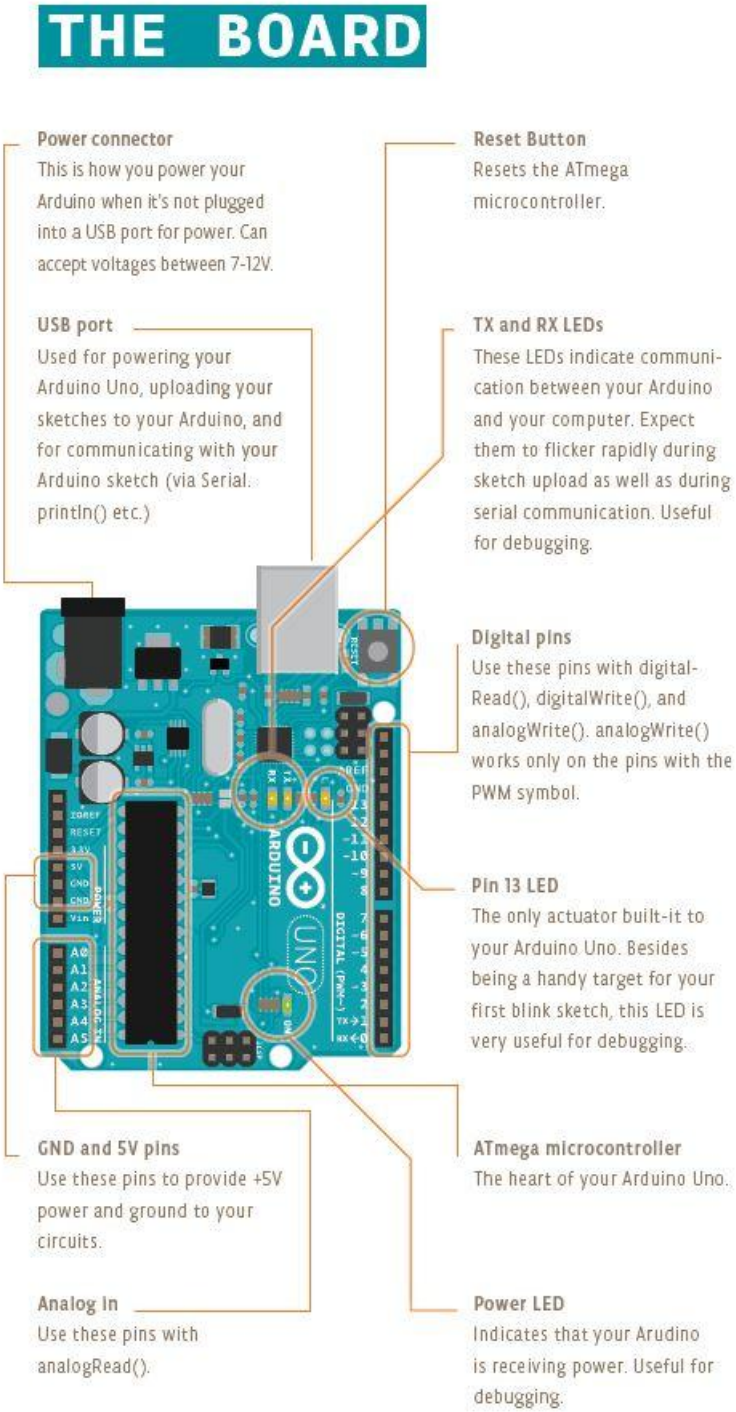
1 General Information



Arduino, natively, supports a language that we call the Arduino Programming Language, or Arduino Language.

This language is based upon the Wiring development platform, which in turn is based upon Processing, which if you are not familiar with, is what p5.js is based upon. It's a long history of projects building upon other projects, in a very Open Source way. The Arduino IDE is based upon the Processing IDE, and the Wiring IDE which builds on top of it.

When we work with Arduino we commonly use the Arduino IDE (Integrated Development Environment), a software available for all the major desktop platforms (macOS, Linux, Windows), which gives us 2 things: a programming editor with integrated libraries support, and a way to easily compile and load our Arduino programs to a board connected to the computer.

The Arduino Programming Language is basically a framework built on top of C++. You can argue that it's not a real programming language in the traditional term, but I think this helps avoiding confusion for beginners.

A program written in the Arduino Programming Language is called sketch. A sketch is normally saved with the .ino extension (from Arduino).

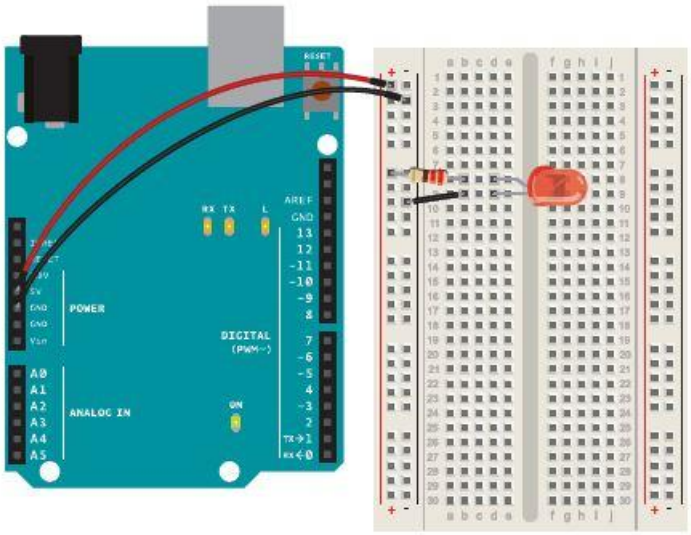
The main difference from "normal" C or C++ is that you wrap all your code into 2 main functions. You can have more than 2, of course, but any Arduino program must provide at least those 2.

One is called setup(), the other is called loop(). The first is called once, when the program starts, the second is repeatedly called while your program is running.

We don't have a main() function like you are used to in C/C++ as the entry point for a program. Once you compile your sketch, the IDE will make sure the end result is a correct C++ program and will basically add the missing glue by preprocessing it.

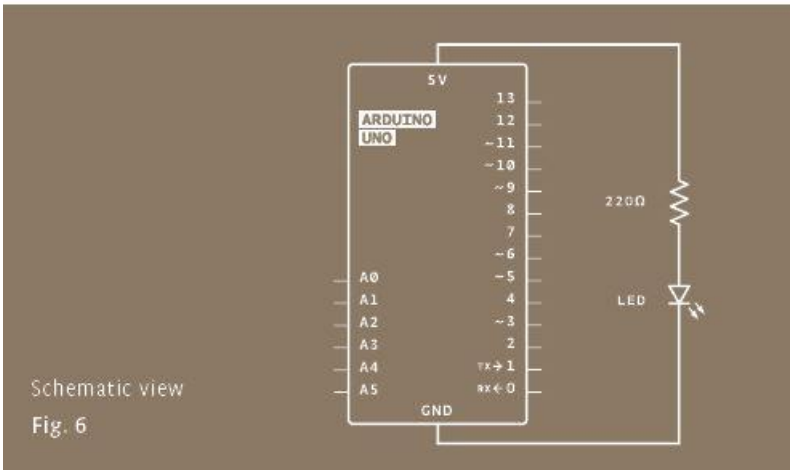
2 Projects

2.1 A SIMPLE CIRCUIT



Circuit illustration.

Fig. 5



Schematic view

Fig. 6

2.2 SPACESHIP INTERFACE

A control panel with a switch and lights that turn on when you press the switch. You can decide whether the lights mean “Engage Hyperdrive” or “Fire the lasers!”. A green LED will be on, until you press a button. When the Arduino gets a signal from the button, the green light will turn off and 2 other lights will start blinking.

