Microcontrollers Systems and Interfacing

week 3 experimental lab

1 Create a steady tone on the loudspeaker

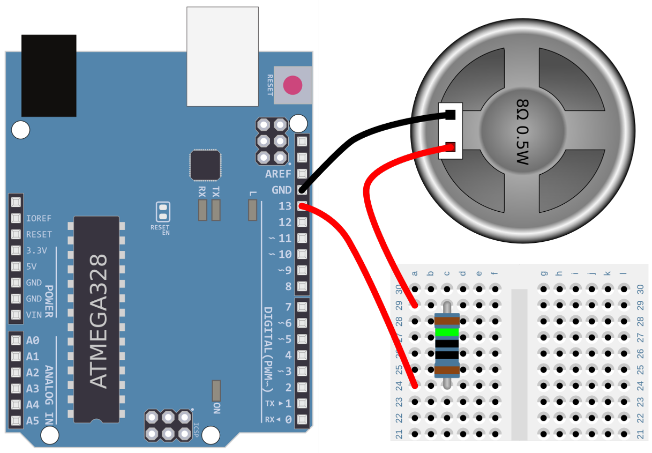
Creating a tone is just like blinking an LED — except that we need to 'blink' a loudspeaker instead of an LED, and to make an audible tone we need to 'blink' it *much* faster than a few times per second.

▶ Connect a loudspeaker to a digital output pin.

The loudspeaker has a resistance of 8 Ω. If you connect it directly to the microcontroller's 5 V output, it will draw far too much current. (Remember: the limit is 40 mA per pin.) To protect the microcontroller from too much current, you will have to place a 150 Ω resistor in series with the loudspeaker. Identify the correct resitor using its colour code. You can double-check the value using the '200 Ω' resistance range on your multimeter. (Remember to switch it off after use or the batter will be discharged!)

Any digital output pin will work. Use pin 13 again if you are most comfortable with that one.

Don't forget to configure the pin as an `OUTPUT` in your `setup()` function using `pinMode()`.



▶ Send a 1 kHz tone to the loudspeaker using `digitalWrite()` and `delayMicroseconds()`.

You will have to set the loudspeaker pin to `HIGH` and then back to `LOW` 1,000 times per second. For a delay of less than one millisecond you can use the `delayMicroseconds(`*N*`)` function to delay the program for N microseconds.

2 Use an input to control an output

Use the techniques from last week, for reading an analogue input voltage, to allow a potentiometer to control the frequency of the tone played on a loudspeaker.

▶ Vary the 1 kHz tone according to the input voltage read from the potentiometer.

Increasing the delay time will reduce the frequency of the tone you produce. The `analogueRead()` function returns a value between 0 and 1023. One possibilitity is simply to add that value to your delay time to obtain a pitch control.

What are the minimum and maximum frequencies of the tones that you can produce using this method?

3 Challenges

If you want to explore further, try some of the following challenges...

3.1 Use the `tone()` function

The `tone()` library function is called with two or three parameters, like this:

`tone(`*`pin, frequency`*`)`

Generates a square wave of the given *frequency* on the given *pin*. For example, the musical note 'A' has the frequency 440 Hz. To play 'A' on pin 13 you could use

```
tone(13, 440);
```

The tone will continue playing until you change is to another frequency, or until you call `noTone()`.

`noTone(`*`pin`*`)`

Stops generating a tone on the given *pin*.

▶ Use `tone()` to generate a steady tone.

Reproduce your 1 kHz tone from earlier, usuing the `tone()` function.

▶ Use `tone()` to generate the variable (voltage-controlled) frequency from earlier.

Look up the `tone()` function in the documentation. (You can use it to set the frequency of tone on a pin directly.) Use it to set the pitch of the note produced by your loudspeaker.

3.2 Play a tune

If you generate a sequence of tones of just the right frequencies then you can play tunes. Here are the frequencies of the notes near the middle of the range of a piano. Use some of these frequencies in the correct order to play a simple tune.

