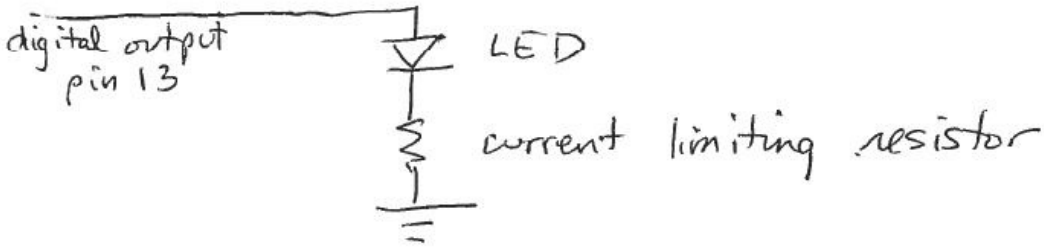
First Exercise: A Blinky LED

First, build the circuit:



The resistor is to keep the current draw down to around 20 mA. Figure out what you'll need: the digital output pin gives 5 V, treat the LED as a short (no resistance) when it is conducting.

LED facts: Short leg = cathode = - side (some have a flat side on the case)
Long leg = anode = + side

Here's a sketch that will run it

```
//firstcode-blinkyLED

const int LED =13 ; // connect LED to pin 13

void setup ()
{
pinMode(LED, OUTPUT);  //sets the digital pin as an output

}

void loop()
{
digitalWrite(LED, HIGH);
delay(1000);
digitalWrite(LED, LOW);
delay(1000);

}
```

Type this into the IDE and compile it. Attach the Arduino with the USB cable, and upload it.
Voila, blinky lights.

Add Ons – More Blinky Lights:

Add another digital output pin going to a second LED, or two more pins to two more LEDs, etc.

Modify the code accordingly. Don't forget to define the pin up at the top, and in the setup function.
Try making them blink different sequences or patterns....

Add a Switch:

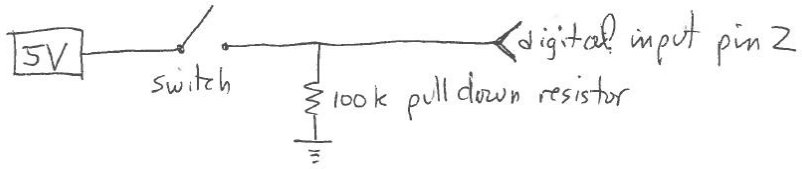
You can easily add a digital line used as an INPUT rather than an OUTPUT... to use as a switch. When that line is HIGH, the blinkys work, when it is LOW, they shut off. The code looks something like this:

```
// two blinky LEDs with a switch to control them

const int LED =13 ; // connect LED to pin 13
const int LED2=5;   // connect second LED to pin 5
const int SWITCH=2;
int val=0;

void setup ()
{
  pinMode(LED, OUTPUT);  //sets this digital pin as an output
  pinMode(LED2,OUTPUT); //sets the digital pin as an output
  pinMode(SWITCH,INPUT); //sets pin 2 as a switch digital input
}

void loop()
{
   val=digitalRead(SWITCH); //decide if switch pin 2 is hi or lo

   if (val==HIGH) {         //   if its hi, blinky pattern
  … insert your code for whatever blink pattern you like …
   }

  else {                          //if its lo, shut the LEDs off
  digitalWrite(LED,LOW);
  digitalWrite(LED2,LOW);}
}
```

Note that what gets done, in the “if ...else” statements, must be enclosed in curly brackets.

At first, use just a wire from the digital input pin (2, in this sketch) and you manually touch it to either ground or the 5 V source. Note that if the input pin is not connected to anything, it acts as if it is HIGH. This is common, logic “floats UP” rather than acting like ground.