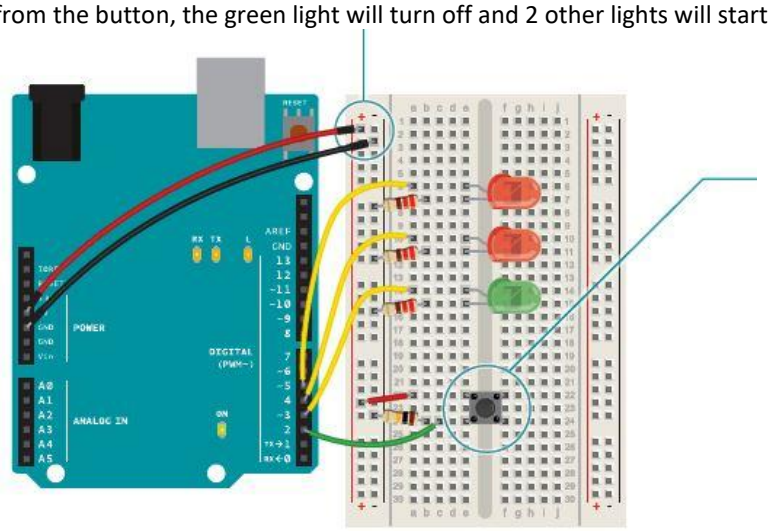


Fig. 1

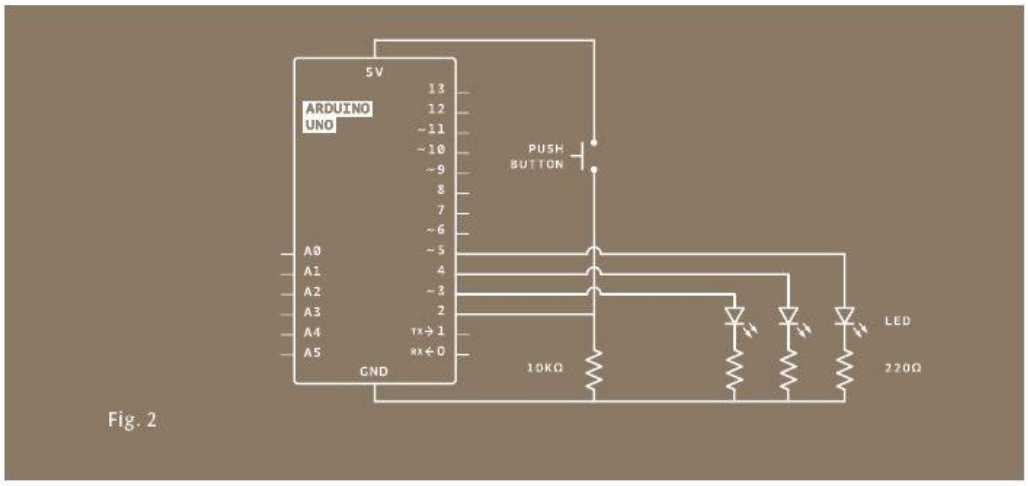
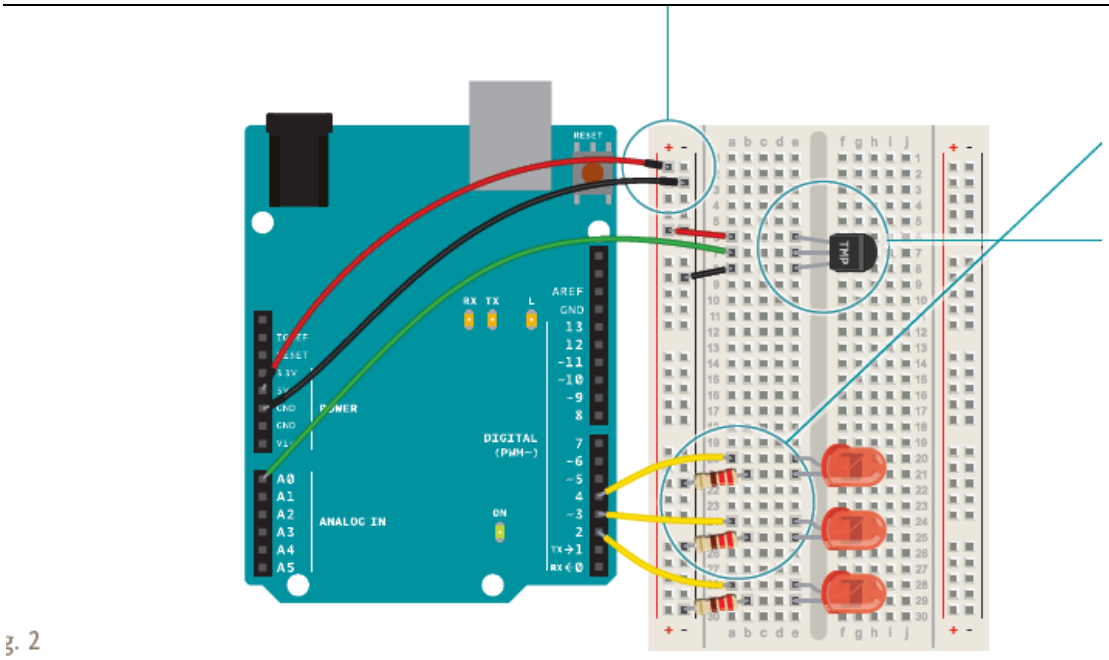


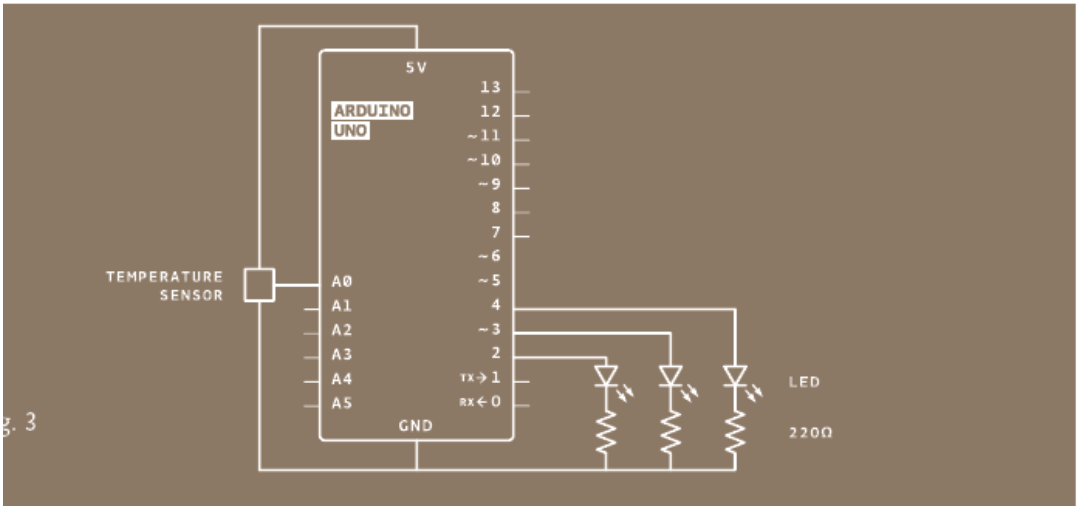
Fig. 2

2.3 LOVE-O-METER

While switches and buttons are great, there’s a lot more to the physical world than on and off. Even though the Arduino is a digital tool, it’s possible for it to get information from analog sensors to measure things like temperature or light. To do this, you’ll take advantage of the Arduino’s built-in Analog-to-Digital Converter (ADC). Analog in pins A0-A5 can report back a value between 0-1023, which maps to a range from 0 volts to 5 volts. You’ll be using a temperature sensor to measure how warm your skin is. This component outputs a changing voltage depending on the temperature it senses. A new led will be light up at every 2 degrees.



