Introduction to Design (2)

Microcontrollers and Interfacing

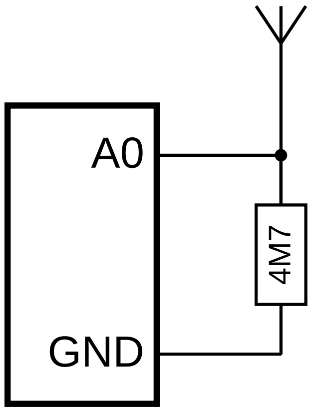
Week 14

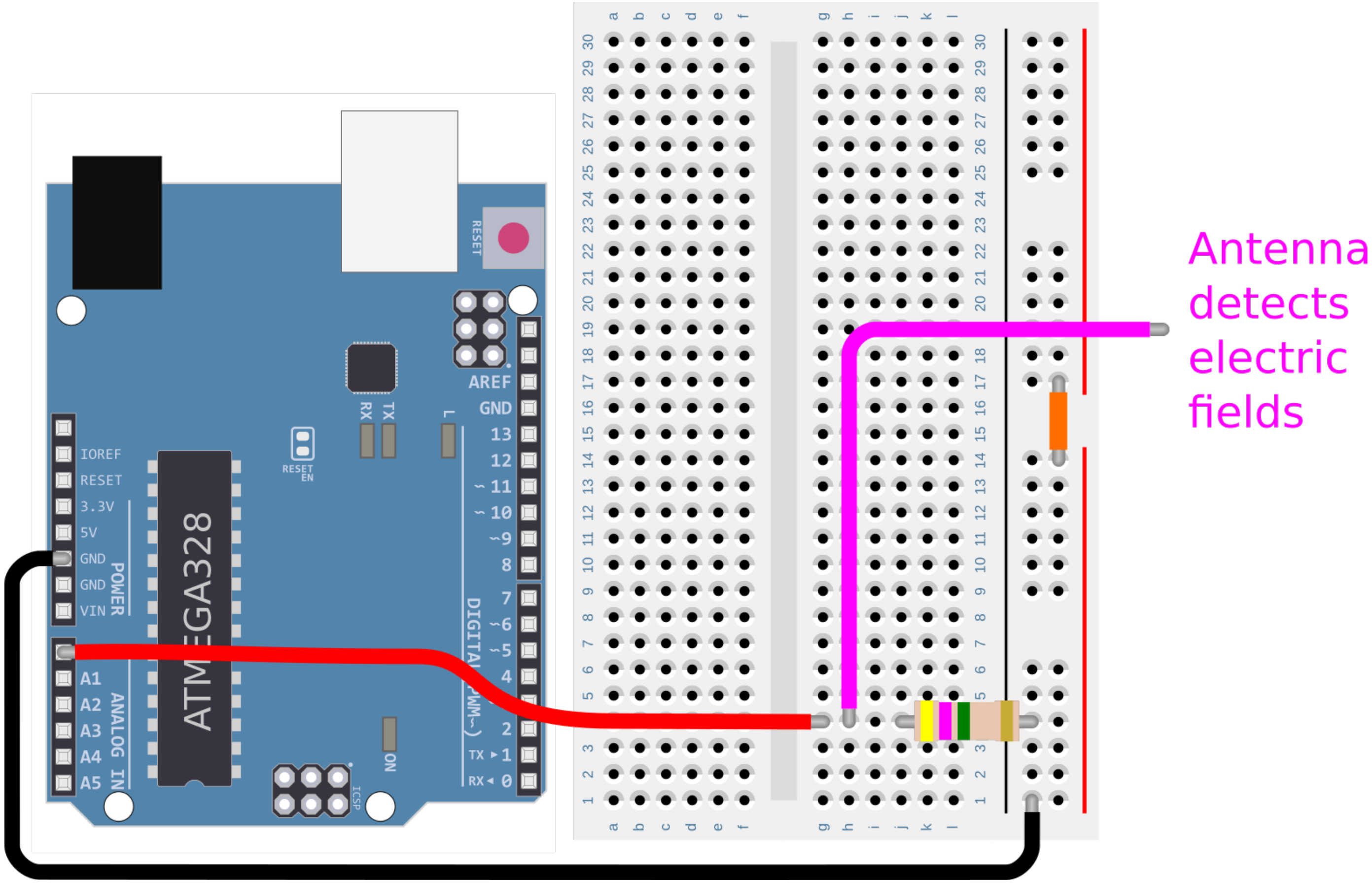
project suggestions

KUAS 京都先端科学大学
KYOTO UNIVERSITY of ADVANCED SCIENCE
Department of Mechanical and Electrical System Engineering

complete project suggestions: easy

1. electrostatic field detector





explore fields near electrical equipment

- or your hand

display results on

- serial plotter [see week 2], or a
- self-calibrating [week 6] bar-graph display [week 8]

complete project suggestions: easy

2. toggle switch from push-button

   microcontrollers are good at making simple components more versatile

   the switches we have are only ‘on’ while they are pressed

   use the microcontroller to convert a push-button into a toggle switch

   a toggle switch alternates between ‘on’ and ‘off’

