Arduino Dog (or Cat) Treat Dispenser

https://www.sciencebuddies.org/science-fair-projects/project-ideas/Elec_p106/electricity-electronics/automatic-dog-treat-dispenser
(https://www.sciencebuddies.org/science-fair-projects/project-ideas/Elec_p106/electricity-electronics/automatic-dog-treat-dispenser)

Procedure PDF Date: 2024-04-17

Experimental Procedure

Note: This engineering project is best described by the **engineering design process,** as opposed to the **scientific method.** You might want to ask your teacher whether it's acceptable to follow the engineering design process for your project before you begin. You can learn more about the engineering design process in the Science Buddies Engineering Design Process Guide (http://www.sciencebuddies.org/science-fair-projects/engineering-design-process/engineering-design-process-steps).

1. Build your dispenser. Figures 2 through 4 show several different views of the device. You can choose to make your dispenser using craft supplies or 3D-printed parts if you have access to a 3D printer. Your assembly procedure may vary depending on the materials you use. Please note that this dispenser is meant for demonstration purposes or to be placed on a counter or shelf where your pet cannot reach it. Even if you add a lid to the dispenser, many pets may chew through it to get to the treats. You would need to build a much sturdier dispenser using stronger materials (like hard plastic) if you want to leave your dispenser on the floor.
    a. Cut a hole in a piece of cardboard so the top of the servo motor fits through.
    b. Glue the servo to the bottom of the cardboard, so the top sticks through the hole.
    c. Build legs to support the four corners of the cardboard.
    d. Place a circular wall on top of the cardboard, centered around the servo. A cardboard tube from the inside of a roll of duct or masking tape works well.
    e. Make dispenser compartments by gluing popsicle sticks together. Use a protractor if you want to measure exact angles. You will need to decide how many compartments you want, which in turn will determine how far the servo should rotate.
    f. Attach the servo horn to the motor (do not use a screw, just push it on) and carefully glue the popsicle sticks to the servo horn. Be careful not to use too much glue, you may need to pull the servo horn off later to make adjustments.
    g. Cut a pie-shaped wedge in the cardboard, the same size and shape as one of the dispenser slots.
    h. Build a ramp under the hole so the treats will slide down into a tray (omit the tray if you want the treats to fall off a counter or shelf onto the floor).