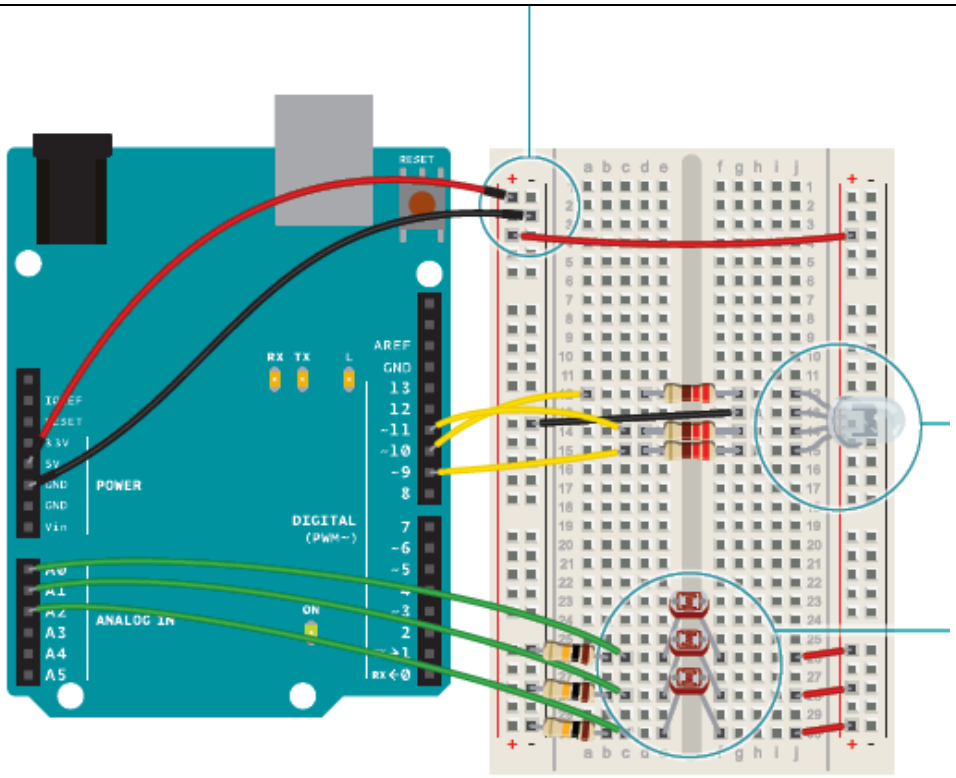
g. 2

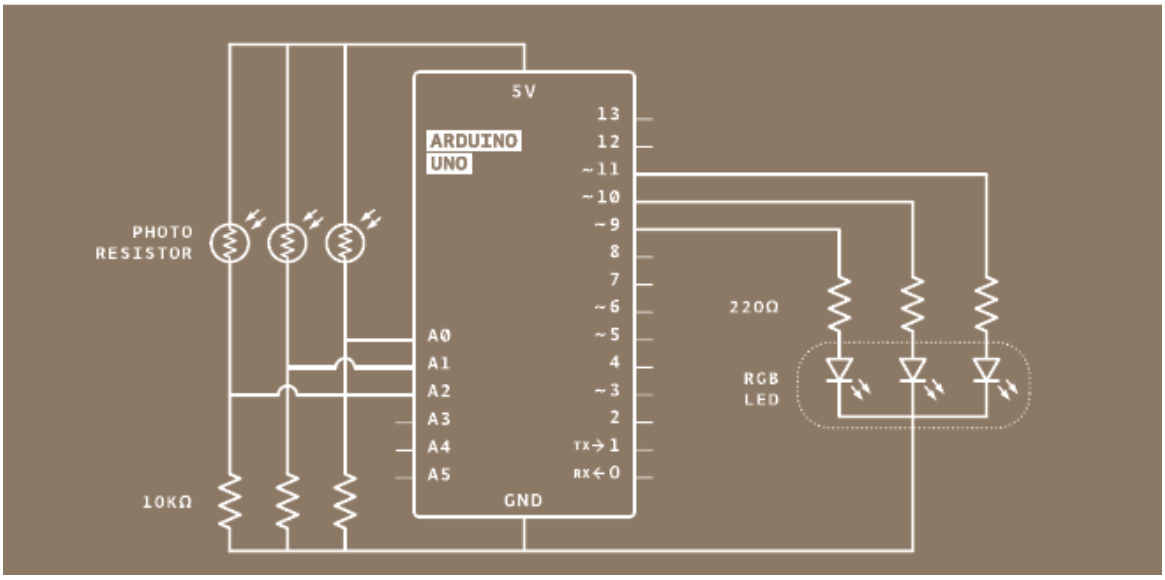


g. 3

2.4 COLOR MIXING LAMP

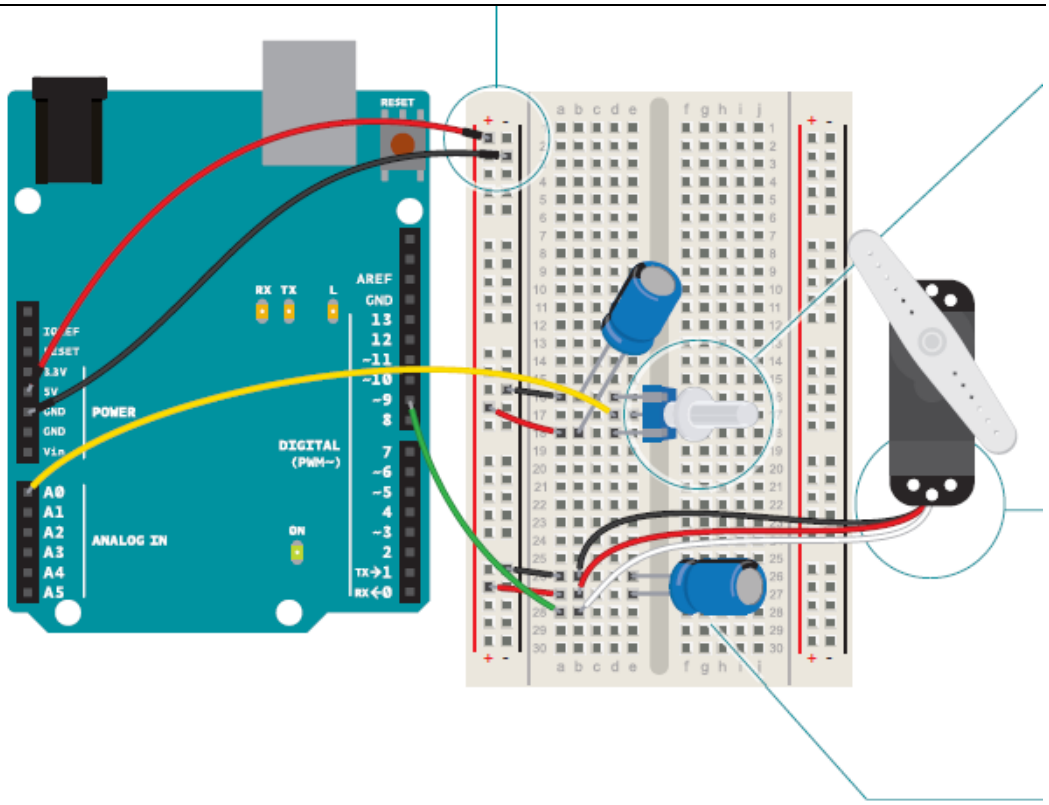
Using a tri-color led and three photoresistors, you'll create a lamp that smoothly changes colors depending on external lighting conditions. The Arduino can't vary the output voltage on its pins, it can only output 5V. Hence you'll need to use a technique called Pulse Width Modulation (PWM) to fade LEDs. PWM rapidly turns the output pin high and low over a fixed period of time. The change happens faster than the human eye can see. It's similar to the way movies work, quickly flashing a number of still images to create the illusion of motion. When you're rapidly turning the pin HIGH and LOW, it's as if you were changing the voltage. The percentage of time a pin is HIGH in a period is called duty cycle. When the pin is HIGH for half of the period and LOW for the other half, the duty cycle is 50%. A lower duty cycle gives you a dimmer LED than a higher duty cycle. For inputs in this project, you'll be using photoresistors (sensors that change their resistance depending on the amount of light that hits them, also known as photocells or light-dependent resistors). If you connect one end of the resistor to your Arduino, you can measure the change in resistance by checking the voltage on the pin.



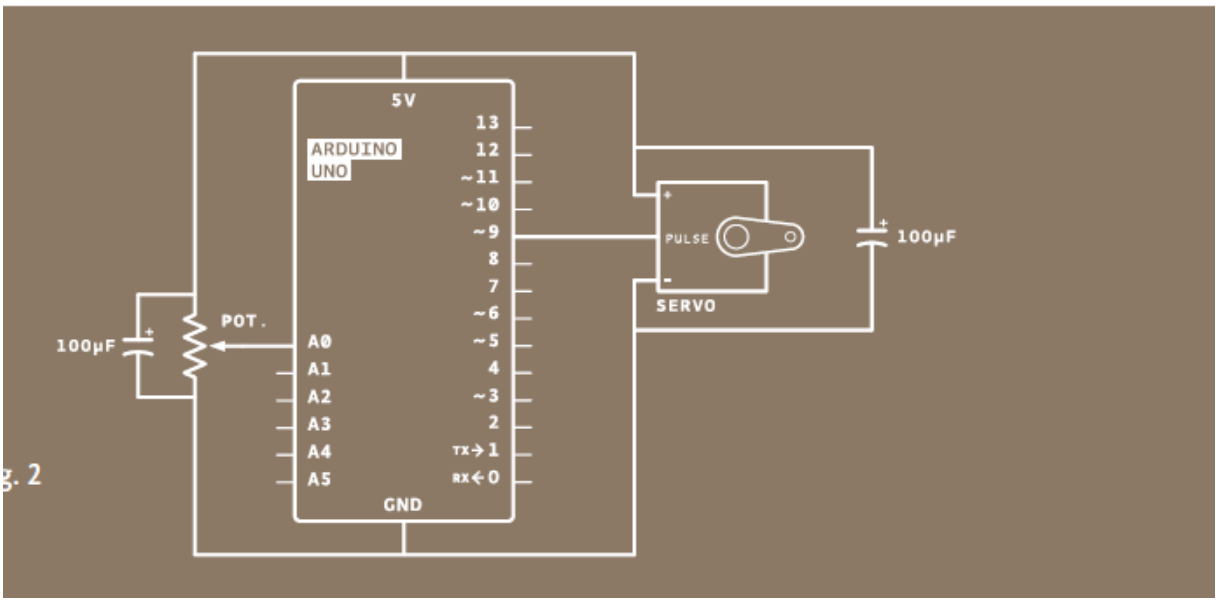


2.5 MOOD CUE

Control a servo motor using a potentiometer.
Servo motors are a special type of motor that don't spin around in a circle, but move to a specific position and stay there until you tell them to move again. Servos usually only rotate 180 degrees (one half of a circle). Because the servo only rotates 180 degrees, and the potentiometer analog input generate a value from 0-1023, you'll need to use a function called map() to change the scale of the values coming from the potentiometer.