After that works, wire in a real push button switch. These are SPDT (single pole, double throw) momentary switches. You only need one side, a SPST switch would have worked, but we don't have any. Use a DVM ohmmeter to figure out how the push button switches work (momentary switch, it makes contact only while you're pushing it). You will need to “pull down” the line so it doesn't float up to HIGH all the time:



Note what this resistor does. When the switch is open, it just ties the digital input pin to ground (LOW). When the switch is closed, the digital pin will be HIGH, and there will be almost no current drain through the resistor since it's 100 kohms.

Dimming LED, Pulse Width Modulation (PWM), and analogWrite():

So far, you've used digitalRead(pin) and digitalWrite(pin,HiLo) where pin is the constant for the pin number and HiLo is either HIGH or LOW. Thus you have digital in, and out.

We can also sort of do an analog output. The Arduino doesn't have a “real” analog output. What it does, is take some of the digital output pins, and send a pulse train to them but modulate the width of each pulse. Averaged, this is a cheesy kind of analog signal. It's only a 8 bit conversion on the pulse width. Effectively, this means that we can get 1 part in 2^8=256 resolution or (5 V/256= 20 mV “step size”). Still, it's an easy and effective rough analog output.

The digital output pins that have a ~ before the number, as printed on the Arduino board, are capable of PWM analog output. Those are pins 3, 5, 6, 9, 10 and 11.

The command is

analogWrite(pin,N)

where pin is the pin number, and N is an integer between 0 and 255.

To make a slowly increasing and decreasing brightness LED, wire the LED up to one of the PWM pins, with a current limiting resistor in line of course (200 to 300 Ohms should do it).

You will also need to be able to “loop” through the various settings. So far you have not used a loop of any sort... here's an example:

```
const int LED =3 ; // connect LED to pin 3
int i=0;

void setup ()
{
  pinMode(LED, OUTPUT);  //sets the digital pin as an output

}

void loop()
{

  for (i=20; i<255; i++)
  {  analogWrite(LED, i);
     delay(15); }

  for (i=255; i>20; i--)
  {  analogWrite(LED,i);
     delay (15);}
}
```

Can you make it work?

Piezo-Buzzer

Piezo buzzers make noises. The Arduino can control the pitch, volume, and duration of the sound.

The piezo is not polarized and can be attached directly from a digital pin to ground. Wire it up!

Piezo elements work best at higher frequencies. They aren't very loud on a good day.... and are even softer down at lower frequencies. These are best for sound frequencies above 2 kHz.

Set up PZ as a constant for whatever pin you've wired to, and set that pinmode as an output pin (you know how to do these, in your code!)

In the main loop put something like:

```
{
  tone(PZ,1760);
  delay(1000);
  tone(PZ,880);
  delay(1000);
  noTone(8);
  delay(1000);
}
```

where the first argument to the tone function is the pin number and the second is the freqency in Hz. Note how stupid the Arduino is, you must turn the tone on, tell it how long, turn the tone off, etc.

Try it. Note that this is a pretty quiet little buzzer, you may have trouble hearing it.

You can look up what standard pitches are here https://www.arduino.cc/en/Tutorial/toneMelody and write a tune if you like. Mine can play Yankee Doodle.

If you don't care about the pitch but just want a noise, you can also turn the buzzer on with this:

```
const int PZ = 3 ; // connect piezo to pin 3 for analog write
int time = 500;
void setup ()
{
  pinMode(PZ, OUTPUT);  //sets the digital pin as an output
}

void loop()
{
  analogWrite(PZ,128);
  delay(time);
  digitalWrite(PZ,LOW);
  delay(time);
}
```

Try this.

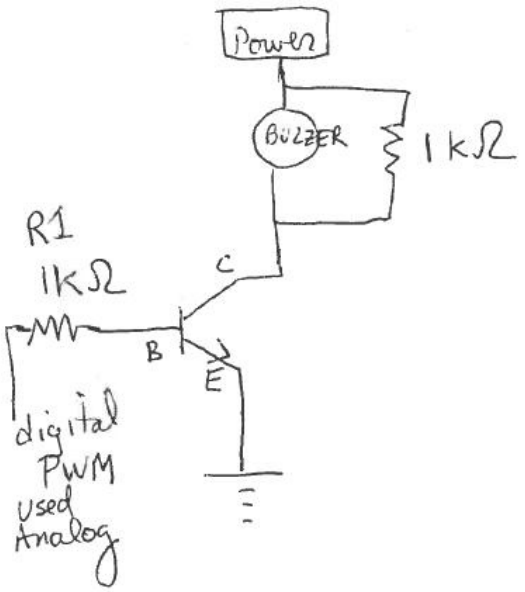
Louder Piezo – How to use a Transistor for Heavier Loads