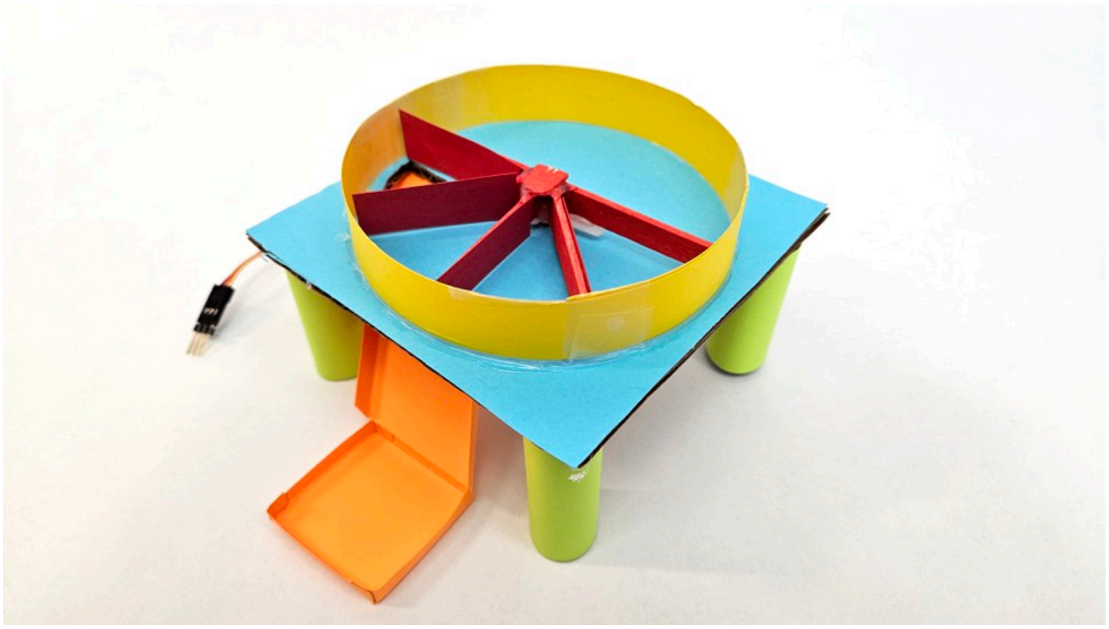

Image Credit: Ben Finio (http://www.sciencebuddies.org/image-credit?id=20310)

Figure 2. Top view of the mechanism.

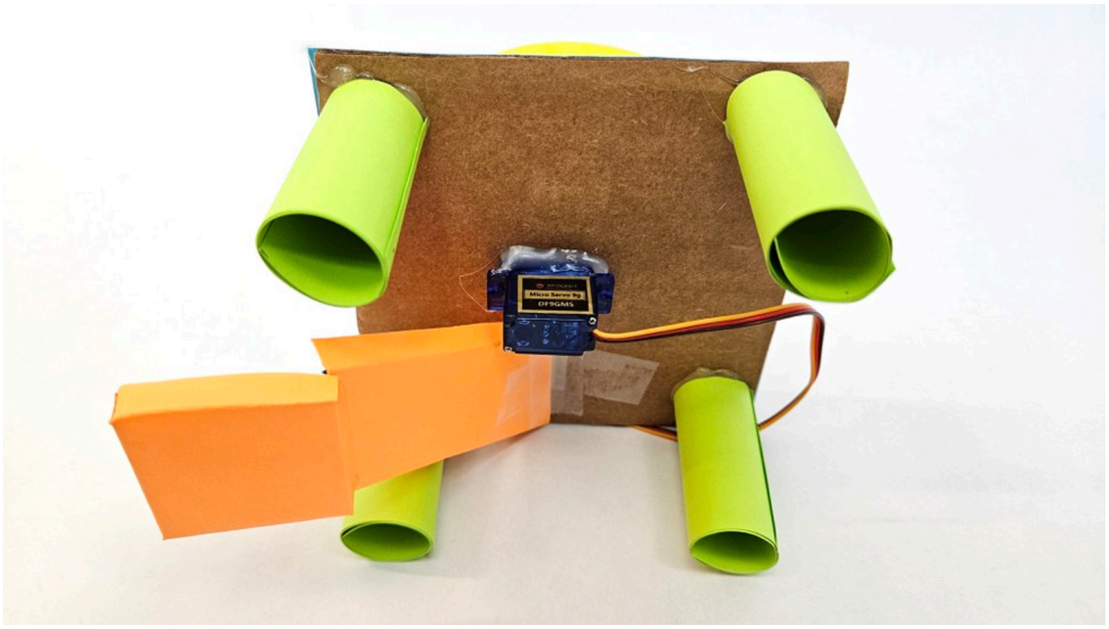

Image Credit: Ben Finio (http://www.sciencebuddies.org/image-credit?id=20311)

Figure 3. Bottom view showing the servo glued to the underside of the cardboard.