If you do not know the notes of a particular tune, try using some of these frequencies (particularly the ones without a '♯' sign next to them) in a pattern 4, 8, 12, or 16 notes long. Many of the sequences you can make should sound musical.

note	*frequency*
C	262
C♯	277
D	294
D♯	311
E	330
F	349
F♯	370
G	392
G♯	415
A	440
B♭	466
B	494

3.3 Vary the speed of your tune

Use the analogue input voltage to control the speed at which your tune is played.

3.4 [TRICKY] Vary the pitch of your tune

Use the analogue input voltage to control the pitch at which your tune is played.

3.5 [DIFFICULT] Implement a simple musical instrument

Allow the user to play a tune by typing a sequence of notes into the serial monitor.

► Read and print the chatacters typed into the serial monitor.

The area at the top of the serial monitor is for text input. You can type text (such as a sequence of notes) into the area, then press RETURN, and the text will be sent to the microcontroller. To access this text you can use something like this in your program:

```
void loop() {
  if (Serial.available()) {   // data from the user is available
    int c = Serial.read();    // read one character of the data
    Serial.println(c);        // print it back to the user
    delay(100);               // wait a while
  }
}
```

In the Arduino programming language characters are really integers, so this program prints back the integer value of each of the characters that you type. You should be able to verify that space is 32, the digits start at 48, and the lower-case letters start at 65.

► Convert the characters into notes.

The `switch` and `case` commands let you choose an action based on a value. The general form of these commands is this:

```
switch (someValue) {
  case value1: action1;  break;
  case value2: action1;  break;
  case value3: action1;  break;
  // ... and so on ...
  default:     actionX;  break;
}
```

In the case *someValue* is equal to *value1* then *action1* is performed. Otherwise, if *someValue* is equal to *value2* then *action2* is performed. Otherwise, if *someValue* is equal to *value3* then *action3* is performed. By defaultm, if *someValue* is equal to none of the `case` values, then *actionX* is performed.

For example, to check which character was typed and convert it into a frequency, you could use something like this in your program:

```
switch (c) {
  case 'C': tone(13, 262);  break;  // play a C note
  case 'D': tone(13, 294);  break;  // play a D note
  // ... and so on ...
  case 'B': tone(13, 494);  break;  // play a B note
  default:  noTone(13);     xbreak; // stop playing
}
```

If you complete this part please make a video of your instrument playing a tune and send it to the instructor.

Microcontrollers Systems and Interfacing

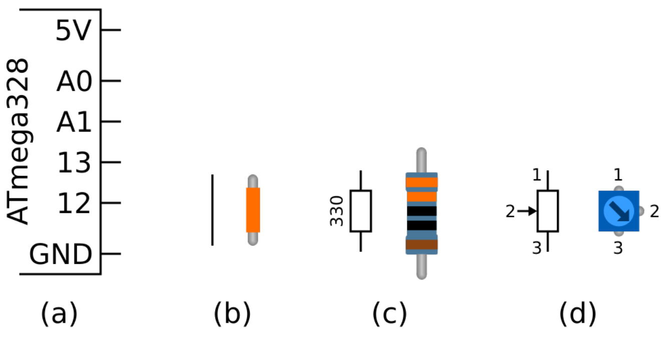
week 3 reference material

A Circuit diagrams

Electrical circuit diagrams show how components are connected to each other electrically in a circuit. Some common components are shown on the right.

(a) Complex integrated circuits like the microcontroller are drawn as a box (or part of a box) and sometimes labelled with their part number. Signal names are written inside the box and connections arrive from outside next to the name of the connected signal. Often only the signals that are relevant are shown and other signals are assumed not to be connected. In this example only the power pins 5 V and GND, analogue pins A0 and A1, and digital pins 12 and 13 have been included.

