Arduino Nanocourse Day 2 Project

Behavior Box Circuit Project

Many neurobiology experiments involve interacting with a behaving animal and monitoring its actions. In this project you will get a feel for the sensors and actuators relevant to a behavioral setup by building a ‘behavior box’. This is a device used to train mice on operant conditioning tasks.

Goal

Construct a rudimentary behavioral training setup that:

1. Triggers the start of a trial
2. Checks for some response
3. Delivers a reward

Constraints

Each trial should last 30 seconds (starting after the trigger is presented). If the mouse responds within three seconds after the trigger, a reward should be given.

Components

Triggering a trial

To trigger the start of a trial, use a piezo buzzer to produce an audible tone. You can connect one lead of the buzzer directly to one of the digital pins and connect the other lead to ground. As an option, you can use a series resistor to control volume. See the tone function for more information: https://www.arduino.cc/en/Reference/Tone

Detecting a response

There are multiple ways to detect a response. Every sensor will have benefits and drawbacks. We are providing two options. Please select only one.

Photointerrupter

> By selecting this component you’ve decided to detect a ‘nosepoke’ by using a photointerrupter to detect when the mouse pokes its nose into a small gap in the sensor. Inside the sensor is an LED paired with a photo-transistor. The LED provides infrared illumination, while the photo-transistor responds to the light and amplifies the photocurrent. The output is a digital “high” when not occluded and “low” when the light

beam is blocked. Please consult the datasheet for connection details (Sparkfun part #GP1A57HRJ00F).

Temperature sensor

By selecting this component you've decided to detect the presence of the mouse by looking for a temperature change. In the instrument, this sensor might be attached to a metal plate on which the mouse stands,but for this course you'll only be using the sensor. The sensor you will use is TMP36; it outputs an analog voltage that is linearly proportional to temperature within its range (10mV per degree C). Please note that the TMP36 output will always range from 100mV to 2V and consult the datasheet for additional details. Also, please note that changes in temperature occur slowly, and it may take several seconds for the sensor to register a response.

Administering a reward

For a specific experiment, you may want to deliver either a liquid or solid reward. This constraint would in part determine what actuator you would use to administer the reward. We are providing you with two options below. Please pick only one.

Solenoid

A solenoid is an electromechanical transducer that pushes a plunger when it's energized. In a real experiment the solenoid would be coupled with a valve that delivers a liquid reward such as water. The solenoid actuator that we provide is a power hungry device, and as you saw in lecture, an individual Arduino pin can not provide enough current on its own. To overcome this, you'll have to use a power transistor as an amplifier/switch to handle the higher current. (Solenoid part # ZH0-0420S-05A4.5, Transistor part #PSMN022-30PL)

Servo

Servo motors are rotary actuators that use feedback to precisely control their position. In a real experiment the servo might be attached to a container of food pellets so that when the servo arm moves to a set location and then returns, a reward is delivered. The servo we provide can rotate through a 180 degree arc. The servo has 3 terminals: Brown or Black wire should be connected to ground, Red to +5V, Orange or Yellow to the control signal. We recommend using the Arduino library <Servo.h> to control the servo. (Note that this library will control servos only on pins 9 and 10.) More information and examples can be found on the Arduino website.
https://www.arduino.cc/en/Reference/Servo

Control Logic

This project introduces several new ideas. You will need to time your events while your code is doing something else. You can utilize the built-in function `millis()`, which keeps track of time in milliseconds since the arduino board started.

As with the previous project, it's best to build up your functionality in parts. Make sure to debug individual components before moving on. You should make use of the serial communication port to understand what the microcontroller is doing. Your general program flow might be something like this:

```
void setup(){
…
}

void loop() {

    tone (buzzerPin, pitch, duration_ms);
    startTime = millis(); //current time

    while (within opportunity window){
      if (sensor is tripped) {
        deliver reward
      }
    }
}
```

Bonus

If you have additional time or want experience with more components. Repeat the project using the other options for the sensor and actuator. Do you have to modify the control code? How might you write control code that would work for any combination of sensors and actuators?