In this example, the diagrams are for controlling a digital recording module that allows you to record and playback 20 seconds of sound, but the basic premise holds for any device that has a switch you can access. While it's possible to use this example without soldering any wires, it certainly makes things easier. For more information on soldering, see p. 134.

BUILD THE CIRCUIT

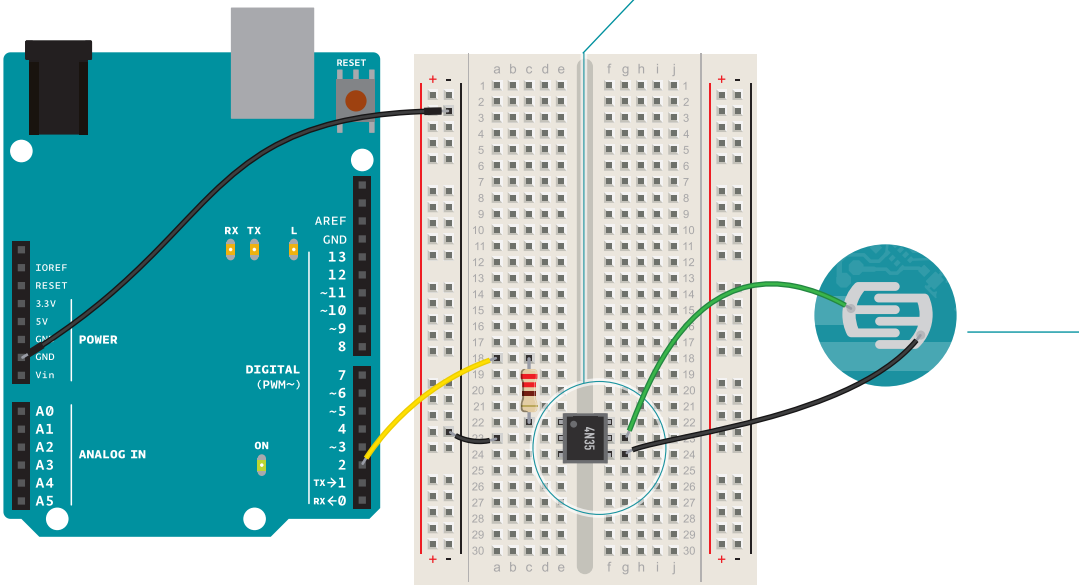


Fig. 1

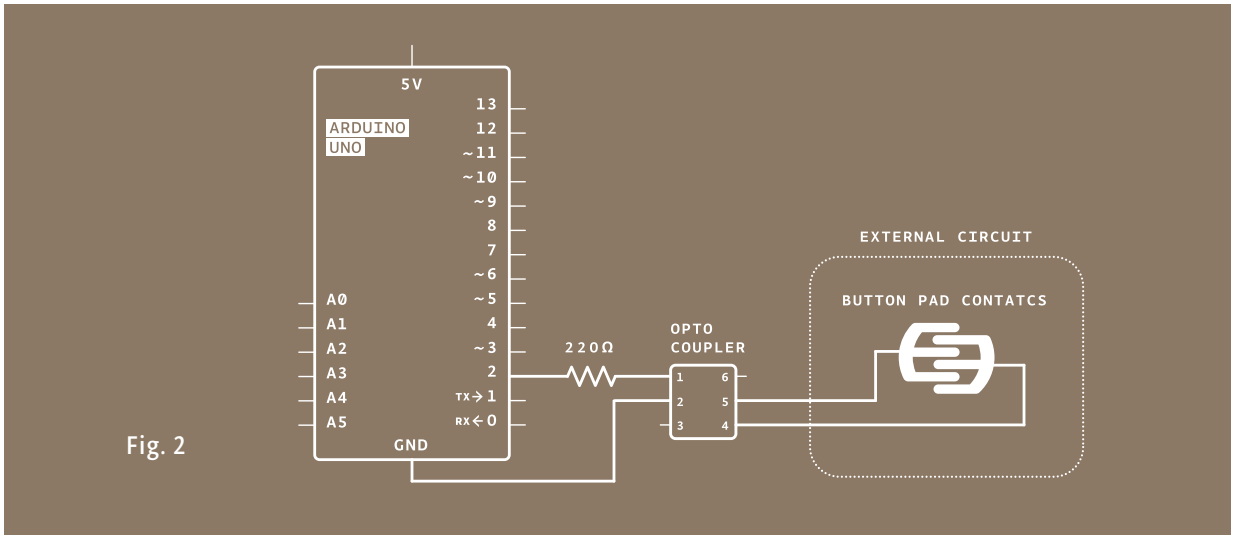


Fig. 2

1. Connect ground to your breadboard through the Arduino.

2. Place the optocoupler on your breadboard so that it straddles the center of the board (see circuit diagram).

3. Connect pin 1 on the optocoupler to Arduino pin 2 in series with a 220-ohm resistor (remember, you're powering an LED inside, you don't want to burn it out). Connect pin 2 of the optocoupler to ground.

4. On the main board of the sound module there are a number of electrical components, including a playback button.
To control the switch, you're going to have to remove the button. Flip the circuit board over and find the tabs that hold the button in place. Gently bend the tabs back and remove the button from the board.

5. Under the button are two small metal plates. This pattern is typical of many electronic devices with pushbuttons. The two "forks" of this pattern are the two sides of the switch. A small metal disc inside the pushbutton connects these two forks when you press the button.

6. When the forks are connected, the switch is closed on the circuit board. You will be closing the switch with the optocoupler.
This method, closing a switch with an optocoupler, works only if one of the two sides of the pushbutton's switch is connected to ground on your device. If you're not sure, take a multimeter and measure the voltage between one of the forks and the ground on your device. You need to do this with the device turned on, so be careful not to touch anywhere else on the board.
Once you know which fork is ground, disconnect the power to your device.

7. Next, connect one wire to each of the small metal plates. If you are soldering these wires, be careful to not join the two sides of the switch together. If you are not soldering and using tape, make sure your connection is secure, or the switch won't close. Make sure neither wire connects to the other fork, or your switch will be closed all the time.