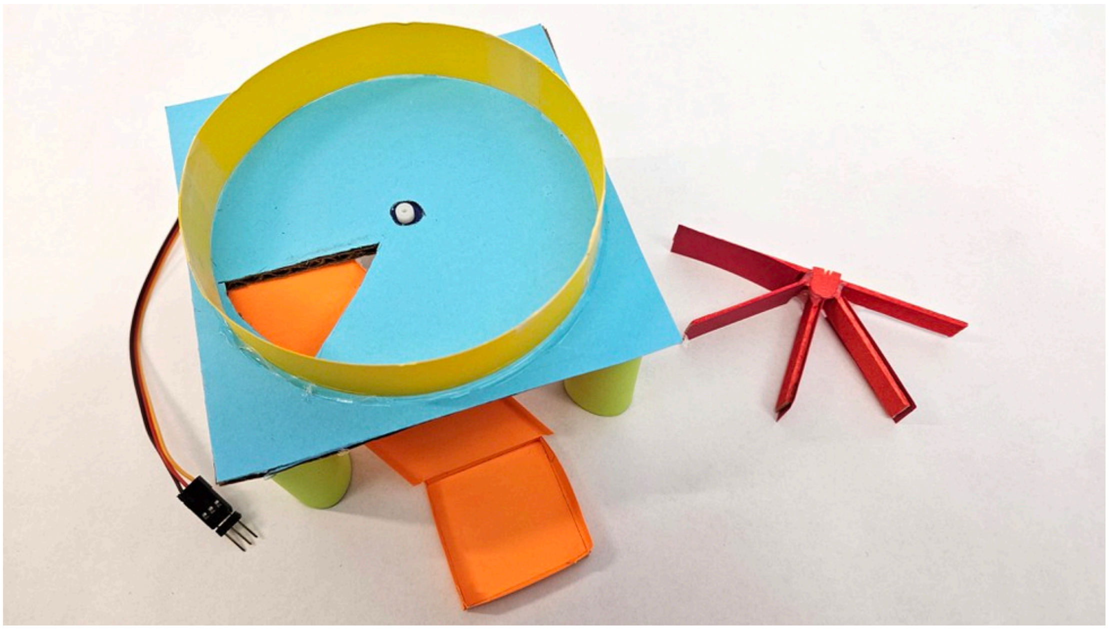

Image Credit: Ben Finio (http://www.sciencebuddies.org/image-credit?id=20312)

Figure 4. Top view with the servo horn removed.

2. Assemble the circuit as shown in Figures 5 and 6. You will need to know how to use a breadboard (http://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard). Make sure your Arduino is *not* plugged into USB power as you build the circuit. That will give you a chance to double-check all of your wiring before powering the Arduino on. Note that this diagram assumes you are using an Arduino UNO R4 with a built-in real-time clock. If you are using an external RTC, you will either need a bigger breadboard, or you will have to remove some components (like the buzzer) from the breadboard to make room for the RTC. See our external RTC tutorial (http://www.sciencebuddies.org/science-fair-projects/references/how-to-use-an-arduino) for instructions on wiring an external RTC. You will need to modify the circuit if you are using other sensors.
   a. LCD connections. Note that pin labels may vary depending on where you bought your board. Pin numbers go from left to right when viewing the screen horizontally, but from *bottom to top* in Figure 6 since the screen is rotated.
      i. Pin 1 (VSS or GND) to GND
      ii. Pin 2 (VDD or VCC) to 5V
      iii. Pin 3 (V0) to potentiometer center pin
      iv. Pin 4 (RS) to Arduino pin 12
      v. Pin 5 (RW) to GND
      vi. Pin 6 (E) to Arduino pin 11
      vii. Pin 7 (D0) - not used
      viii. Pin 8 (D1) - not used
      ix. Pin 9 (D2) - not used
      x. Pin 10 (D3) - not used
      xi. Pin 11 (D4) to Arduino pin 5
      xii. Pin 12 (D5) to Arduino pin 4
      xiii. Pin 13 (D6) to Arduino pin 3
      xiv. Pin 14 (D7) to Arduino pin 2

    xv. Pin 15 (LED or A) to current limiting resistor to 5V
    xvi. Pin 16 (LED or K) to GND

b. Potentiometer
    i. One outer pin to 5V
    ii. Center pin to pin 3 (V0) on LCD screen
    iii. Other outer pin to GND

c. LED
    i. Positive (longer) leg to Arduino pin 7
    ii. Negative (shorter) leg to current limiting resistor to GND

d. Buzzer (optional)
    i. Positive pin to Arduino pin 8
    ii. Negative pin to GND

e. Pushbutton (optional)
    i. One side to 5V
    ii. Other side to GND through 10 kΩ pull-down resistor *and* to Arduino pin 9