   - to make it go on and then off, you have to press it twice
   - most light switches behave like this

	push-button switch	*toggle switch*
push	on	on
release	off	on
push	on	off
release	off	off

   begin with a push-button switch [week 11] controlling an LED

   - press to turn on, release to turn off (don’t forget to de-bounce)

   then change the program to flip between on and off when the button is pressed

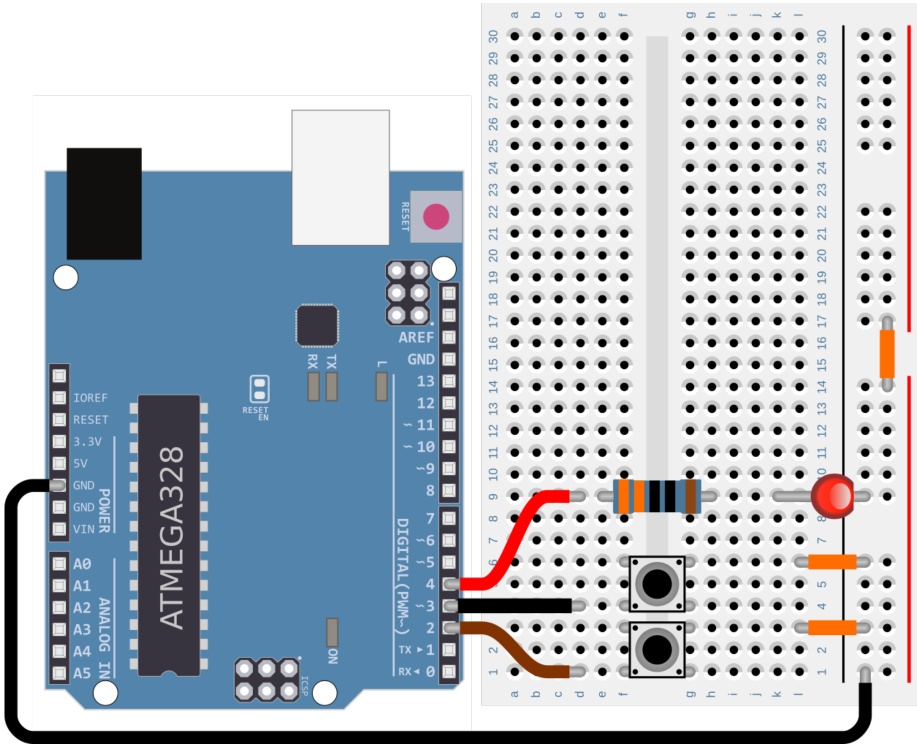
   - press to turn on, release and then press again to turn off

complete project suggestions: easy

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

int state = 0; // light off

void loop(void) {
  if (0 == digitalRead(2)) { // pressed
    state = 1 - state;        // toggle
    digitalWrite(4, state);
    while (0 == digitalRead(2))
      delay(50);
  }
}
```



3. two-way light switches

make a pair of light switches that control one LED

- pressing either switch should turn the light on
- pressing either switch again should turn the light off

(this is how switches at the top and bottom of stairs often work to control one light)

complete project suggestions: easy

4. electronic dice

   use a 7-segment LED [week 10] to display a random value from 1 to 6

   push button to 'roll' the dice

   - don't forget to de-bounce the button!

   for extra realism:

   - display a rapid sequence of random numbers until the dice stops 'rolling'