8. Attach the two wires to pins 4 and 5 of the optocoupler. Connect the side of the switch that is grounded to pin 4 of the optocoupler. Connect the other fork to pin 5 of the optocoupler.

THE CODE

Name a constant

Most of the fun with this project is in the circuit and the optocoupler. The code is similar to the first project you made with the Arduino. You're going to play the sound once every 20 seconds by turning pin 2 `HIGH`.
Create a constant for the optocoupler control pin.

Configure the pin direction

In `setup()`, set the optocoupler pin into an output.

Pull the pin high and low

The `loop()` turns optoPin `HIGH` for a few milliseconds, long enough for the optocoupler to close the switch on the device. Then the optoPin becomes `LOW`.

Wait for a little while

Wait for 21 seconds for the whole message to play back before starting the `loop()` again.

USE IT

Attach the battery to the sound recorder. Press and hold the record button on the device. While you're holding the button, you can record audio into the microphone. Use your voice, the cat, or the pots and pans in the kitchen to make some noise (but be careful with the cat).

Once you've satisfactorily recorded a sound, power your Arduino with the USB cable. Your recording should start to play. If you recorded for the full 20 seconds, the sound should start again just a few moments after it ends.

Try experimenting with different sounds and durations of toggling the playback with the `delay()` in your program.
If you trigger the switch while a sound is playing, it will stop. How can you take advantage of this to create unique sequences of sounds?

```
1 const int optoPin = 2;
```

```
2 void setup(){
3   pinMode(optoPin, OUTPUT);
4 }
```

```
5 void loop(){
6   digitalWrite(optoPin, HIGH);
7   delay(15);
8   digitalWrite(optoPin, LOW);
```

```
9   delay(21000);
10 }
```

Integrated circuits are in virtually every electronic device. The large 28 pin chip on your Arduino is an IC that houses the brains of the board. There are other ICs that support this one with communication and power. The optocoupler and main chip on the Arduino are ***Dual In-line Package (DIP)*** chips. These DIP chips are the kind that most hobbyists use because they easily fit in a breadboard and don't have to be permanently soldered to be used.