g. 1

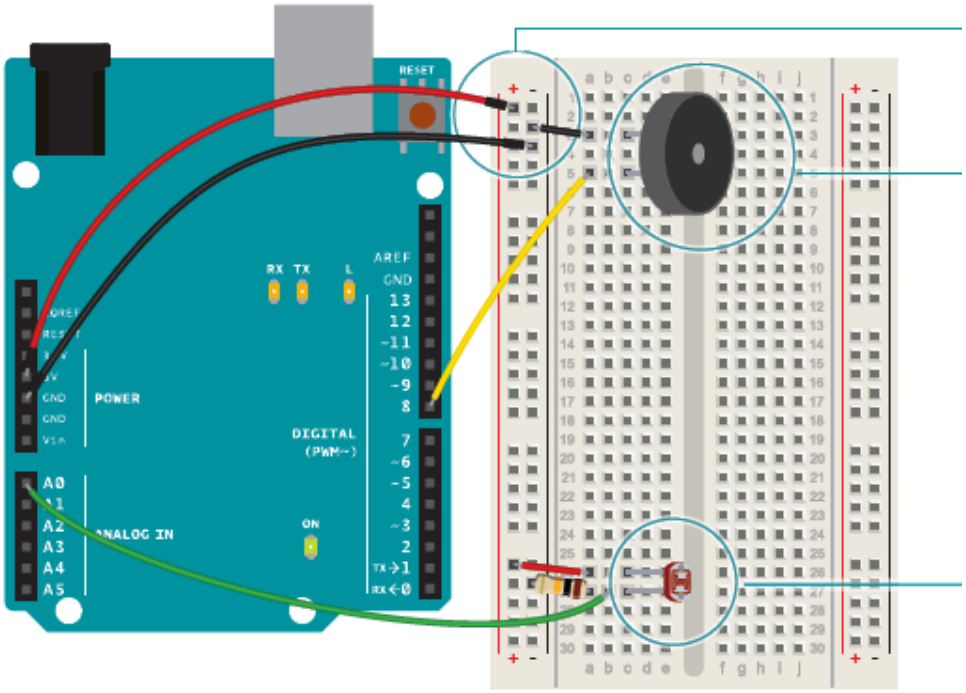


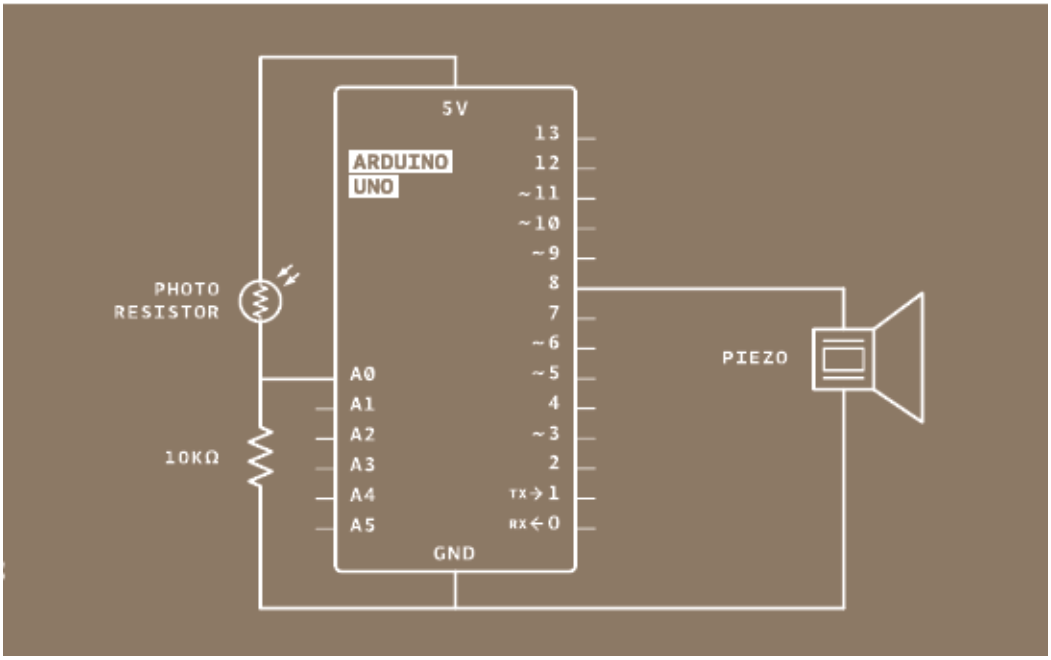
g. 2

2.6 LIGHT THEREMIN

Using a photoresistor and a piezo element, you're going to make a light-based theremin.
A piezo is a small element that vibrates when it receives electricity. When it moves, it displaces air around it, creating sound waves.
A theremin is an instrument that makes sounds based on the movements of a musician's hands around the instrument. You've probably heard one in scary movies. The theremin detects where a performer's hands are in relation to two antennas by reading the capacitive change on the antennas.

These antennas are connected to analog circuitry that create the sound. One antenna controls the frequency of the sound and the other controls volume. Instead of sensing capacitance with the Arduino, you'll be using a photoresistor to detect the amount of light. By moving your hands over the sensor, you'll change the amount of light that falls on the photoresistor's face.