```
while (buttonPressed) {
  for (int i = 0;  i < 10;  i += 1) {
    int digit = random(1, 7);  // random number between 1 and 6
    displayDigit(digit);
    delay(100);
  }
}
```

   for extra satisfaction (medium difficulty): add a second button and second display

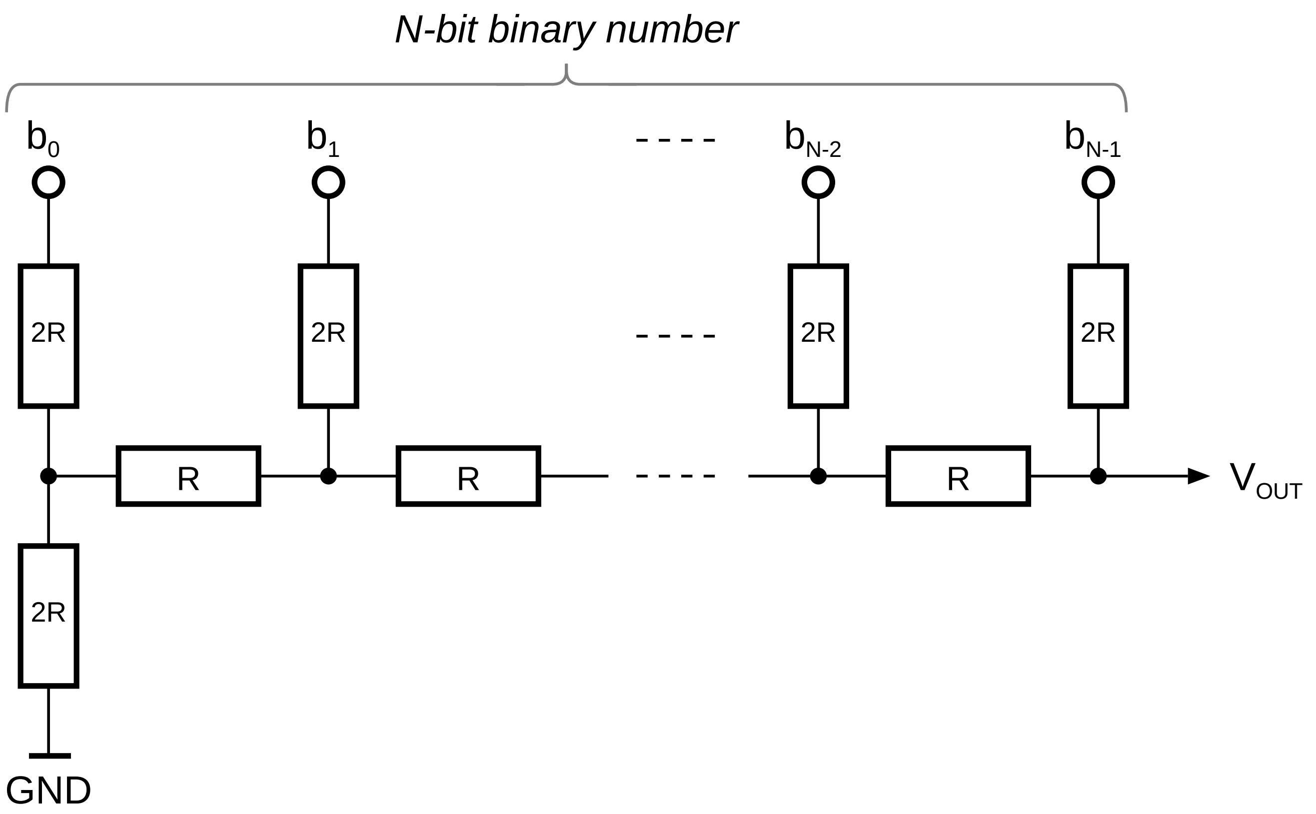
   - you now have two electronic dice!

complete project suggestions: easy

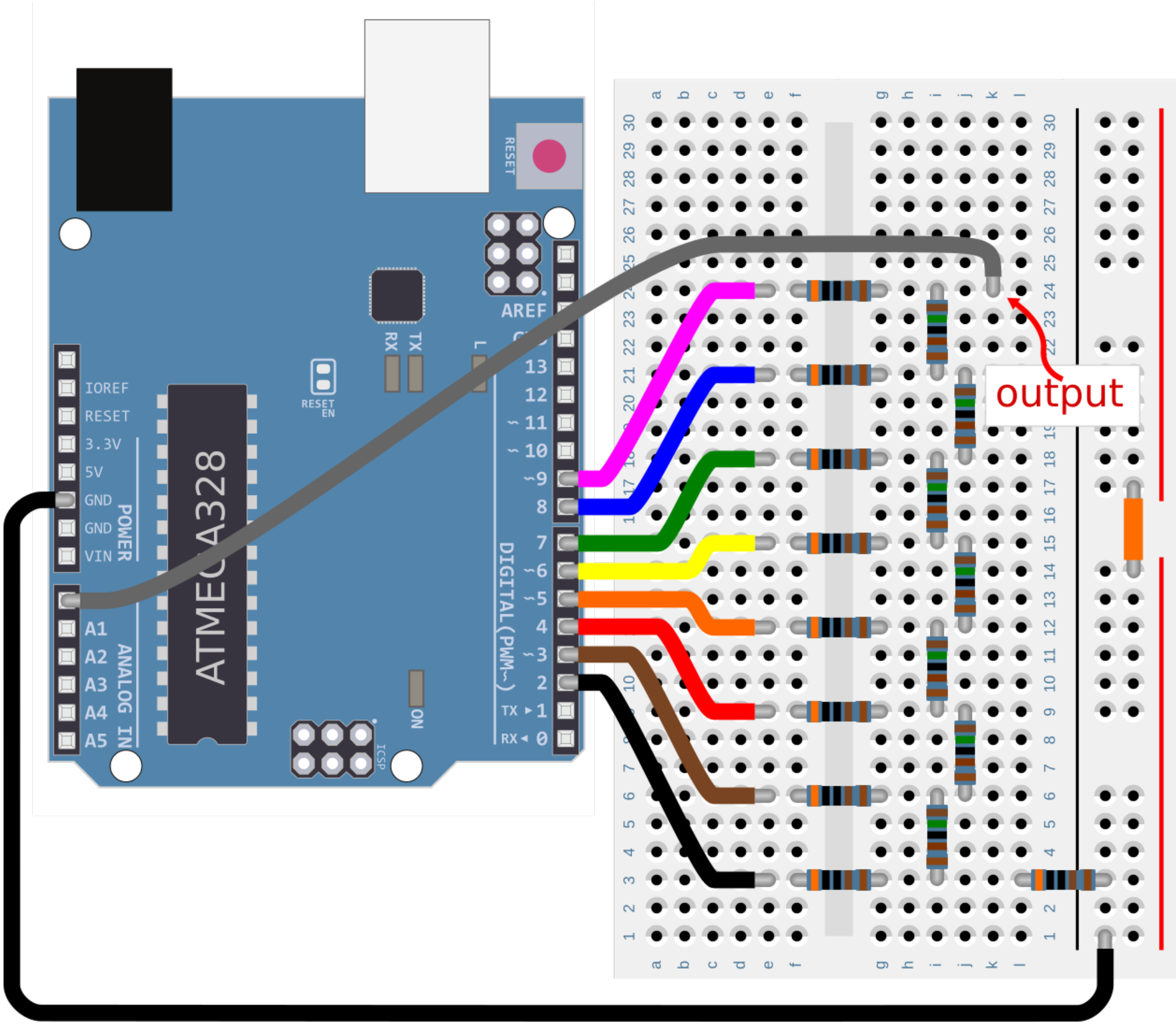
5. digital to analogue converter using resistors

   ladder of resistors with values R and 2R

   - e.g: R = $1.5\,\mathrm{k\Omega}$, 2R = $3\,\mathrm{k\Omega}$
   - build at least 8 bits (your kit has 10 of each of these values)
   - check the accuracy using external SPI ADC



complete project suggestions: easy



