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 by sounding a tone
2. Checks for some response
3. Delivers a reward

Each trial should last 30 seconds. Begin by presenting a tone. If the animal responds within three seconds after this trigger, a reward should be given.

Components

Wire color convention

To make debugging easier, we recommend using a consistent wire color convention. This is good practice not just for this class, but for future electronics projects as well.

Wire Color	***Function***
RED	Carries Power (eg., 5 V, 3.3 V)
BLACK or **GREEN**	Ground (0V)
Other	All other signals

In the kits, we provide an assortment of wires in different colors. You may not have exactly enough to strictly maintain the color convention above. Do the best you can.

Triggering a trial

To trigger the start of a trial, use a piezo buzzer to produce an audible tone. The buzzer is a small plastic can with two leads and a small opening at the top. Plug the buzzer into the prototyping board. Connect one lead to one of the Arduino digital pins. Connect the other lead to ground. As an option, you can use a resistor in series with the buzzer to limit the volume. See the *tone* function for more

information on how to use the buzzer : https://www.arduino.cc/en/Reference/Tone

Detecting a response

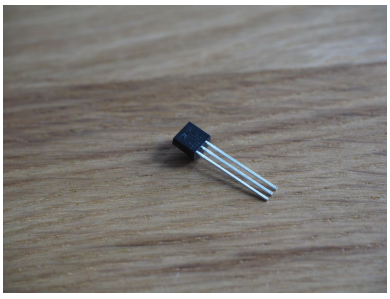
There are multiple ways to detect a response. We provide you with two options. Please select only one.

Photointerrupter

A ‘nosepoke’ can be registered by using a photointerrupter to sense when the mouse pokes its nose into a small gap in the sensor. In one half of the sensor is an LED, and in the other is 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. The photointerrupter is mounted on a small circuit board which includes the current limiting resistor for the LED. To wire it up, please use a red jumper wire to connect PWR to the 5V pin. Connect GND with a green or black wire to one of the GND pins on the Arduino or to the GND bus on the breadboard. The SIG pin should be connected to whichever digital Arduino pin you’ve specified as the input in setup() The datasheet and connection details can be found here: https://www.sparkfun.com/products/9299 https://www.sparkfun.com/products/9322

Temperature sensor

Alternatively, you can use a temperature sensor to detect the presence of the animal. 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 (https://www.sparkfun.com/products/10988); it outputs an analog voltage that is linearly proportional to temperature within its range (10mV per degree C). Connect the sensor power and ground pins. Connect the output pin to one of the *analog in* pins on the Arduino. Please note that the TMP36 output ranges from 100mV to 2V; consult the datasheet for additional details. Keep in mind that changes in temperature occur slowly, and *it may take several seconds for the sensor to register a response*.



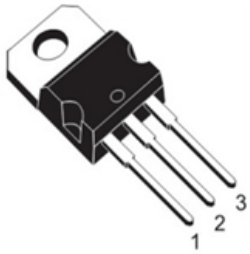
Administering a reward

For a specific experiment, you may want to deliver either a liquid or solid reward. This constraint would, in part, determine what actuator you use to administer the reward. We provide you with a solenoid and a servo motor. Please pick one for reward delivery.

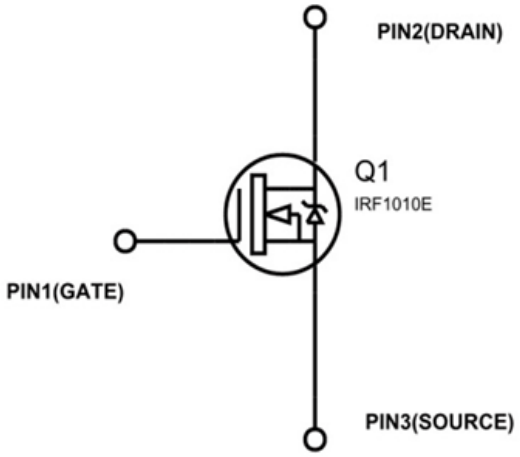
Solenoid

A solenoid is an electromechanical transducer that pushes a plunger when it's energized. In a real experiment, the solenoid might be coupled with a valve that delivers a liquid reward such as water. The solenoid actuator that we provide requires more power than an individual Arduino pin can source. 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)