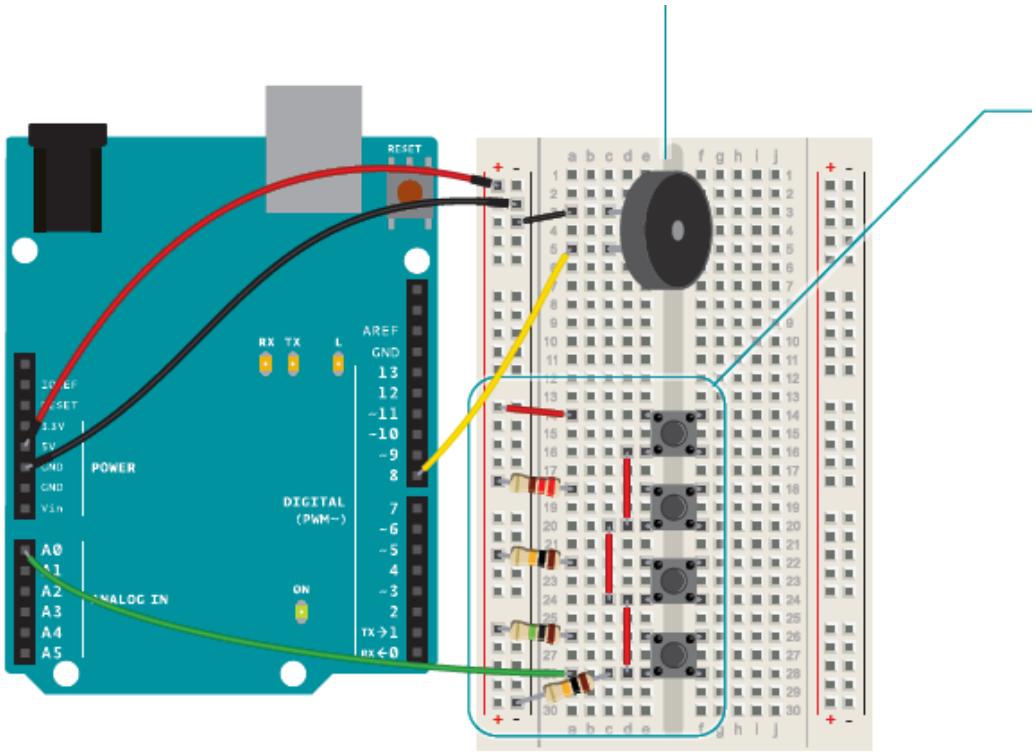


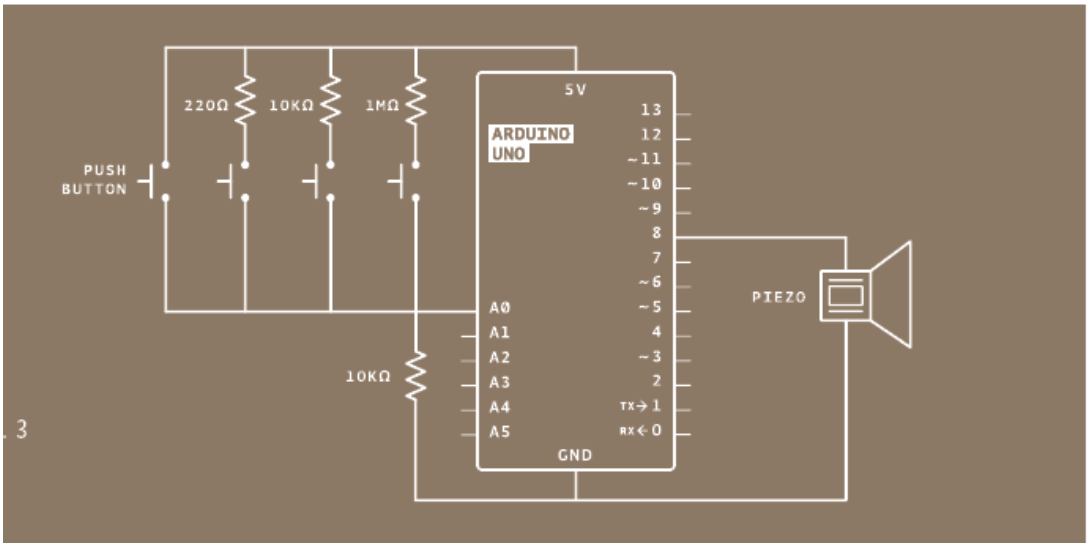


2.7 KEYBOARD INSTRUMENT

With few resistors and buttons we are going to build a small musical keyboard.
This is a way to read a number of switches using the analog input. It's a helpful technique if you find yourself short on digital inputs. You'll hook up a number of switches that are connected in parallel to analog in 0. Most of these will connect to power through a resistor. When you press each button, a

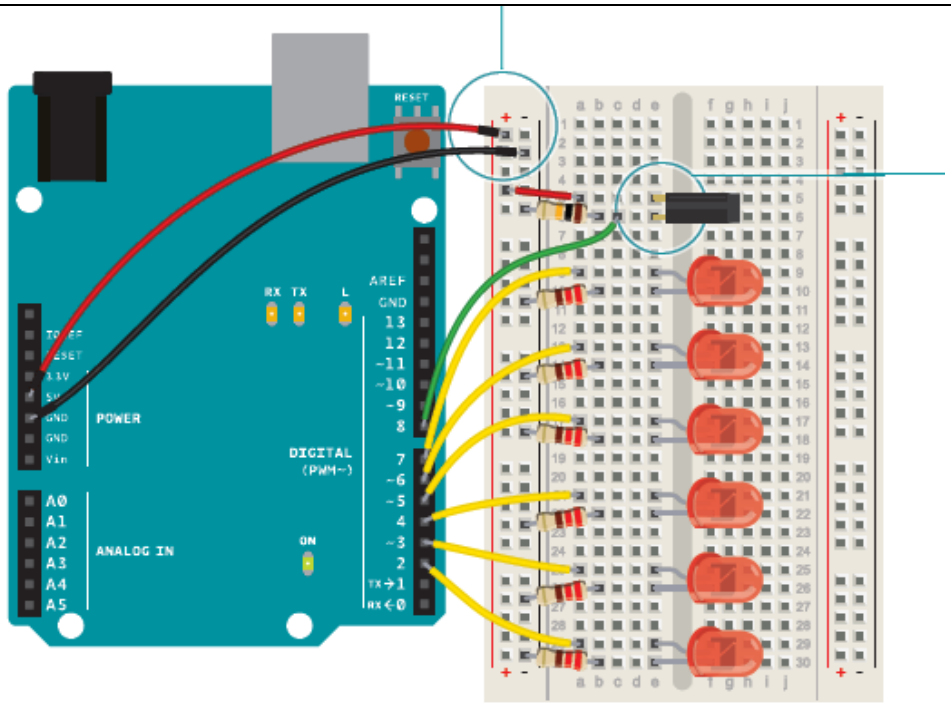
different voltage level will pass to the input pin. If you press two buttons at the same time, you'll get a unique input based on the relationship between the two resistors in parallel.

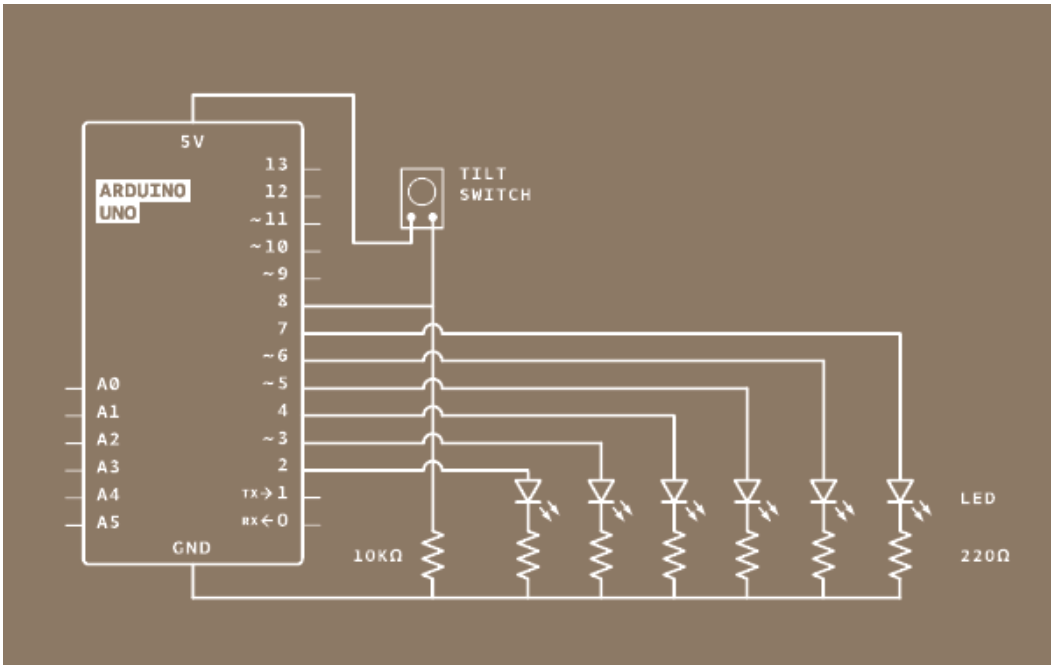




2.8 DIGITAL HOURGLASS

In this project, you'll build a digital hourglass that turns on an led every ten minutes. Know how long you're working on your projects by using the arduino's built-in timer.
When you turn your hourglass over, a tilt switch will change its state, and that will set off another cycle of LEDs turning on. The tilt switch works just like a regular switch in that it is an on/off sensor. You'll use it here as a digital input. What makes tilt switches unique is that they detect orientation. Typically they have a small cavity inside the housing that has a metal ball. When tilted in the proper way, the ball rolls to one side of the cavity and connects the two leads that are in your breadboard, closing the switch. With six LEDs, your hourglass will run for an hour, just as its name implies.