complete project suggestions: medium

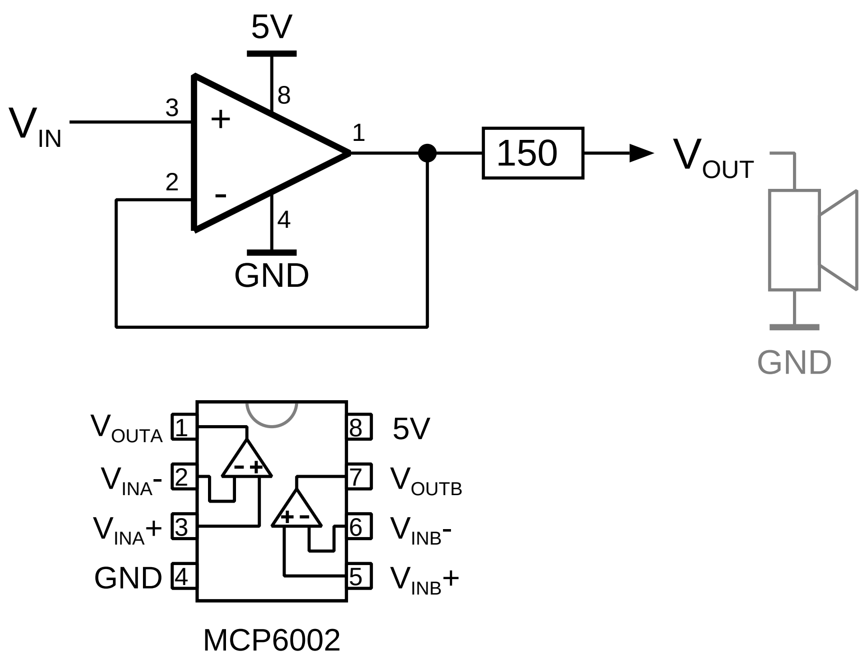
6. build a R-2R DAC with output buffer

   the previous project cannot drive a loudspeaker

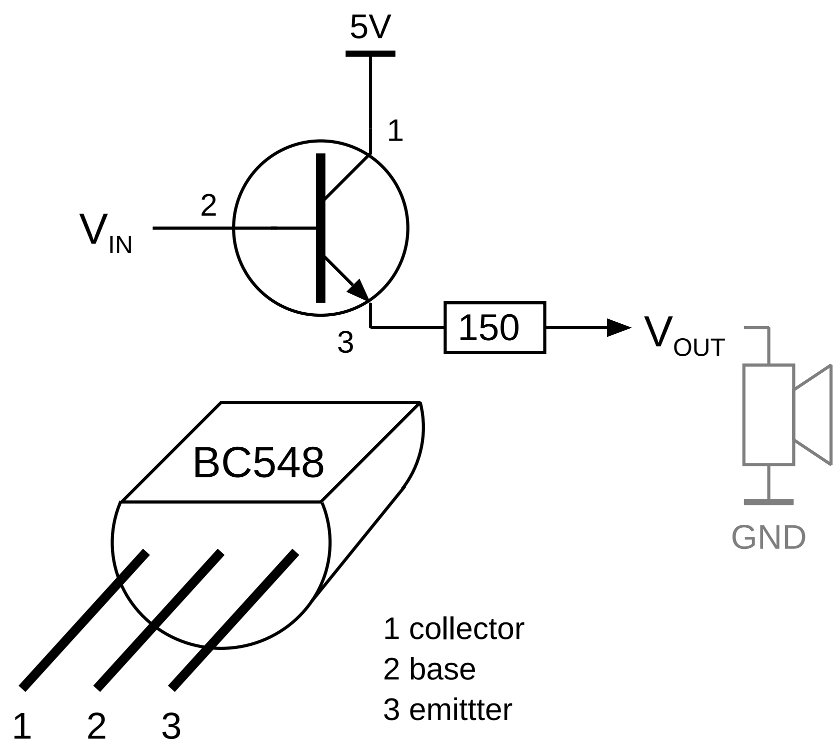
   the loudspeaker resistance is too low and interferes with the the resistor ladder

   add an output buffer with very high impedance to drive (e.g.) a loudspeaker

op-amp buffer

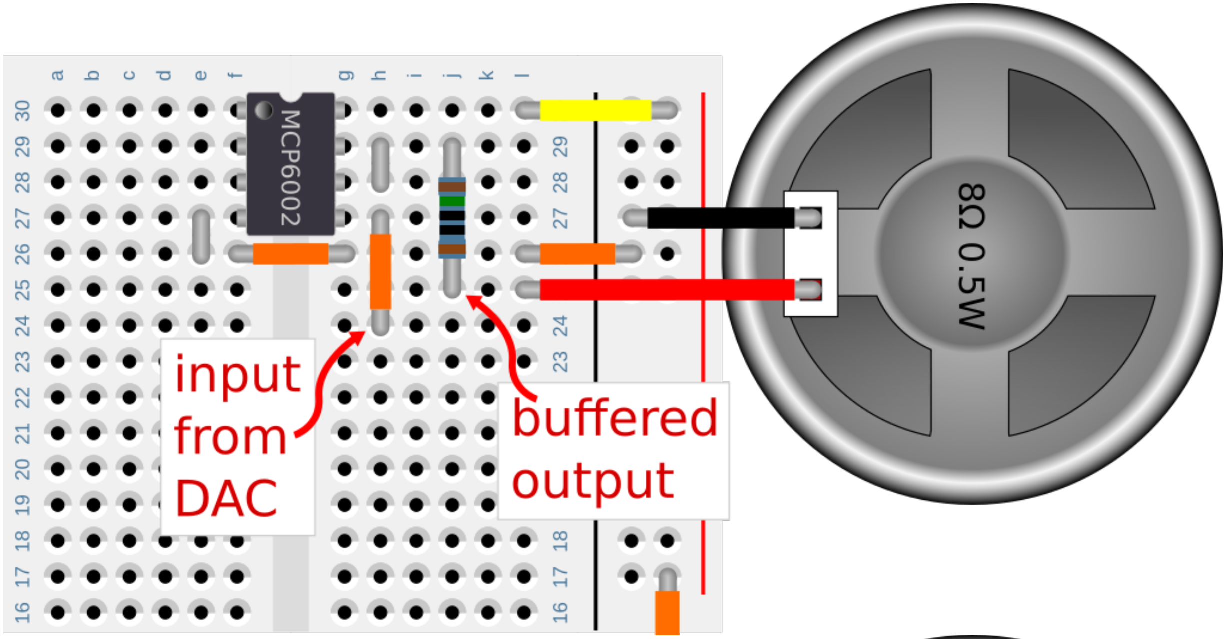


transistor buffer

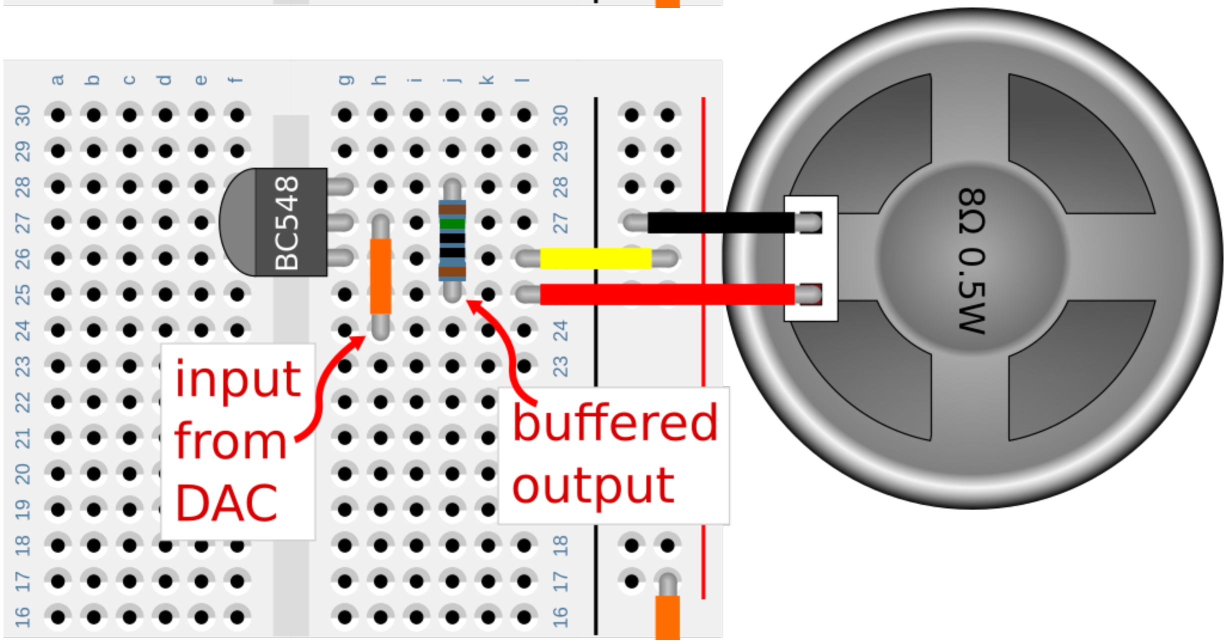


complete project suggestions: medium

op-amp buffer: more complex, better performance



transistor buffer: simpler, worse performance (and input must stay $>0.7\,\mathrm{V}$)



complete project suggestions: medium