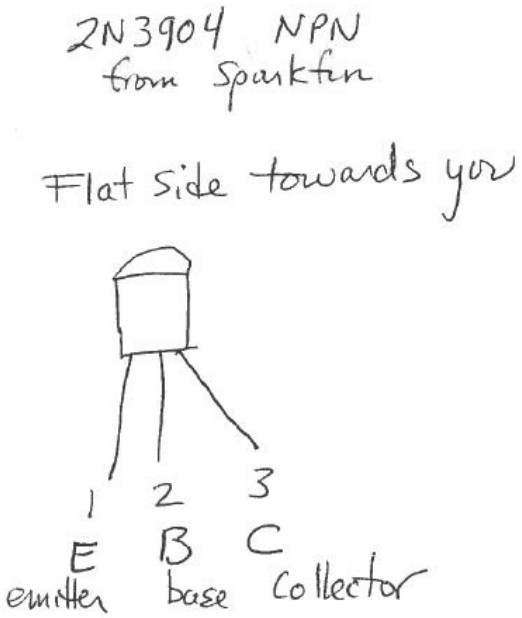
In order to make the piezo louder, we need to switch a heftier power supply through it. The wimpy little 5 V, 20 mA digital pins or their analog PWM version, just isn't sourcing enough power.

The addition of one transistor does the trick.

Remember, a transistor is a current amplifier. Whatever the base current is (which should be small), it amplifies greatly and a much bigger current runs from collector to emitter or the other way around, depending on the type of transistor. So you can treat them as a switch for heftier current loads, or to "switch" a higher voltage supply into the circuit.

Use the 2N3904 transistors. They are rated for 200 mA collector current.





Attach it to power at 5 V, 9 V, 12 V, and run the sketch you used before. Can you hear the difference in loudness? Or put a bench power supply on it and run it up to 30 V.

To "analyze" this circuit, treat the C-E path as a short when the transistor is conducting. What are the two resistors for? Think about the current from the Arduino (limited by R1) and the maximum collector current.

Light Sensor

You have a CdS photoresistor – it has two legs and a weird squiggle on the face of it. This devise has resistance that depends on the intensity of the incident light hitting it. Check this out with an ohmmeter; you should see resistances of several hundred kOhm when it's dark and only a couple kOhm in bright light.

Here's a little circuit that makes the blink-rate for an LED depend on the ambient light level. Depending on what the photoresistor reads, it sets different time intervals for the blinker. (I didn't show the blink circuit, you know how to do that).

This code also spits out the actual value of the analog input read, to the Serial Monitor. To use this, after you upload, click on the magnifying glass like icon at the upper right of the IDE. A second window opens, and it will display the read values.

```
//blinky LED with blink rate set by light level

const int LED =13 ; // connect LED to pin 13
int lightlevel=0;  // lightlevel is the variable to store analog in
int loopindex=0;

void setup ()
{
  pinMode(LED, OUTPUT);  //sets the digital pin as an output
                         // note analog pins are automatically inputs

  Serial.begin(9600);  // set up serial comms with computer at 9600 baud
}

void loop()
{
  lightlevel=analogRead(0);   //sensor on analog pin 0

  Serial.println(lightlevel);  // print out lightlevel to user, via serial

  for (loopindex=0;loopindex<5;loopindex++) {
    digitalWrite(LED, HIGH);
    delay(lightlevel);
    digitalWrite(LED, LOW);
    delay(lightlevel);
  }                        // 5 blinks then checks light level again
                           // this is so it doesn't spew too much at ya
}
```