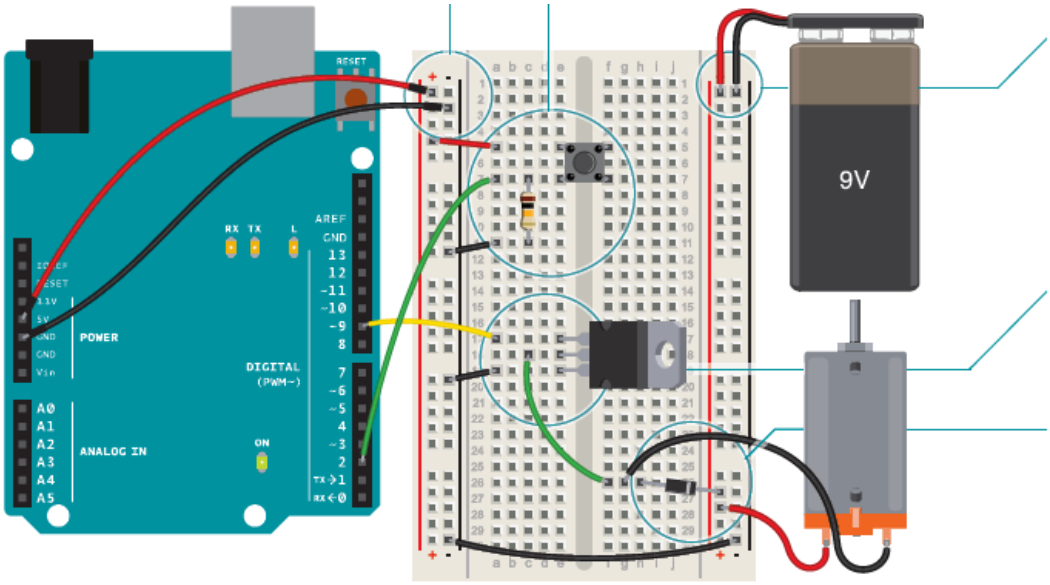
2.9 MOTORIZED PINWHEEL

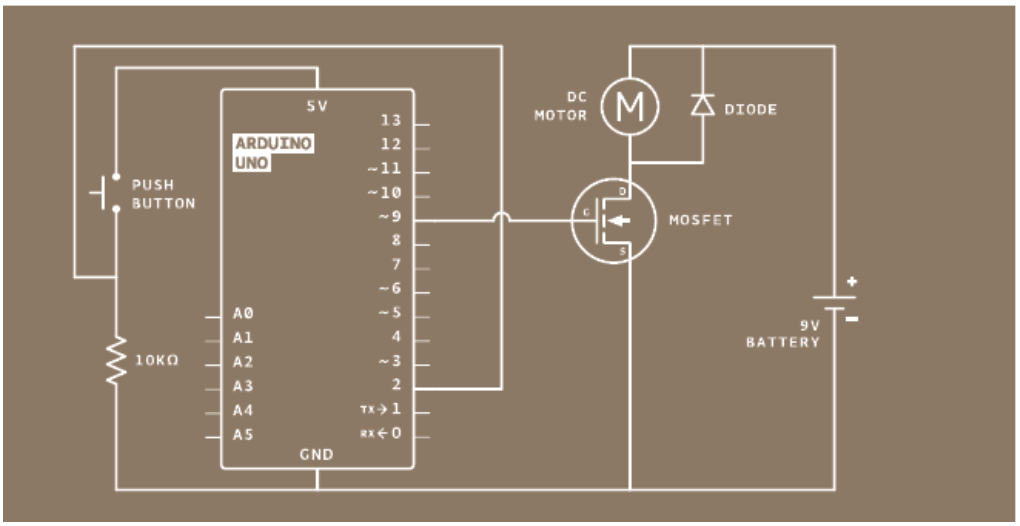
Controlling motors with an Arduino is more complicated than just controlling LEDs for a couple of reasons. First, motors require more current than the Arduino's output pins can supply, and second, motors can generate their own current through a process called induction, which can damage your circuit if you don't plan for it. However, motors make it possible to move physical things, making your projects much more exciting.

Moving things takes a lot of energy. Motors typically require more current than the Arduino can provide. Some motors require a higher voltage as well. To start moving, and when it has a heavy load attached, a motor will draw as much current as it can. The Arduino can only provide 40 milliamps (mA) from its digital pins,much less than what most motors require to work.

Transistors are components that allow you to control high current and high voltage power sources from the low current output of the Arduino. There are many different kinds, but they work on the same principle. You can think of transistors as digital switches. When you provide voltage to one of the transistor's pins, called the gate, it closes the circuit between the other two pins, called the source and drain. This way, you can turn a higher current/voltage motor on and off with your Arduino.

Motors are a type of inductive device. Induction is a process by which a changing electrical current in a wire can generate a changing magnetic field around the wire. When a motor is given electricity, a tightly wound coil inside the housing of copper creates a magnetic field. This field causes the shaft (the part that sticks out of the housing) to spin around.



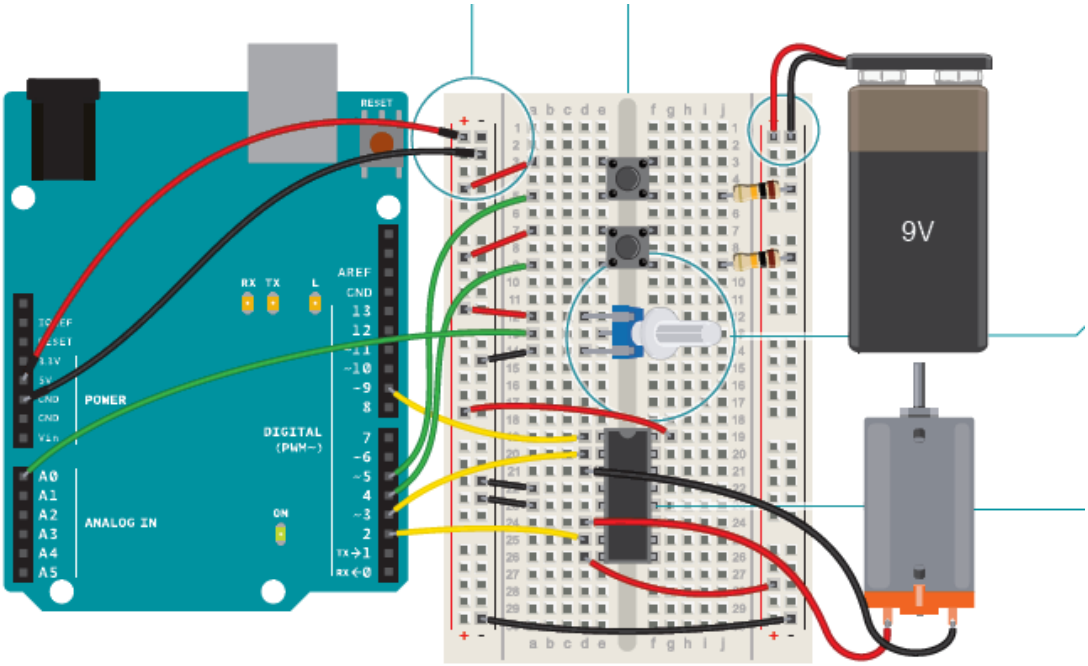


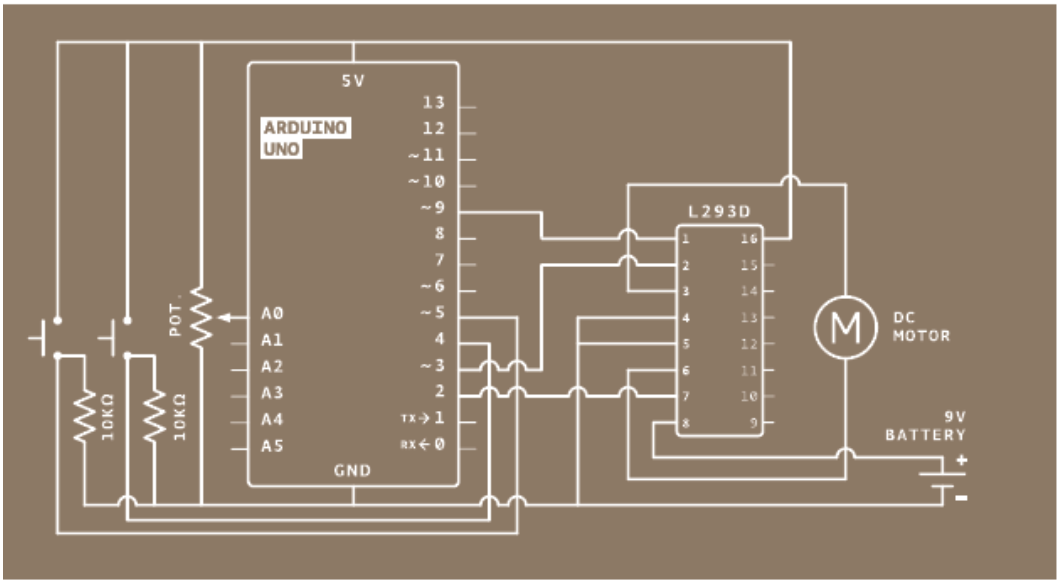
2.10 ZOETROPE

In this project, you'll build your own zoetrope that animates a carnivorous plant. You'll power the motion with a motor. To make this system even more advanced, you'll add a switch that lets you control direction, another to turn it off and on, and a potentiometer to control the speed.