7. morse code transmitter

   rebuild the morse code transmitter from earlier

   make it generate tones on the loudspeaker

   control it by typing messages into the serial monitor

   - use `Serial.read()` to read characters sent from the serial monitor
   - ignore characters you don't recognise

complete project suggestions: advanced

8. morse code receiver

   connect some kind of input to the microcontroller

   - push-button (don't forget to de-bounce it)
   - light-dependent resistor (use a threshold with hysteresis to determine on/off)

   read morse code from the input device

   decode the morse code and print the result on the serial monitor

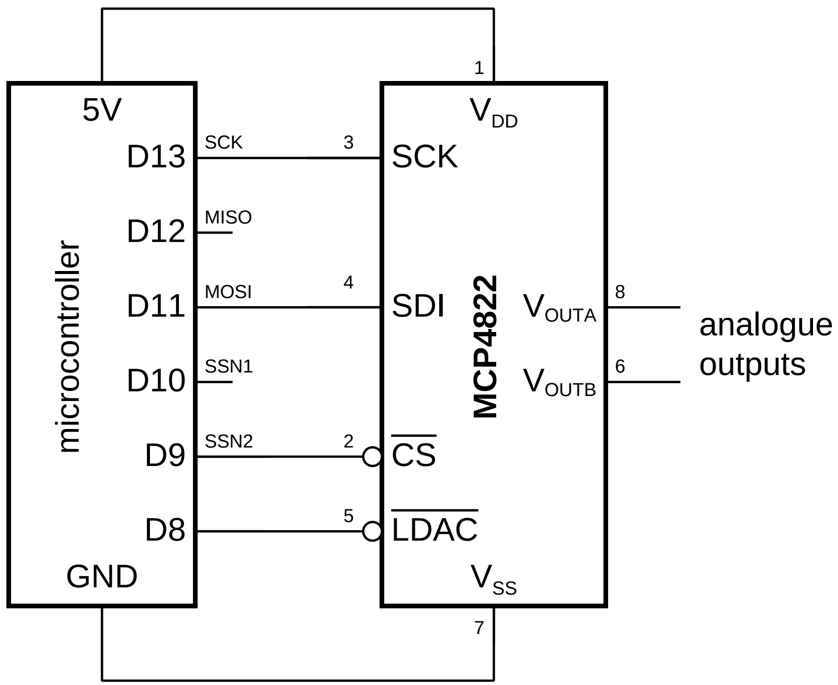
complete project suggestions: advanced