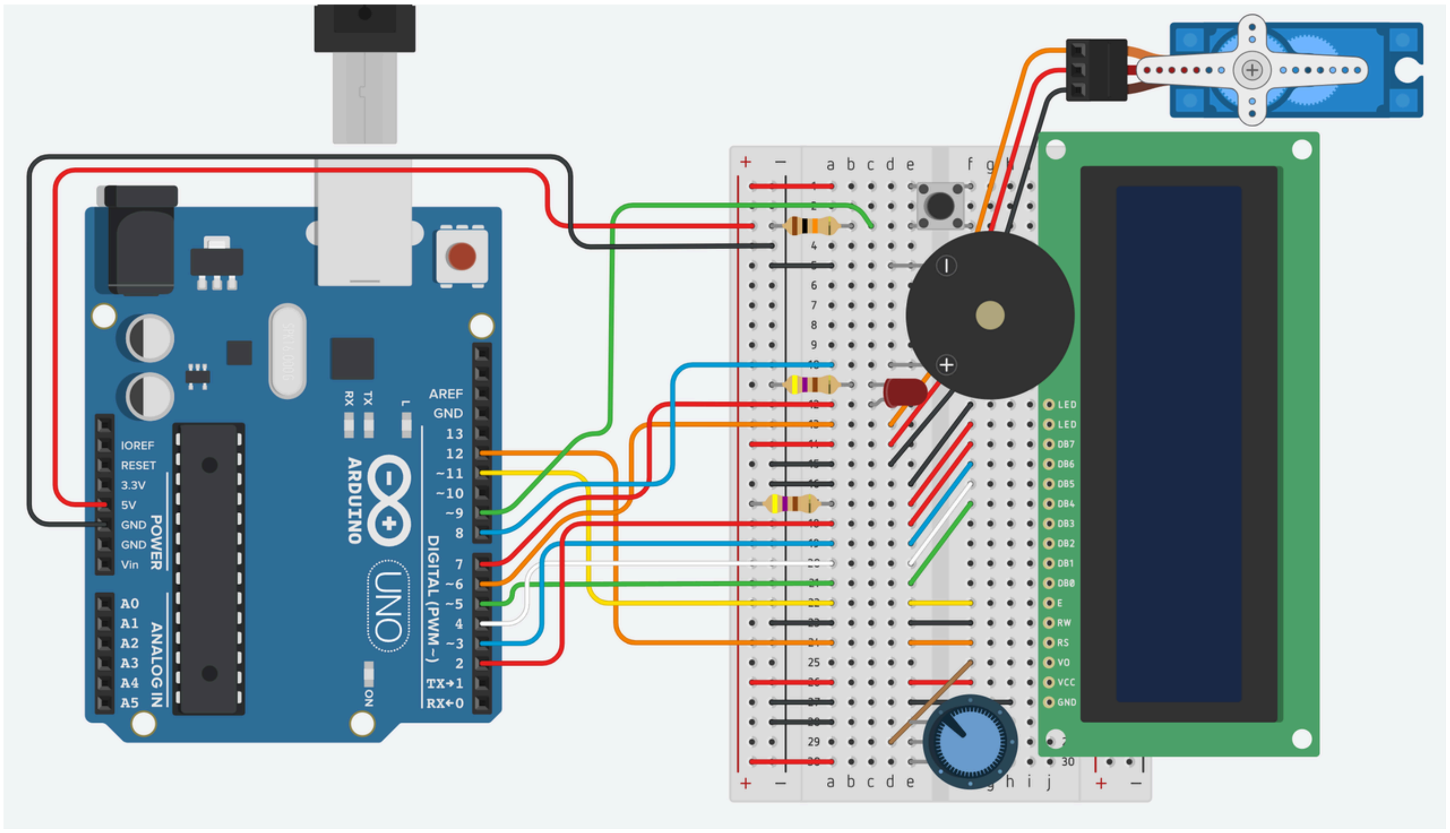

Image Credit: Ben Finio (http://www.sciencebuddies.org/image-credit?id=20315)

Figure 6. Breadboard diagram. Click here to view the circuit in Tinkercad where you can zoom (https://www.tinkercad.com/things/bEedbp22SzJ-automatic-pill-dispenser). Note that the Tinkercad circuit does not contain code so nothing will happen if you run the simulation.