In the Motorized Pinwheel Project you got a motor to spin in one direction. If you were to take power and ground on the motor and flip their orientation, the motor would spin in the opposite direction. It's

not very practical to do that everytime you want to spin something in a different direction, so you'll be using a component called an H-bridge to reverse the polarity of the motor.
H-bridges are a type of component known as integrated circuits (IC). ICs are components that hold large circuits in a tiny package. These can help simplify more complex circuits by placing them in an easily replaceable component. For example, the H-bridge you're using in this example has a number of transistors built in. To build the circuit inside the H-bridge you would probably need another breadboard.



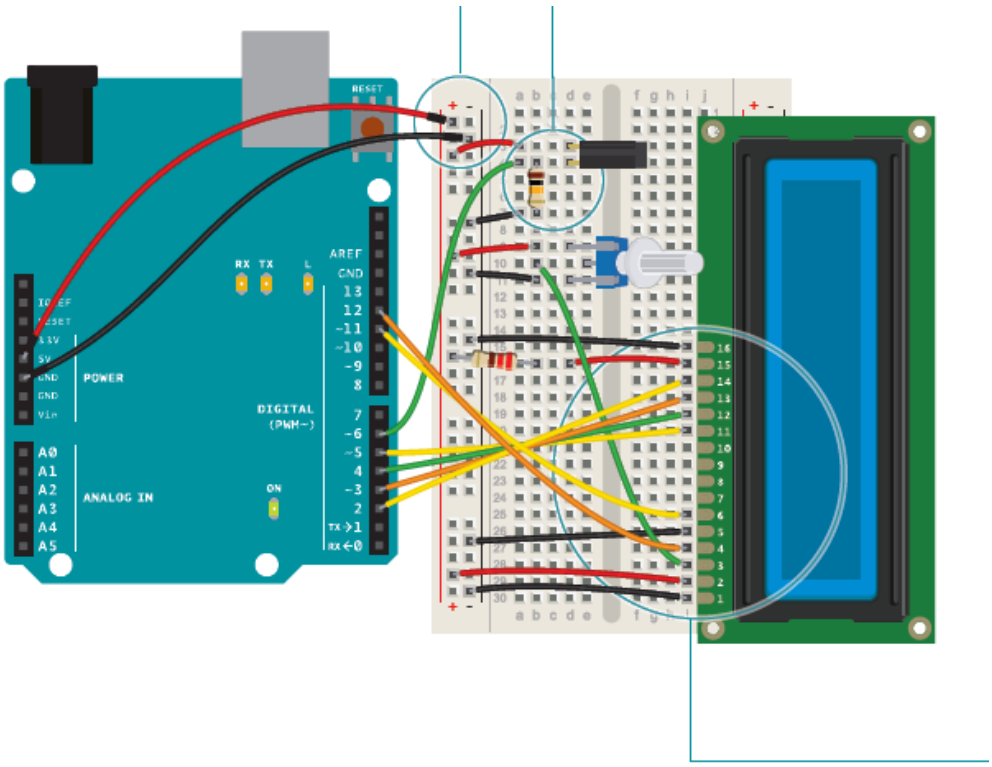


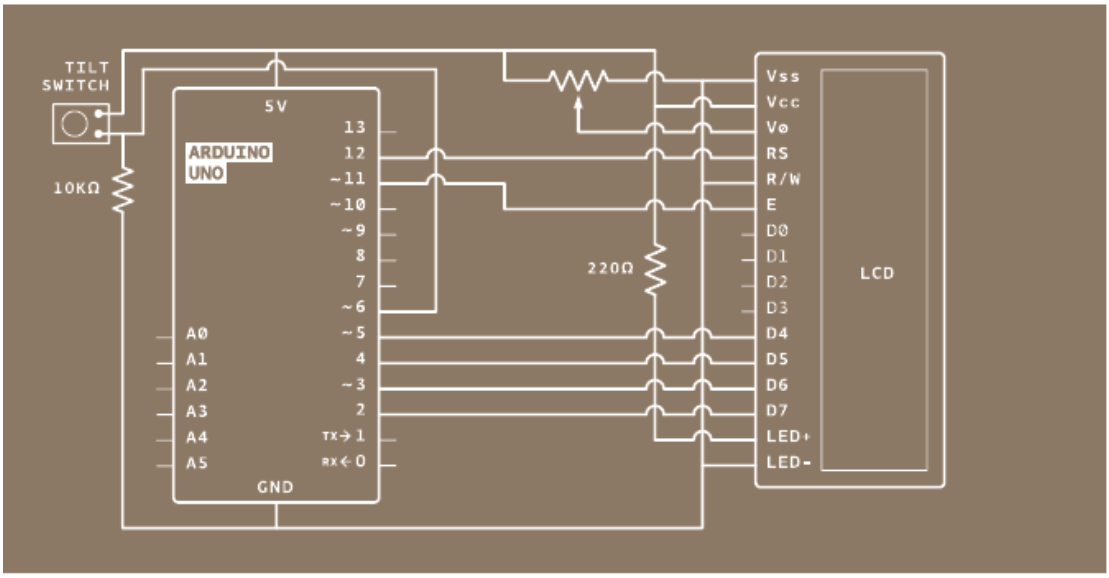
2.11 CRYSTAL BALL

Crystal balls can help "predict" the future. You ask a question to the all-knowing ball, and turn it over to reveal an answer. The answers will be predetermined, but you can write in anything you like. You'll

use your Arduino to choose from a total of 8 responses. The tilt switch in your kit will help replicate the motion of shaking the ball for answers.

The LCD can be used to display alphanumeric characters. The one in your kit has 16 columns and 2 rows, for a total of 32 characters. There are a large number of connections on the board. These pins are used for power and communication, so it knows what to write on screen, but you won't need to connect all of them.