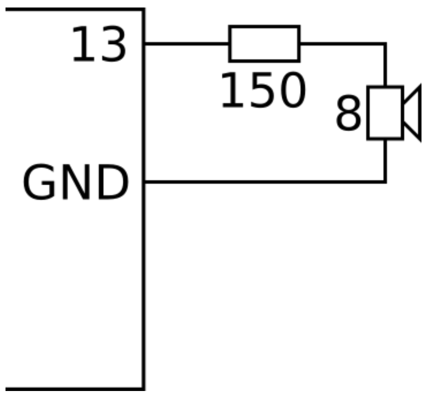
(b) Wires are drawn as a line.



(c) Resistors are drawn as a box between two wires. The value of the resistor is written next to it.

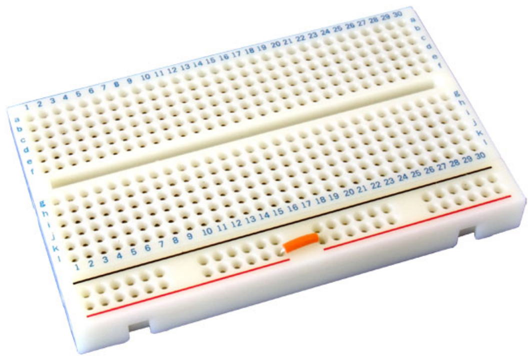
(d) Potentiometers (variable resistors) are drawn as a resistor with an extra terminal pointing to halfway along the resistor's body. If the value of the potentiometer is important it will be written next to it.

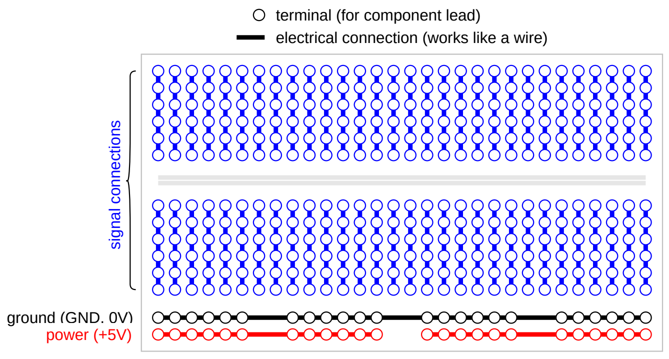
The circuit on the right uses only digital pin 13 and the ground pin of the microcontroller. Pin 13 is connected to one terminal of a 150 Ω resistor. The other terminal of the resistor is connected to one terminal of a loudspeaker. The other terminal of the loudspeaker is connected to the ground pin of the microcontroller.



B Prototyping with breadboards

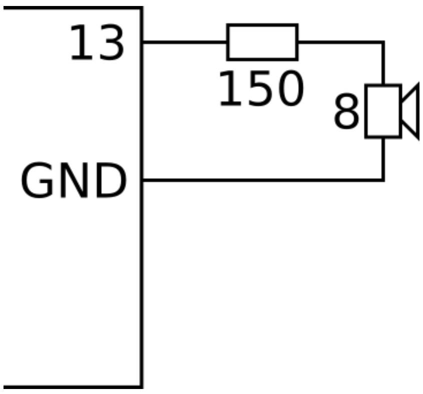
Breadboards provide a matrix of terminals into which component leads can be plugged. Internal connections in the breadboard connect the component leads together, playing the part of wires in the circuit. These internal connections cannot be modified, so part of the challenge of building a circuit on a breadboard is to arrange the components so that the internal connections create exactly the desired circuit.

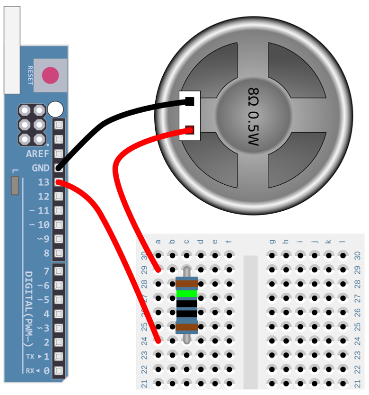


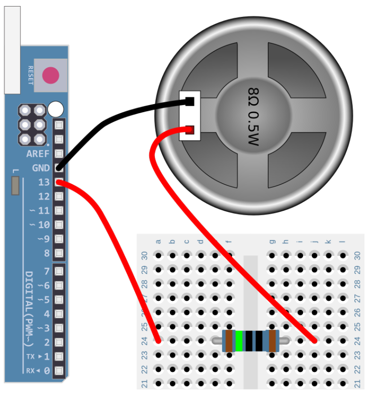

terminal (for component lead)
electrical connection (works like a wire)
signal connections
ground (GND, 0V)
power (+5V)