The project example only played sound back at a regular interval. How could you incorporate the inputs from earlier projects to trigger these sounds? What other battery powered things do you have around the house that need an Arduino to control them? This technique of controlling an electronic device with an optocoupler by connecting to the two sides of a switch can be used in many other devices. What other devices do you want to control?

Optocouplers can control devices that are on a different circuit. The two circuits are electrically separated from each other inside the component.

A/Z

Accelerometer -
Actuator -
Alternating current -
Amperage (amps or amperes) -
Analog -
Analog-to-Digital Converter (ADC) -
Anode -
Argument -
Array -
Back-voltage -
Baud -
Binary -
Bit -
Boolean -
Byte -
Calibration -
Capacitance -
Cathode -
Circuit -
Common cathode LED -
Conductor -
Constant -
Current -
Datasheet -
Datatype -
Debugging -
Decoupling capacitors -
Digital -
Direct current -
Drain (transistor) -
Dual In-line Package (DIP) -
Duty cycle -
Electricity -
Float -
Function -
Gate -
Global variable -
Ground -
IDE -
Index -
Induction -
Instance -
Insulator -
Int -
Integrated Circuit (IC)-
Library -
Load -
Local variable -
Long -
Microcontroller -
Millisecond -
Object -
Ohm -
Ohm's Law -
Optocoupler -
Parallel -
Parameter -
Period -
Photocell -
Photoresistor -
Phototransistor -
Polarized -
Power supply -
Processing -
Pseudocode -
Pulse Width Modulation (PWM) -
Resistance -
Sensor -
Serial buffer -
Serial communication -
Serial monitor -
Series -
Short circuit -
Sketch -
Soldering -
Source (transistor) -
Square wave -
Switch -
Transducer -
Transistor -
Unsigned -
USB -
Variable -
Voltage -
Voltage divider -

GLOSSARY

THERE ARE A NUMBER OF NEW TERMS YOU'VE LEARNED IN THESE PROJECTS. WE'VE COLLECTED THEM ALL HERE FOR REFERENCE

A

Accelerometer **-** A sensor that measures acceleration. Sometimes, they are used to detect orientation, or tilt.

Actuator **-** A type of component that changes electrical energy into motion. Motors are a type of actuator.

Alternating current **-** A type of current where electricity changes its direction periodically. This is the sort of electricity that comes out of a wall socket.

Amperage (amps or amperes) **-** The amount of electrical charge flowing past a specific point in your circuit. Describes the current as it flows through a conductor, like a wire.

Analog **-** Something that can continuously vary over time.

Analog-to-Digital Converter (ADC) **-** A circuit that converts an analog voltage into a digital number representing that voltage. This circuit is built-in to the microcontroller, and is connected to the analog input pins A0-A5. Converting an analog voltage into a digital number takes a tiny bit of time, so we always follow the analogRead() with a short delay().

Anode **-** The positive end of a capacitor or diode (remember that an LED is a type of diode).

Argument **-** A type of data supplied to a function as an input. For example, for digitalRead() to know what pin to check, it takes an argument in the form of a pin number.

Array **-** In programming, this is a group of variables that are identified by one name, and accessed by an index number.

B

Back-voltage - Voltage that pushes back against the current that created it. It can be created by motors spinning down. This can damage circuits, which is why diodes are often used in conjunction with motors.

Baud - Shorthand for "bits per second", signifying the speed at which computers are communicating with each other.

Binary - A system with only two states, like true/false or off/on.

Bit - The smallest piece of information a computer can send or receive. It has two states, 0 and 1.

Boolean - A datatype that indicates if something is true or false.

Byte - 8 bits of information. A byte can hold a number between 0 and 255.

C

Calibration - The process of making adjustments to certain numbers or components to get the best results from a circuit or program. In Arduino projects, this is often used when sensors in the real world may give different values in different circumstances, for instance the amount of light on a photoresistor. Calibration can be automatic, as in Project 6, or manual, as in Project 3.

Capacitance - The ability of something to hold an electrical charge. This charge can be measured with the Capacitive Sensor library, as seen in Project 13.

Cathode - The end of a capacitor or diode that typically connects to ground.