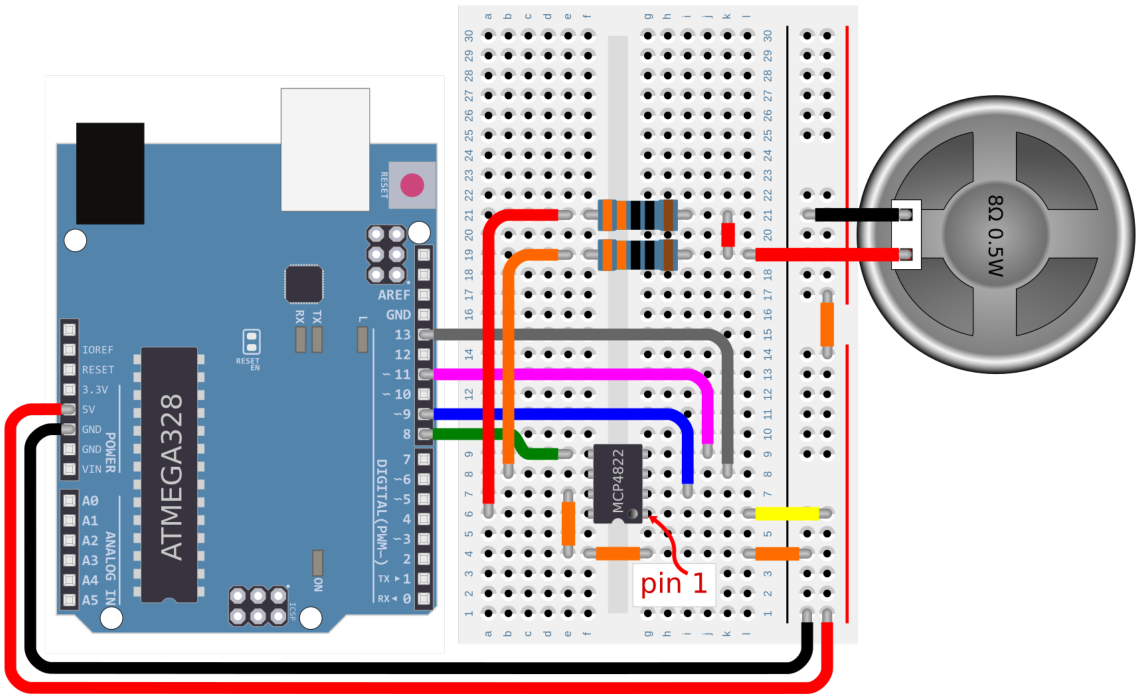
9. SPI DAC

use an external SPI [week 13] DAC (MPC4822) to generate sine waves

this DAC circuit is designed to work alongside the SPI ADC circuit, if desired

- DAC and ADC can share SCK and MOSI, but need separate SSN signals
- make sure they are never enabled at the same time!





(the lab reference material from week 13 shows how to communicate with the DAC)

complete project suggestions: advanced

the file `sine.h` can be downloaded from the course web page

- it contains a table of 4096 integers describing a sine wave
- it also defines a function for you: `sine(N)` $= 2048 + 2047\sin(2\pi N \div 4096)$

use a TimerOne interrupt [week 9] and `sine()` to output a sample every 50 μs