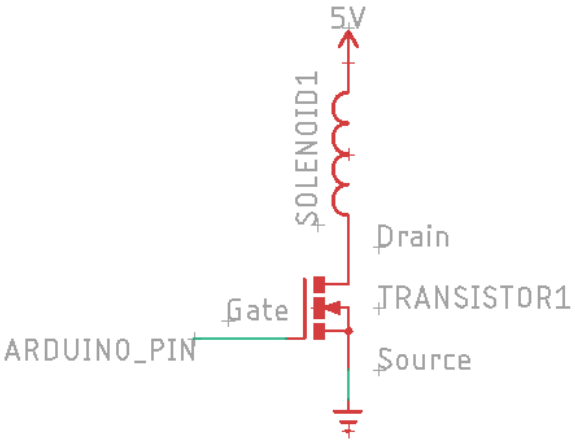
The goal is to have the Arduino control the transistor, which, in turn, switches current through the solenoid. As introduced in lecture, a transistor is a 3-terminal device. The control terminal is called the *gate* and the other terminals are called *drain* and *source*. When the voltage on the *gate* exceeds ~2V, the transistor conducts between its *drain* and *source* terminals. Refer to the diagram below to identify the terminals and



TO-220



corresponding pins and wire up the circuit as shown to the left. If you're having a hard time interpreting the schematic, you can refer to the Fritzing diagram at the end of the assignment for extra guidance. When you write your code, be sure to keep the solenoid on-time to less than 1 second. This reduces the chance of overheating and brown-out. If you're interested to learn more, refer to the bonus section at the end, which has additional details.



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 pellet 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, Yellow or White to the control signal. We recommend using the Arduino library <Servo.h> to control the servo. More information and examples can be found on the Arduino website. The servo comes with a small hardware pack. You can attach the servo arm to the splined output shaft and secure it with a small screw using a Phillips screwdriver.
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 returns the time in milliseconds since the Arduino board started. You can think of it as a wall clock: to time an event, you can check `millis()` before the event begins, and once `millis()` has accumulated the required duration, you can end the event.

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 the partial example below. Please add in the missing parts.