Circuit - A circular path from a power supply, through a load, and then back again to the other end of the power supply. Current flows in a circuit only if it is closed, that is, if the outgoing and return path are both uninterrupted, or closed. If either path is interrupted, or open, then current will not flow through the circuit.

Common cathode LED - Types of LEDs that have multiple colors in one fixture, with one cathode and multiple anodes.

Conductor - Something that enables electricity to flow, like a wire.

Constant - A named identifier that cannot change its value in a program.

Current - The flow of electrical charge through a closed circuit. Measured in amps.

D

Datasheet - A document written by engineers for other engineers that describes the design and functionality of electrical components. Typical information in a datasheet includes the maximum voltage and current a component requires, as well as an explanation of the functionality of the pins.

Datatype - A classification system that determines what values a particular constant, variable, or array will hold. Int, float, long and boolean are all types that can be used in Arduino.

Debugging - The process of going through a circuit or code, and finding errors (also referred as "bugs"), until the expected behavior is achieved.

Decoupling capacitors - Capacitors that are used to regulate spikes and dips in voltage, often placed close to a sensor or actuator.

Digital - A system of discrete values. As Arduino is a type of digital device, it only knows of two discrete states, off and on, nothing in between.

Direct current - A type of current which always flows in the same direction. All the projects in this kit use direct current.

Drain (transistor) - The pin that connects to the higher current/voltage load to be controlled.

Dual In-line Package (DIP) - A type of packaging for integrated circuits that allows the components to be easily inserted into a breadboard.

Duty cycle - A ratio indicating the amount of time over a certain period that a component is turned on. When using a PWM value of 127 (out of a total of 256), you're creating a 50% duty cycle.

E

Electricity - A type of energy generated by electric charges. You can use electronic components to change electricity to other forms of energy, like light and heat.

F

Float - A datatype that can be expressed as a fraction. This entails the use of decimal points for floating point numbers.

Function - A block of code that executes a specific task repeatedly.

G

Gate - The pin on a transistor that is connected to the Arduino. When the gate is turned on, by applying 5V, it closes the junction between drain and source, completing the circuit it is connected to.

Global variable - A named variable that can be accessed anywhere inside your program. It is declared before the setup() function.

Ground - The point of a circuit where there is 0 potential electrical energy. Without a ground, electricity will not have a place to flow in a circuit.

I

IDE - Stands for "Integrated Development Environment". The Arduino IDE is the place where you write software to upload to the Arduino. It contains all the functions the Arduino can understand. Other programming environments, like Processing, have their own IDE.

Index - The number supplied to an array that indicates which element you're referring to. Computers are zero-indexed, which means they start counting at 0 instead of 1. To access the third element in an array named tones, for example, you would write tones[2].

Induction - The process of using electrical energy to create a magnetic field. This is used in motors to spin them around.

Instance - A copy of a software object. You're using instances of the Servo library in Projects 5 and 12, in each case, you're creating a named instance of the Servo library to use in the project.

Insulator - Something that prevents electricity from flowing. Conductive materials like wires are often covered in insulators like rubber.

Int - A datatype that holds a whole number between –32,768 and 32,767.

Integrated Circuit (IC) - A circuit that has been created on a tiny piece of silicon and embedded in plastic (or epoxy). Pins, or legs, protruding from the body allow you to interact with the circuit inside. Very often we can make good use of an IC knowing only what to connect to the pins without having to understand how it functions internally.

L

Library - A piece of code that expands the functionality of a program. In the case of Arduino libraries, they either enable communication with a particular piece of hardware, or are used for manipulating data.

Load - A device that turns electrical energy into something else, such as light, heat, or sound.

Local variable - A type of variable that is used for a short amount of time, then forgotten. A variable declared inside the setup() of a program would be local: after the setup() finished running, the Arduino would forget that the variable ever existed.

Long - A datatype that can hold a very large number, from –2,147,483,648 to 2,147,483,647.

M

Microcontroller - The brains of the Arduino, this is a small computer that you will program to listen for, process, and display information.

Millisecond - 1/1,000th of a second. The Arduino goes through its programs so fast, when calling delay() and other time based functions, it's done in milliseconds.