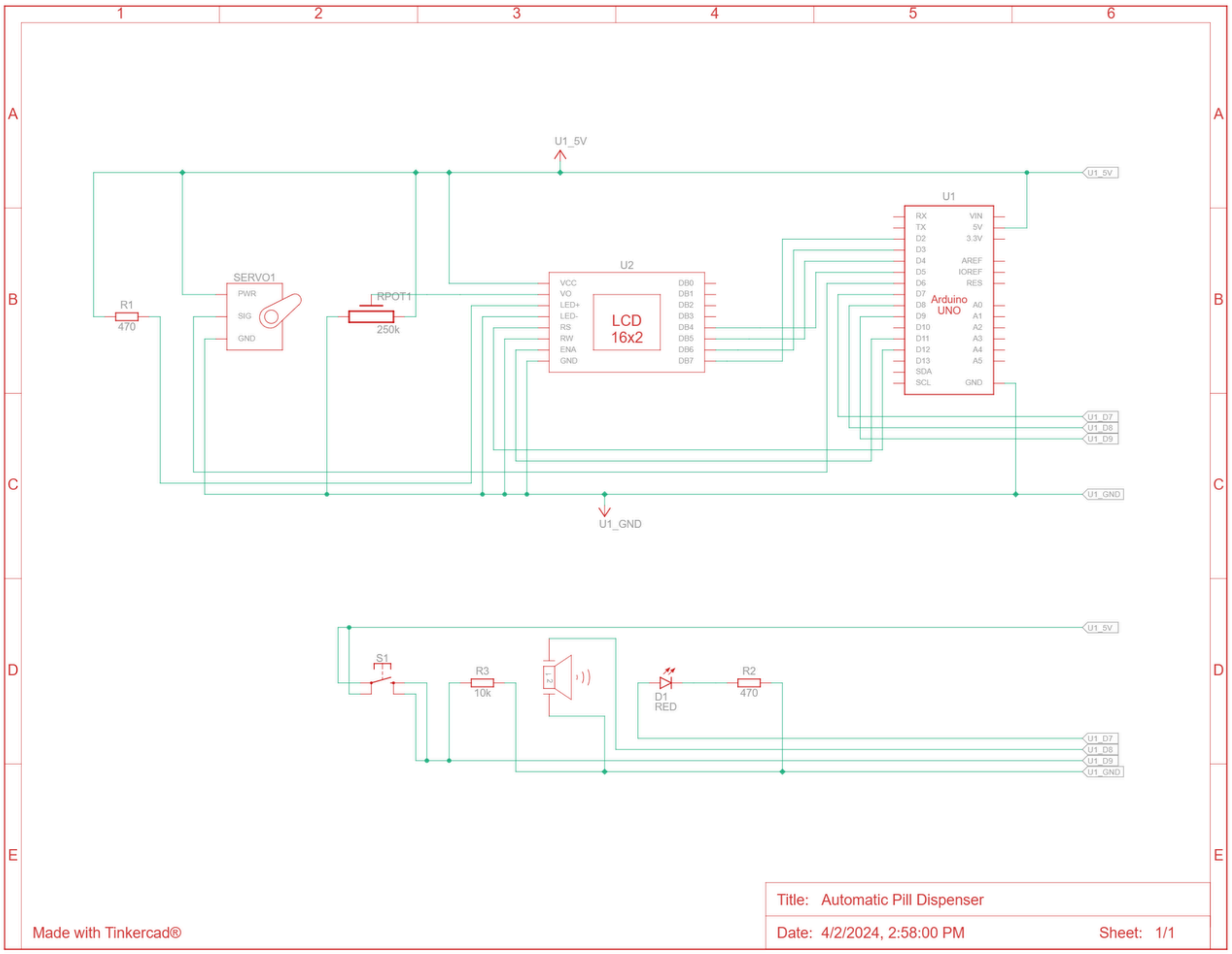

Image Credit: Ben Finio (http://www.sciencebuddies.org/image-credit?id=20314)

Figure 7. Circuit schematic.