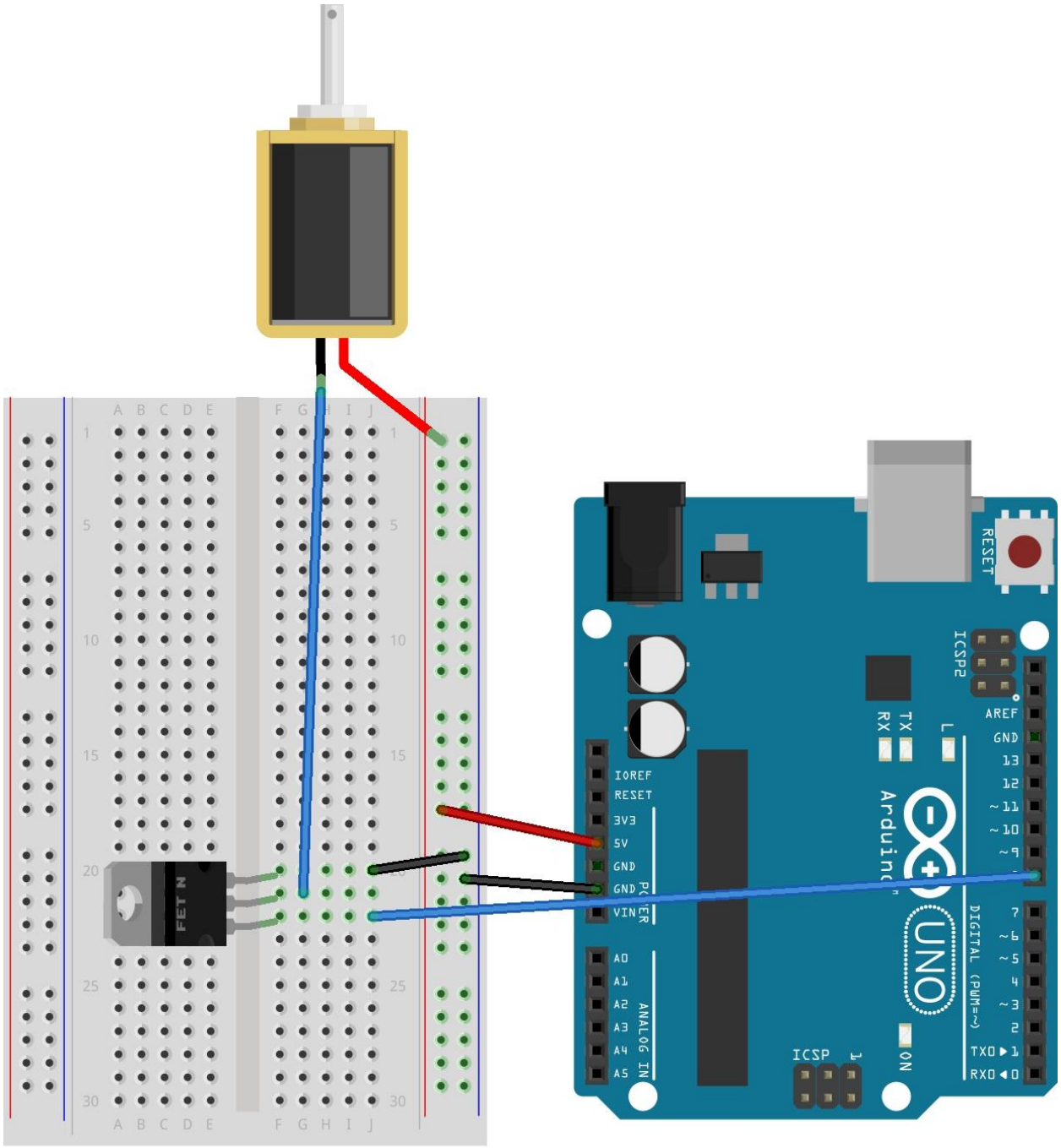
```
unsigned long int trialStartTime;
//declare variables you plan to use
//...
void setup(){
    //specify input and output pins you plan to use below
    //...
    Serial.begin(9600);//Start serial comm
}

void loop() {
    //loop runs once per trial
    //update trial counter
    //...
    tone (buzzerPin, pitch, duration_ms);//start-of-trial tone
    trialStartTime = millis(); //time at start of trial

    //determine what test condition should be within the while loop
    while (within reward window){
      if (sensor is tripped) { //determine how to test if a poke has occurred
        //deliver reward
        //...
        Serial.print("reward delivered at time ");
        Serial.println(millis());
      }
    }
    while (within trial){
      if (sensor is tripped){
      //unrewarded attempt
      Serial.print("unrewarded nose poke ");
      }
    }
}
```

HINT: One strategy to see if a nose poke just occurred, is to look for a state transition. For example, if using the beam break, you can poll the sensor to see if the beam is blocked, but this isn't sufficient to know if the poke just occured or if the beam had been blocked for a while. To resolve this, you would also keep track of the previous state. So if the previous state is "not blocked" and the current state is "blocked" then you know that a transition just occurred.

Solenoid connection diagram.



Bonus

Alternate Components

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?

Aversive Stimulus

Use both actuators at once. When the animal performs a nose poke outside of the reward window, provide a punishment stimulus by using whichever actuator you didn't use previously. (For example, the motor might deliver a food pellet reward, while the solenoid generates an aversive air puff.)

Solenoid Connectivity Options

If you need to use a solenoid in the lab, the schematic below is the one you should use. It features a flyback diode to protect the circuit from voltage spikes that occur when the solenoid switches. It also has a pull-down resistor on the gate of the transistor to ensure that when an explicit control signal is not present, the solenoid remains in the 'off' state.