O

Object - An instance of a library. When using the Servo library, were you to create an instance named myServo, myServo would be the object.

Ohm - Unit of measurement of resistance. Represented by the omega symbol (Ω).

Ohm's Law - A mathematical equation that demonstrates the relationship between resistance, current and voltage. Usually stated as V (voltage) = I (current) x R (resistance).

Optocoupler - Also known as an opto-isolator, photo-coupler, photo-isolator, photo-switch, and opto-switch. An LED is combined in a sealed case with a phototransistor. The LED is positioned to illuminate the phototransistor, so that when the LED is turned on, the phototransistor will conduct. Used to provide a high degree of isolation as there is no electrical connection common to the input and the output.

P

Parallel - Components connected across the same two points in a circuit are in parallel. Parallel components always have the same voltage drop across them.

Parameter - When declaring a function, a

named parameter serves as the bridge between the local variables in the function, and the arguments it receives when the function is called.

Period - A specific span of time in which something happens. When the period changes, you're adjusting the frequency at which something will occur.

Photocell - A device for converting light energy to electrical energy.

Photoresistor - A resistive device whose resistance varies according to how much light falls on it.

Phototransistor - A transistor which is controlled by light rather than by current.

Polarized - The leads of polarized components (e.g. LEDs or capacitors) have different functions, and thus must be connected the right way. Polarized components connected the wrong way might not work, might be damaged, or might damage other parts of your circuit.
Non-polarized components (e.g. resistors) can be connected either way.

Power supply - A source of energy, usually a battery, transformer, or even the USB port of your computer. Comes in many varieties such as regulated or unregulated, AC or DC. Usually the voltage is specified, along with the maximum current the supply can deliver before failing.

Processing - A programming environment based on the Java language. Used as a tool to introduce people to the concepts of programming, and in production environments.
The Arduino IDE is written in Processing, and so will look very familiar. In addition, Processing and Arduino share a similar philosophy and motive, of creating free open source tools allowing non-technical people to work with hardware and software.

Pseudocode - A bridge between writing in a computer programming language and using natural speech. When creating pseudocode, it's helpful to write in short declarative statements.

Pulse Width Modulation (PWM) - A way to simulate an analog output when using a digital device, PWM involves turning a pin on and off at a very rapid rate.The ratio of ON time to OFF time determines the simulated analog result.

R

Resistance - A measure of how efficiently a material will conduct electricity. In particular, resistance can be calculated by Ohm's Law as: R = V/I.

S

Sensor - A component that measures one form of energy (like light or heat or mechanical energy) and converts it to electrical energy, which the Arduino can understand.

Serial buffer - A place in your computer's and microcontroller's memory where information received in serial communication is stored until it is read by a program.

Serial communication - The means by which the Arduino communicates with computers and other devices. It involves sending one bit of information at a time in a sequential order. The Arduino has a USB-to-serial converter onboard, which enables it to talk with devices that don't have a dedicated serial port.

Serial monitor - A tool built in to the Arduino IDE allowing sending and receiving serial data to and from a connected Arduino. See the

Serial() set of functions.

Series - Components are in series when current flows from the first into the next. The current flowing through both is the same, and the voltage drops across each component.

Short circuit - A short circuit between power and ground will make your circuit stop working and thus should be avoided. In some cases this might damage your power supply or parts of your circuit, and rare cases might start a fire.

Sketch - The term given to programs written in the Arduino IDE.

Soldering - The process of making an electrical connection by melting solder over electrical components or wires that are to be connected. This provides a solid connection between components.

Source (transistor) - The pin on a transistor that connects to ground. When the gate receives power, the source and drain are connected, completing the circuit that is being controlled.

Square wave - A type of waveform that is identified by having only two states, on and off. When used to generate tones, they can sound "buzzy".

Switch - A component that can open or close an electrical circuit. There are many different kinds of switches; the ones in the kit are momentary meaning, they only close the circuit while being pressed.

T

Transducer - Something that changes one form of energy into another.