```
#include "sine.h"          // download from the web site
const long rate = 20000; // number of output samples per second
void setup(void) {
  Timer1.initialize(1000000L / rate); // microseconds between samples
  Timer1.attachInterrupt(timer);      // sample generator function
}
volatile unsigned int angle = 0, omega = 1000 * 4096L / rate;
void timer(void) {
  setDAC(sine(angle)); // sine() is defined in "sine.h"
  angle += omega;
}
```

`omega` controls the frequency f of sine wave that is generated

- if r is sample rate and there are 4096 entries in one cycle, `omega` $= f \times 4096 \div r$

complete project suggestions: advanced

10. chord generator

    extend the SPI DAC circuit to play major and minor chords

    use the same sine wave table and `sine()` function to generate three sine waves

    - use *three pairs* of 'phase' and 'omega' variables to scan the table simultaneously
    - add the three sample values together, divide the result by 3, send to DAC
    - for major chords, use: $f_0, f_1 = \frac{5}{4} f_0, f_2 = \frac{3}{2} f_0$
    - for minor chords, use: $f_0, f_1 = \frac{6}{5} f_0, f_2 = \frac{3}{2} f_0$
    - use a push button to swap between major/minor
    - use another push button to change f_0 so that you can play several chords
      - e.g., it might cycle f_0 between 1000 Hz, 1333 Hz, and 1500 Hz
      - then f_1 and f_2 are recalculated based on f_0
      - you should now be able to 'play the blues' on your microcontroller

complete project suggestions: advanced

11. waveform generator

    use an external SPI (or R-2R) DAC to generate

    - sine waves, triangle, or square waves
    - with variable frequency, amplitude

    use Serial interface to control the waveform

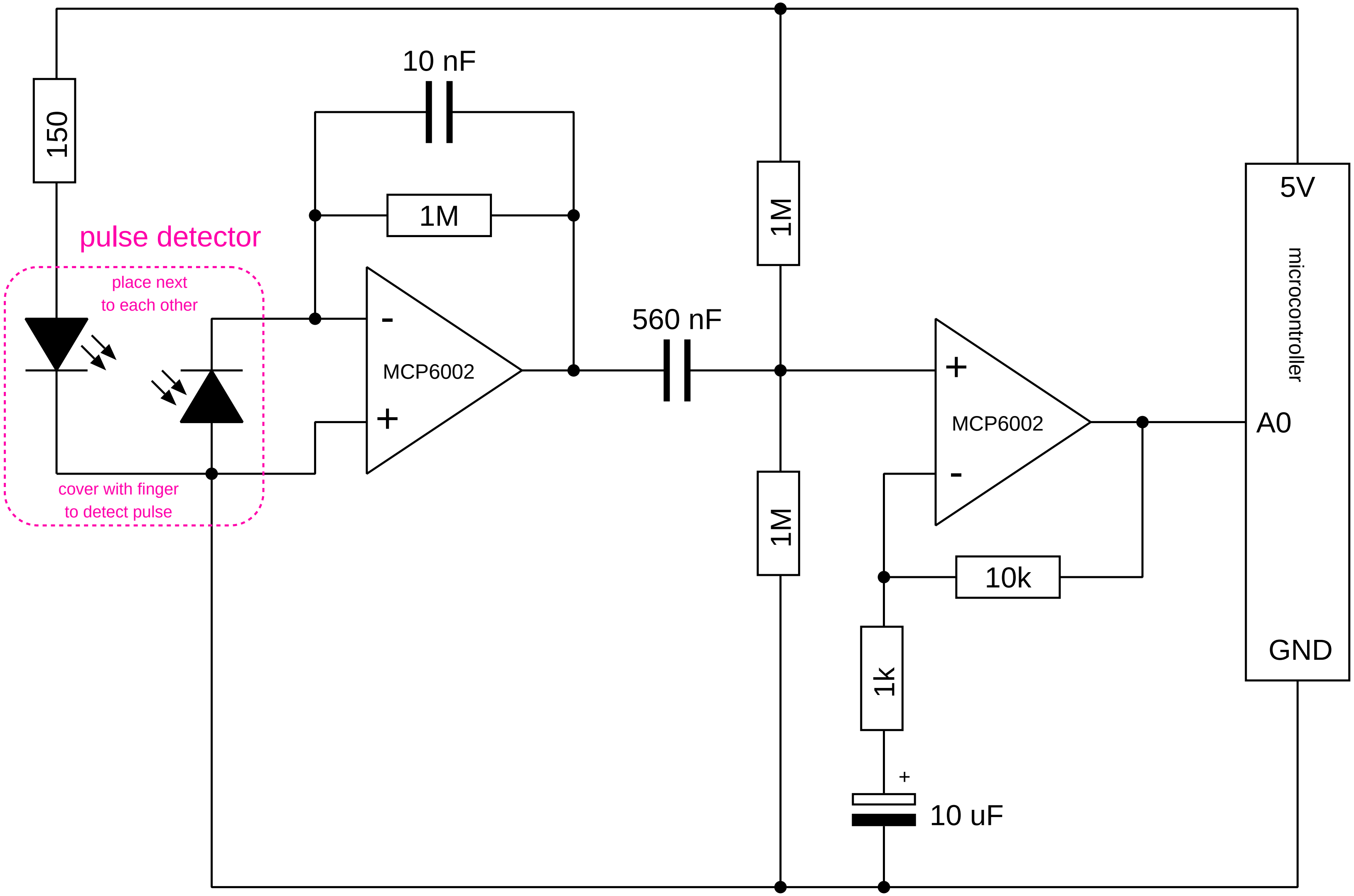
`S 1000 100`	*sine wave, 1000 Hz, 100% amplitude*
`Q 1200 66`	*square wave, 1200 Hz, 66% amplitude*
`T 800 5`	*triangle wave, 800 Hz, 5% amplitude*

