firmware and control software are released under open-source license and can be freely modified. The control software we provide here is flexible enough to be used in various experiments without modifications. However, modification of the source code allows the control program to be tailored to run any specific experiment.

One interesting possibility for modification is to change the behavior of the program from predetermined to dynamic. In the current version, the program runs according to preset parameters throughout the experiment. However, the rate of refill operations of independent flowers could be adjusted dynamically in response to learning of the animal, based on visitation data from sensors. For example, the refill rate of a specific (type of) flower could be increased or decreased when it is visited more often than other (types of) flowers.

Conclusion

We have developed a complete, affordable, and automatic robotic flower (feeding) system for pollination and behavioral experiments. Our system not only handles flower refill operations but also data collection and storage, in a user-friendly way. All software and hardware design are open source and freely modifiable.

The development of low-cost open-source microcontroller prototyping platforms in recent years made this system possible. We encourage other researchers to explore the possibilities of this approach.

Acknowledgments

We thank the Finnish Cultural Foundation, Oskar Öflunds stifelse, Oulu University Scholarship Foundation and staff at Oulu University Botanical Gardens for making this work possible. Also thanks to V.-M. Kangas, P. Watts and two anonymous reviewers for constructive feedback.

Conflict of Interest

None declared.

References

Cartar, R. V., and L. A. Real. 1997. Habitat structure and animal movement: the behavior of bumble bees in uniform and random spatial resource distributions. Oecologia 112:430–434.

Cnaani, J., J. D. Thomson, and D. R. Papaj. 2006. Flower choice and learning in foraging bumblebees: effects of variation in nectar volume and concentration. Ethology 112:278–285.

Eisenberg, C. J. 2015. Flobots: robotic flowers for bee behaviour experiments. J. Pollinat. Ecol. 15:1.

Grossmann, K. E. 1973. Continuous, fixed-ratio, and fixed-interval reinforcement in honey bees. J. Exp. Anal. Behav. 20:105–109.

Hartling, L. K., and R. C. Plowright. 1979. Foraging by bumble bees on patches of artificial flowers: a laboratory study. Can. J. Zool. 57:1866–1870.

Keasar, T. 2000. The spatial distribution of nonrewarding artificial flowers affects pollinator attraction. Anim. Behav. 60:639–646.

Leadbeater, E., and L. Chittka. 2008. Social transmission of nectar-robbing behaviour in bumble-bees. Proc. Biol. Sci. 275:1669–1674.

Lihoreau, M., L. Chittka, and N. E. Raine. 2010. Travel optimization by foraging bumblebees through readjustments of traplines after discovery of new feeding locations. Am. Nat. 176:744–757.

Lynn, S. K., J. Cnaani, and D. R. Papaj. 2005. Peak shift discrimination learning as a mechanism of signal evolution. Evolution 59:1300–1305.

Ohashi, K., D. D’Souza, and J. D. Thomson. 2010. An automated system for tracking and identifying individual nectar foragers at multiple feeders. Behav. Ecol. Sociobiol. 64:891–897.

Sokolowski, M. B., and C. I. Abramson. 2010. From foraging to operant conditioning: a new computer-controlled Skinner box to study free-flying nectar gathering behavior in bees. J. Neurosci. Methods 188:235–242.

Winter, Y., and K. P. Stich. 2005. Foraging in a complex naturalistic environment: capacity of spatial working memory in flower bats. J. Exp. Biol. 208:539–548.

Supporting Information

Additional Supporting Information may be found online in the supporting information tab for this article:

Figure S1. An example of visitation data produced by our software, viewed in Microsoft Excel.

Figure S2. Circuit diagram of a single flower.

Figure S3. Circuit diagram of the connector cards of the control unit.

Figure S4. Use of the system.
**Figure S5.** Movement range of the servo arm: in refill position (left) and in normal feeding position (right).

**Figure S6.** Flower with the top cover removed.

**Figure S7.** Control unit.

**Figure S8.** Examples of individual flowers (with top cover in place) we used (Lämsä et al., unpublished).

**Appendix S1.** ZIP package file containing control software, its source code, firmware’s sources and user instructions.