Transistor - A 3 terminal (usually) electronic device which can act as either an amplifier or a switch. A control voltage or current between two leads controls a (usually) higher voltage or current between a different pair of leads. Common types of transistors include the Bipolar Junction Transistor (BJT) and the Metal Oxide Semiconductor Field Effect Transistor (MOSFET). Often used to allow a small current from an Arduino (limited to 40 mA) to control substantially larger currents, such as those needed by motors, relays, or incandescent lamps. Depending on how they are constructed, transistors are either N-channel or P-channel, which determines how they should be connected.

U

Unsigned - A term used to describe some datatypes, indicating that they cannot be a negative number. It's helpful to have an unsigned number if you only need to count in one direction. For instance, when keeping track of time with millis(), it's advisable to use the unsigned long datatype.

USB - Stands for Universal Serial Bus. It's a generic port that is standard on most computers today. With a USB cable, it's possible to program and power an Arduino over a USB connection.

V

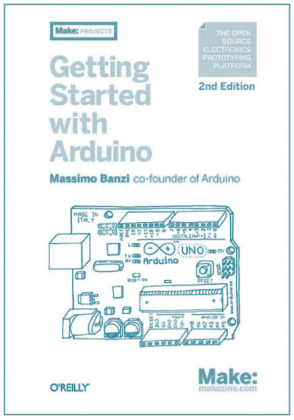
Variable - A place in your computer's or microcontroller's memory for storing information needed in a program. Variables store values which are likely to change as your program runs. A variable's type depends on the type of information you want to store, and the maximum size of the information; for example, a byte can store up to 256 differ-

ent values, but an int can store up t 65,536 different values. Variables can be local to a particular block of code, or global to an entire program. (see Global variable, Local variable).

Voltage - A measure of potential energy, that a charge might be pushed with if provided a closed circuit.

Voltage divider - A type of circuit that provides an output that is a fraction of its input voltage. You are building a voltage divider when you combine a photoresistor with a fixed resistor to provide an analog input. A potentiometer is another example of a voltage divider.

FURTHER READING



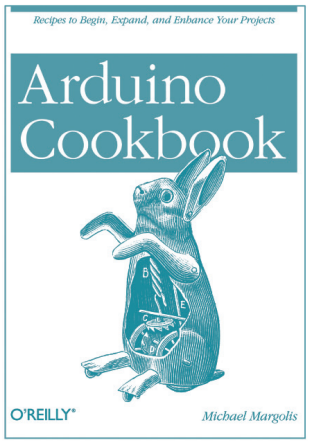
Getting Started with Arduino by **Massimo Banzi** [O'Reilly Media / Make, 2011]. The definitive introduction to Arduino.

Getting Started with Processing by **Casey Reas** and **Ben Fry** [O'Reilly Media / Make, 2010]. This short guide to the Processing programming environment tells you more about how to program graphics, sounds, and multimedia on your computer.

Making Things Talk, 2nd Edition by **Tom Igoe** [O'Reilly Media / Make, 2011]. Written for more experienced Arduino users, this book gives you many techniques for communicating between Arduino microcontrollers and other devices on the internet, and beyond.

Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction by **Daniel Shiffman** [Morgan Kaufman, 2009]. An in-depth introduction to programming using Processing, for beginners of all ages.

Getting Started with RFID by **Tom Igoe** [O'Reilly Media / Make, 2012]. A short introduction to using Radio Frequency Identification with Arduino and Processing.



The Arduino Cookbook, 2nd Edition by **Michael Margolis** [O'Reilly Media / Make, 2011]. This book has a lot of great recipes for how to use Arduino in more advanced ways.

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists by **Dustyn Roberts** [McGraw-Hill, 2010]. A great resource on building movable mechanisms to interface with your projects.

Make: Electronics, by **Charles Platt** [O'Reilly Media / Make, 2009]. Cleverly written introduction to electronics suitable for just about anyone. No Arduinos were used in the making of this book, but it's a valuable text to understand electronics better.

iOS Sensor Apps with Arduino, by **Alasdair Allan** [O'Reilly Media / Make, 2011]. With this concise guide, you'll learn how to connect an external sensor to an iOS device and have them talk to each other through Arduino. You'll also build an iOS application that will parse the sensor values it receives and plot the resulting measurements, all in real-time.