3. Download the treat dispenser example code (http://www.sciencebuddies.org/cdn/Files/20351/2/automatic_dog_treat_dispenser.ino). **Read through the commented code so you understand how it works.**
    a. This code is written for the internal RTC on an Arduino UNO R4 Minima. If you are using an external RTC module, you will need to modify the code. We have example code for an external RTC here (http://www.sciencebuddies.org/science-fair-projects/references/how-to-use-an-arduino).
    b. This code is written for a dispenser with four compartments. It automatically rotates the servo 45° once every 15 seconds. This works well for testing the device, but you will need to change the code to make it dispense treats at different times of day.
    c. You will need to change the code if you want to activate the motor based on another sensor instead of (or in addition to) the RTC.
4. Upload the code to your Arduino and test it. Watch the motion of the dispenser. You may need to pop off the servo horn (this is why we said not to attach it with a screw earlier) and adjust it so the compartments align with the hole in the cardboard.
5. If your device does not work as expected (the servo does not rotate, the LCD screen does not display correctly, etc.) you will need to troubleshoot or debug your circuit (http://www.sciencebuddies.org/science-fair-projects/references/how-to-use-an-arduino). This is a complicated circuit with a lot of parts. It is easy to misplace a wire on the breadboard, which can prevent one or more parts of the circuit from working properly. You will need to carefully double and triple-check all of your connections. It also helps to have someone else double-check your wiring, just like you might ask someone else to proofread your writing. Sometimes it can be hard to spot your own mistakes. If you get stuck or frustrated, take a break from working on your circuit and come back to it later. You might notice something that you did not notice before.
6. Load some treats into your dispenser compartments and test it again. Make sure there are no issues with your ramp or tray, such as the treats getting stuck or sliding down too fast and bouncing out of the tray. Make adjustments to your ramp and tray if needed.
7. Once you have the device working with the example code, it is time to customize your code. The example code dispenses treats once every 15 seconds, which works well when testing or for a demonstration at a science fair. There are many things you can change and customize in the code:
    a. How often it dispenses treats. The example code does this once every 15 seconds, but for real-life use you will need to set different times of day. You can do this by adjusting the condition(s) in the `if` statement that detects the time.
    b. How far the servo rotates. The example code rotates the servo 45° at a time, which works for four compartments. You will need to change the `angleIncrement` variable if you have a different number of compartments.
    c. The messages displayed on the LCD screen (useful for debugging or to display the time for a human, assuming your pet cannot read!).
    d. Whether you use the `flashLED` alarm function to alert your pet that a treat has been dispensed using a buzzer and/or LED.
8. Once you have completed testing your device and customizing the code, try some real-world testing if possible.
    a. Remember to put your dispenser up high on a counter or shelf where your pet cannot reach it, so it will drop the treats onto the floor.

b. If you have any sort of motion-activated camera (home security camera, pet cam, trail cam, etc.), set it up with a good view of your treat dispenser before you leave for the day.
c. Review the footage when you get home. Did your treat dispenser work as intended? Did it help give your dog something to do during the day? Does your dog learn to expect the treats (for example, do they start sitting near the dispenser waiting for the next one)?
d. Based on your testing, what changes could you make to your treat dispenser to improve it?

Copyright © 2002-2024 Science Buddies. All rights reserved. Reproduction of material from this website without written permission is strictly prohibited.
Use of this site constitutes acceptance of our Terms and Conditions of Fair Use (https://www.sciencebuddies.org/about/terms-and-conditions-of-fair-use).
Privacy Policy (https://www.sciencebuddies.org/about/privacy-policy)