A typical small breadboard has 30 rows of signal connections, each row divided into two sets of six internally-connected terminals. (Cheaper breadboards only have 5.) Running along all the rows is a ground connection (often marked with a black line) and a power connection (often marked with a red line).

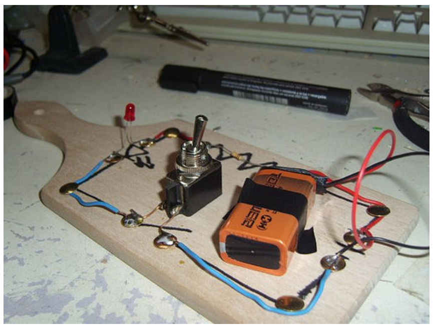
A simple circuit is shown on the left. The loudspeaker connected to a 150 Ω current-limiting resistor. Even such a simple circuit can be built in different ways on a breadboard. Below we show the same circuit built with two different breadboard layouts.





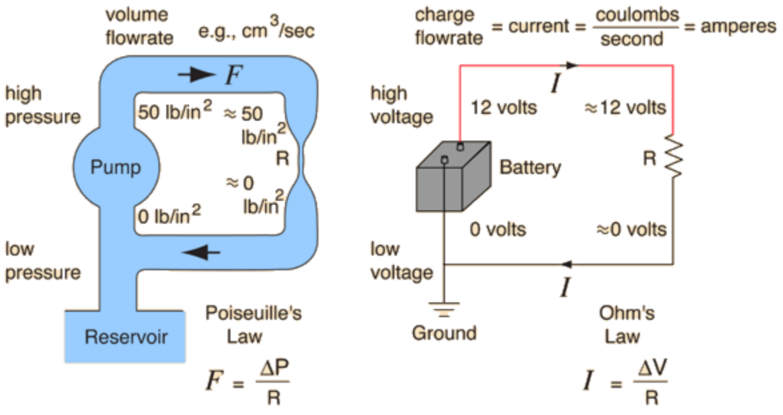


The name "breadboard" comes from the practice of early radio engineers who would nail components to a piece of wood (often actually a cutting board for bread) before soldering wires to them to create a circuit. (The technically-correct name for the ones we use is *solderless breadboard*.)



C Voltage, current and resistance

Electronic circuits are concerned with the *flow* of *electricity* within *wires*. A useful (and surprisingly accurate) analogy for simple circuits is with the flow of water within pipes.



Electrical voltage is equivalent to water pressure. Electrical current is equivalent to water flow rate. Electrical resistance is equivalent to a narrower pipe, opposing the flow of current.

water		*electricity*			
name	*unit*	*name*	*symbol*	*unit*	
molecule	h_2o	electron	e		
volume	m^3	charge	Q	C (Coulombs)	
flow	m^3/s	current	I	A (Amperes)	= Coulombs per second
pressure	Pa	voltage	V	V (Volts)	

The battery is part of the electrical circuit and so the current flowing through the circuit also flows through the battery, just like water flows through the pump in a water circuit.