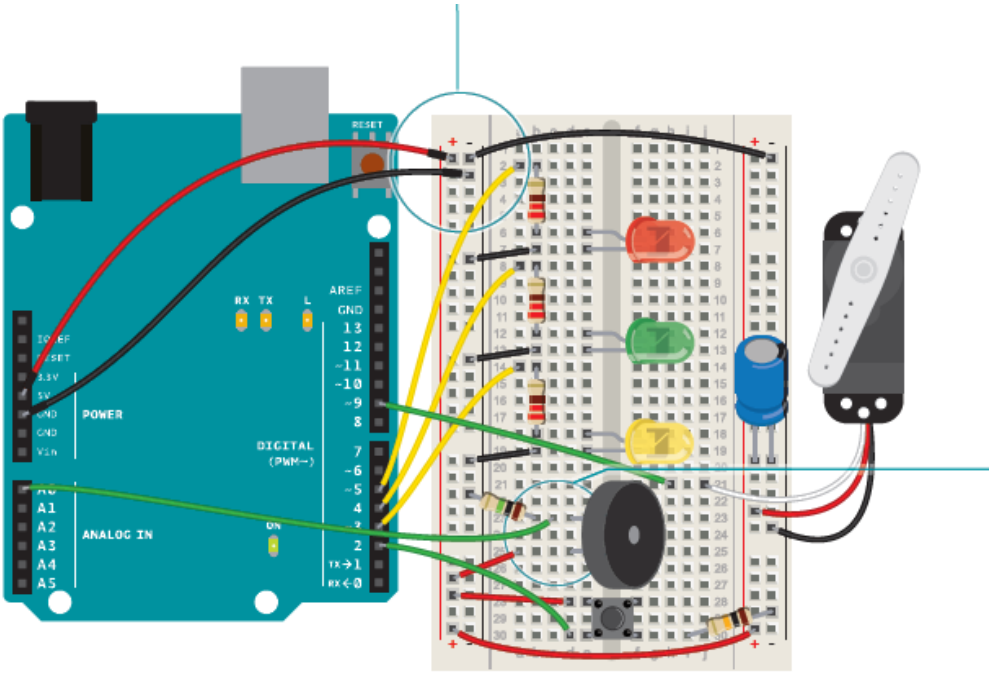


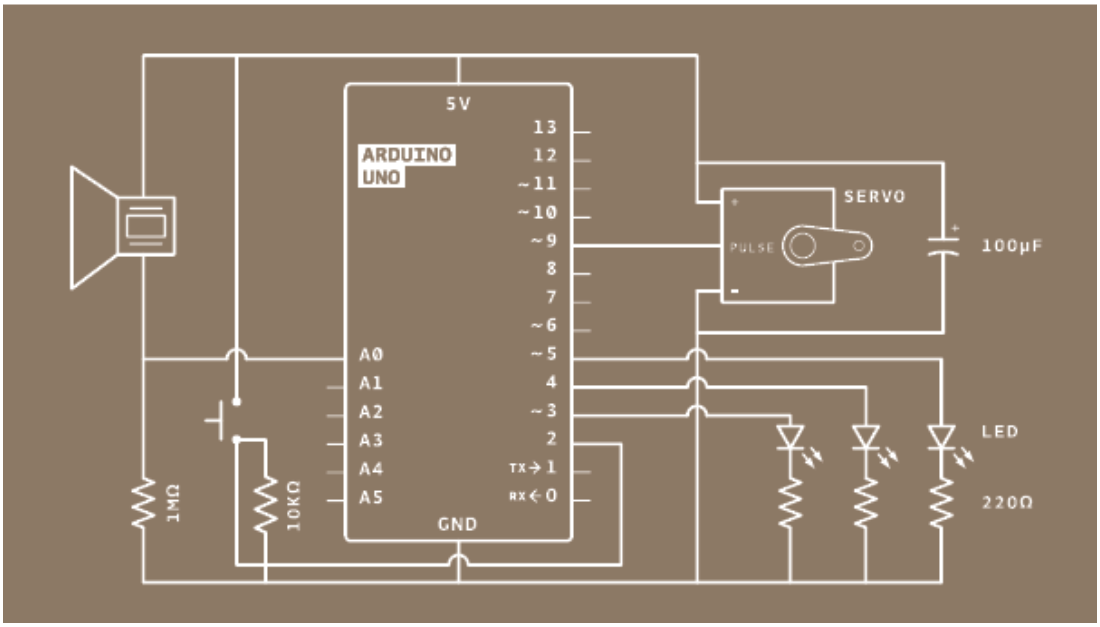


2.12 KNCK LOCK

The piezo you used for playing back sounds in the theremin and keyboard projects can also be used as an input device. When plugged into 5V, the sensor can detect vibrations that can be read by the Arduino's analog inputs. You'll need to plug in a high value resistor (like 1-megohm) as the reference to ground for this to work well.

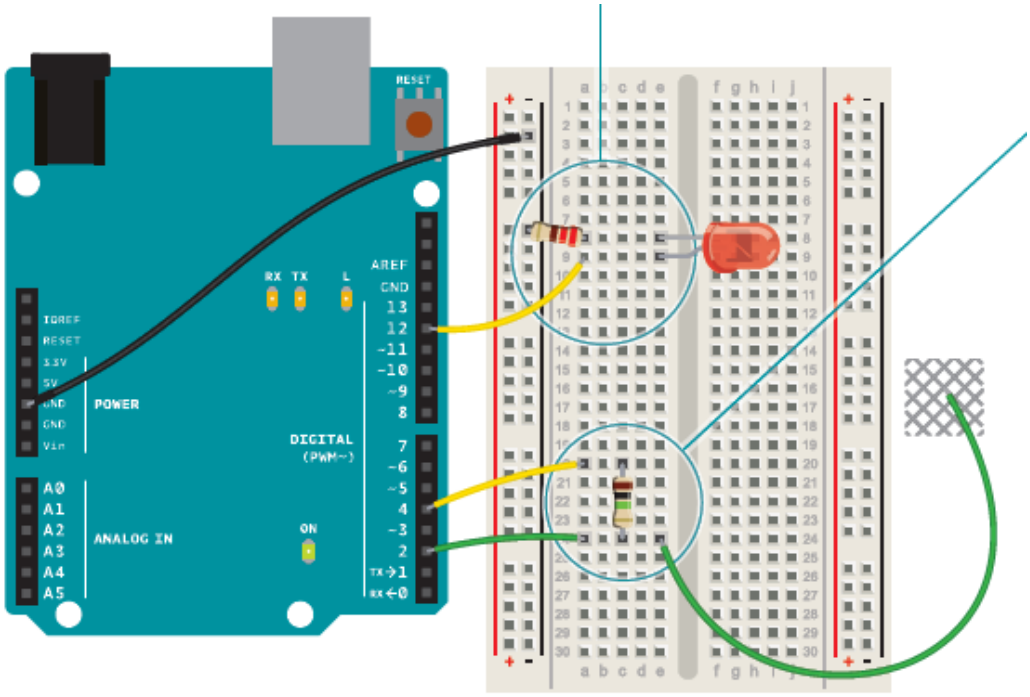
When the piezo is pressed flat against a solid surface that can vibrate, like a wooden table top, your Arduino can sense how intense a knock is. Using this information you can check to see if a number of knocks fall in an acceptable range. In code you can track the number of knocks and see if they match your settings.
A switch will let you lock the motor in place. Some LEDs will give you status: a red LED will indicate the box is locked, a green LED will indicate the box is unlocked, and a yellow LED lets you know if a valid knock has been received.
You'll also be writing your own function that will let you know if a knock is too loud or too soft. Writing your own function helps save time programming by reusing code instead of writing it out many times. Functions can take arguments and return values. In this case, you'll give a function the volume of the knock. If it is in the right range, you'll increment a variable.

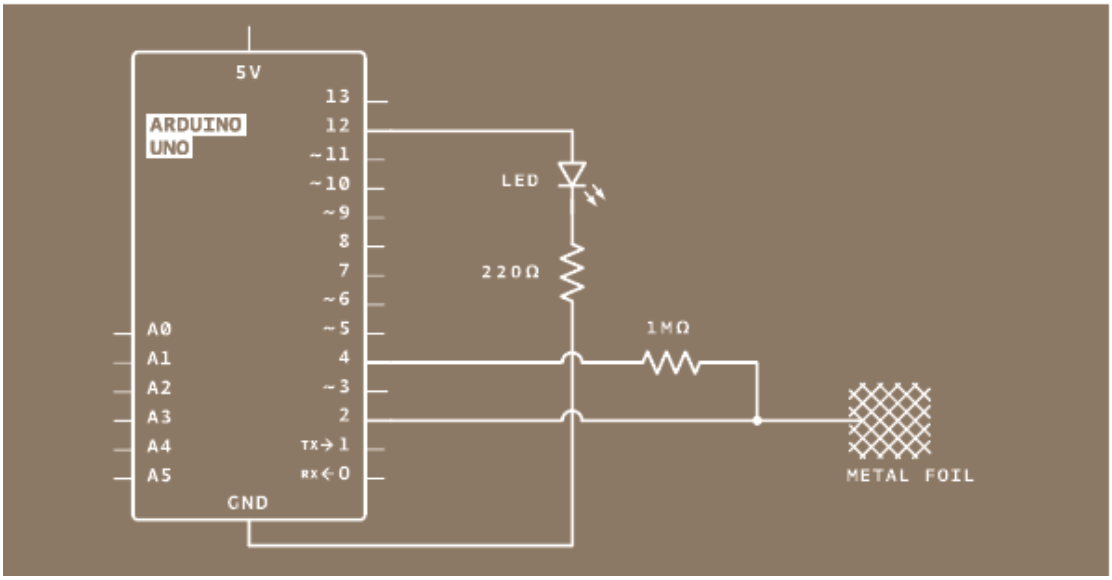




2.13 TOUCHY-FEELY LAMP

We will create a lamp that turns a light on and off when you touch a piece of conductive material





2.14 HACKING BUTTONS

While the Arduino can control a lot of things, sometimes it's easier to use tools that are created for specific purposes. Perhaps you want to control a television or a music player, or drive a remote control car. Most electronic devices have a control panel with buttons, and many of those buttons can be hacked so that you can "press" them with an Arduino. Controlling recorded sound is a good example. If you wanted to record and play back recorded sound, it would take a lot of effort to get the Arduino to do that. It's much easier to get a small device that records and plays back sound, and replace its buttons with outputs controlled by your Arduino.

Optocouplers are integrated circuits that allow you to control one circuit from a different one without any electrical connection between the two. Inside an optocoupler is an LED and a light detector. When the LED in the optocoupler is turned on by your Arduino, the light detector closes a switch internally. The switch is connected to two of the output pins (4 and 5) of the optocoupler. When the internal switch is closed, the two output pins are connected. When the switch is open, they're not connected. This way, it's possible to close switches on other devices without connecting them to your Arduino.