D Resistors and Ohm’s law

A *resistor* is a component that opposes the flow of current. For the same voltage, as resistance increases the current flowing through the resistor decreases. This is expressed in Ohm’s law,

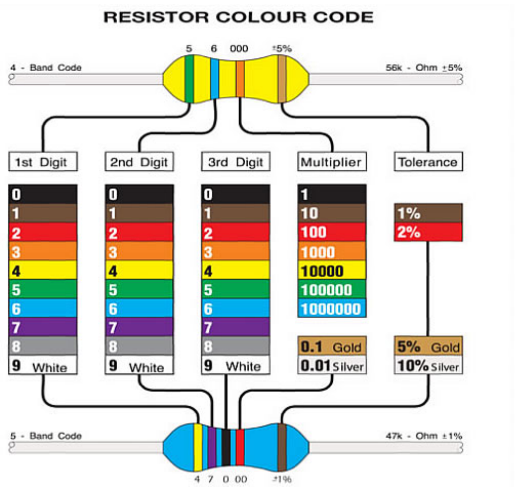
$$V = IR$$

where V , I and R are voltage, current and resistance, respectively. This law can be applied to solve *any* problem involving voltage, current and resistance. For example:

- to determine the current through a series circuit, consider the voltage across is and the total resitance of the circuit; then: $I = V/R$
- to determine the voltage across a series circuit, consider the current flowing through it; then: $V = IR$
- to determine the resistance required to produce a given voltage drop at a given current: $R = V/I$

E Resistor colour code

Resistors are marked with a four (or five) band colour code, each colour corresponding to a digit 0–9. The first two (three) colours represent the first two (three) digits of the value; the next colour is a base-10 exponent multiplier, the final digit encodes the tolerance (the percentage by which the resistor might be above or below the indicated value).



F Microcontroller limits

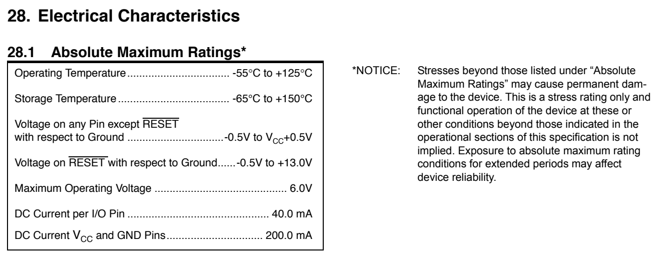
The microcontroller itself behaves like a battery, supplying a small amount of power to external devices. The current flowing through the microcontroller is equal to the sum of the currents flowing through its own GND connections (for internal circuitry) and through the output pins (for external circuitry).

Each output pin can tolerate 40 mA max. To be safe let’s reduce that to 20 mA. The *minimum* resistance required per output pin is therefore:

$$R = V/I = 5/0.02 = 250\,\Omega$$

The current through all outputs must not exceed 300 mA. To be safe let’s reduce that to 150 mA. If you are using all 14 digital output pins then the maximum current per pin is $150/14 = 10.7$ mA and the *minimum* resistance required per output pin is therefore:

$$R = V/I = 5/0.01 = 500\,\Omega$$


28. Electrical Characteristics

28.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

G Limiting the loudspeaker current

Taking too much current from the microcontroller will damage it. According to the data sheet, each pin can provide no more than 40 mA. The loudspeaker has a resistance of 8 Ω. We can calculate the current that will flow when the pin is set to `HIGH` (5 V).

$$\begin{aligned} I &= V/R \\ &= 5/8 \\ &= .625\text{ A} = 625\text{ mA} \end{aligned}$$

This is *far* too much current!

To limit the current to a safe value of 40 mA or less, a resistor connected in series with the loudspeaker is required. The `HIGH` voltage is known (5 V) and the total resistance of the circuit is R + 8.

In a series circuit the current through all components is the same, and is also equal to the current taken from the microcontroller pin. To limit the current through all components to 40 mA or less we can calculate the minimum required value of R + 8 using Ohm's law:

$$\begin{aligned} R + 8 &\geq V/I \\ &\geq 5.0/0.04 \\ &\geq 125\,\Omega \end{aligned}$$

We therefore need $R \geq 117\,\Omega$.

To leave a comfortable margin of safety let's choose a standard resitor value that is about 50% higher. Using a 150 Ω series resistor the current taken from the pin will be at most:

$$\begin{aligned} I &= V/(R+8) \\ &= 5.0/(158) \\ &= 0.0316\,\text{A} = 31.6\,\text{mA} \end{aligned}$$

This will ensure the microcontroller operates safely within its limits.