```
void loop(void) {
  if (Serial.available() > 0) {
    int c = toupper(Serial.read());
    int f = Serial.parseInt();
    int a = Serial.parseInt();
    if      ('S' == c) playSineWave(f, a);
    else if ('Q' == c) playSquareWave(f, a);
    else if ('T' == c) playTriangleWave(f, a);
    else               disableWave();
  }
}
```

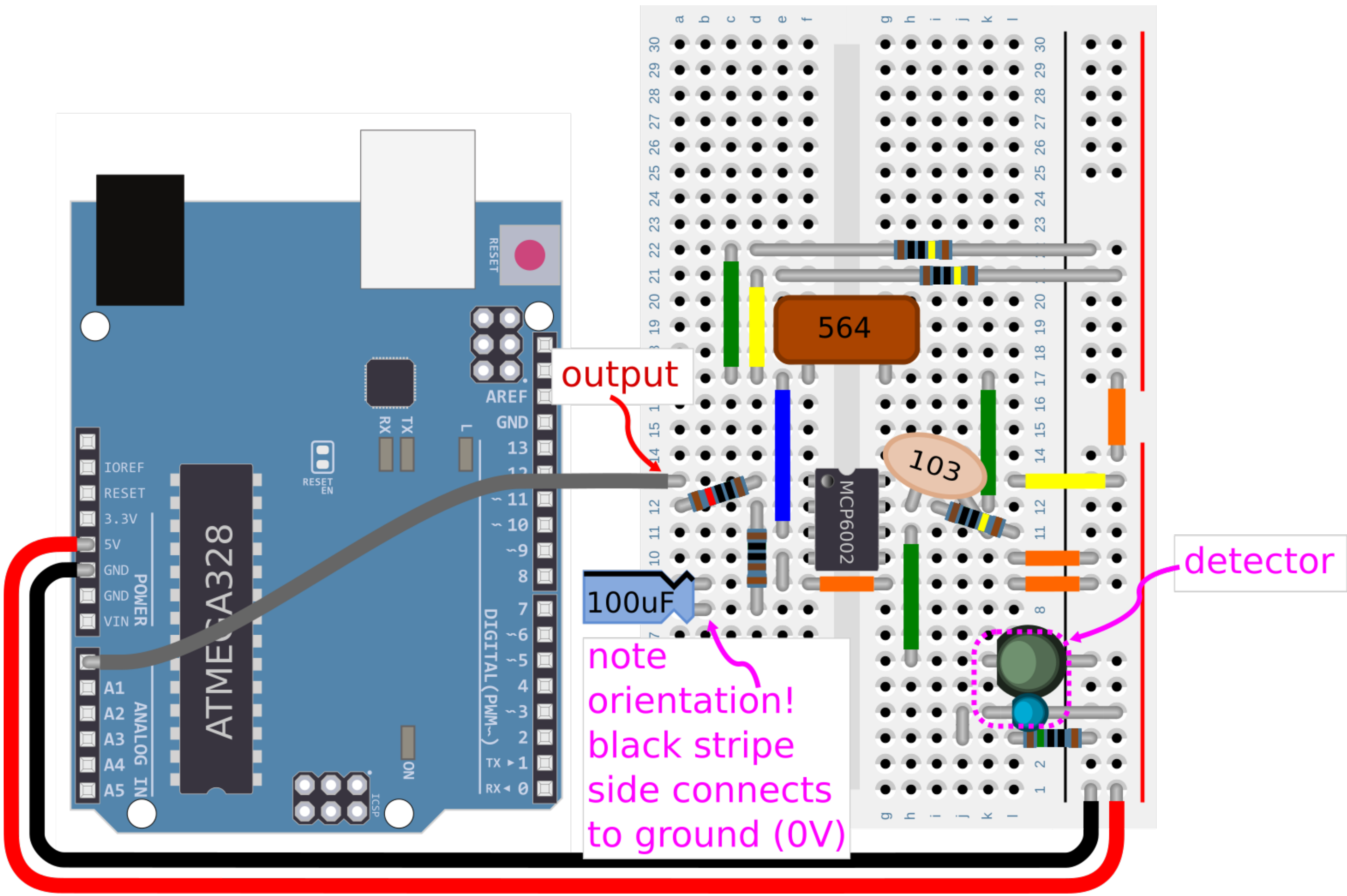
use an analogue input (or SPI ADC) to show the result on the serial plotter

complete project suggestions: advanced

12. pulse rate monitor



complete project suggestions: advanced



challenge: add two 7-segment displays showing pulse rate in beats per minute

complete project suggestions: difficult

if you are able to work with another student, as a team of two...

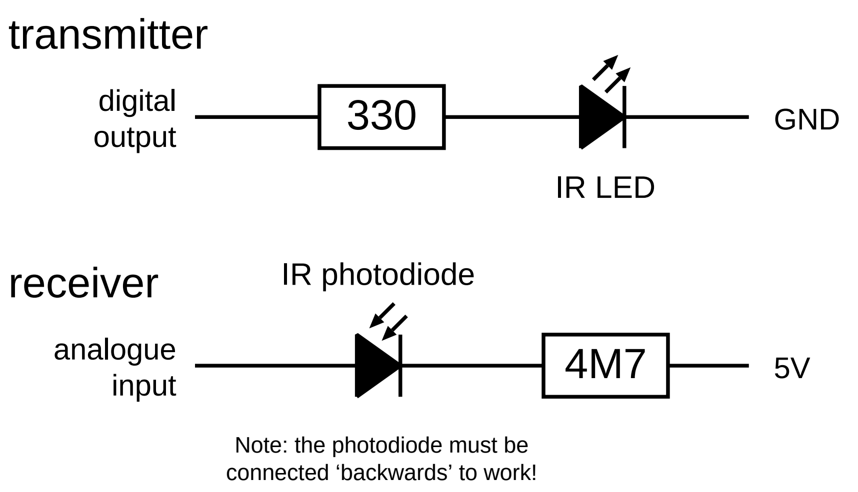
13. infra-red (IR) communication with 'bit-banged' serial protocol

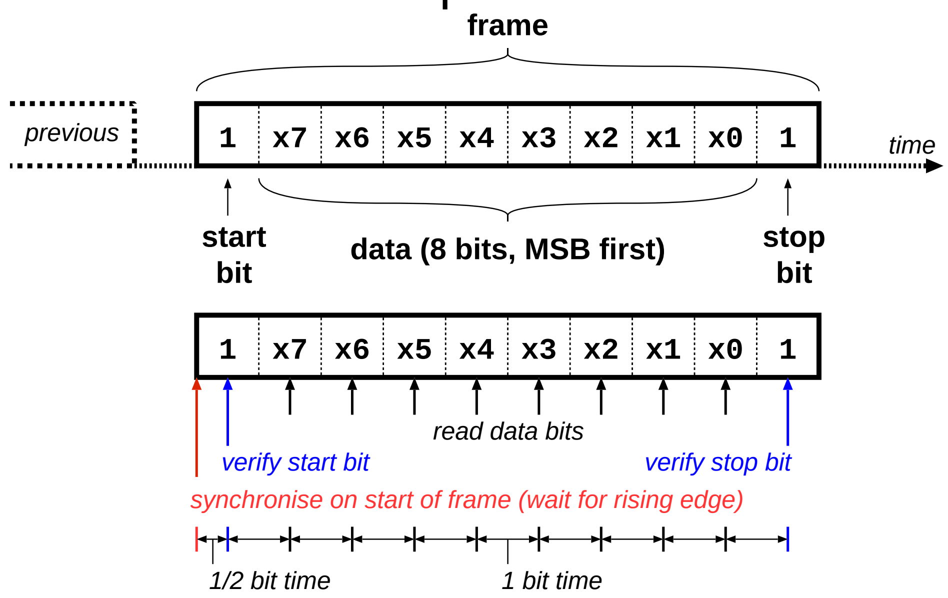
implement your own serial communication (with, e.g., text chat)

- IR LED 'transmitter', IR photo-diode 'receiver'
- explicit encoding/decoding, e.g: 1 start bit, 8 data bits, 1 stop bit

one person makes a transmitter, the other makes a receiver

first step: make sure you can make the photodiode detect pulses from the IR LED!





last step: `Serial.read()` + IR transmit → IR receive + `Serial.write()`

complete project suggestions: flexible

invent your own project!

- use any input/output devices to gather/display information
- perform any function in between

your project

choose a project you feel *confident that you can finish*

- an easy completed project is much better than an ambitious unfinished project
- use one of the suggested projects if you like
  - possibly modified/extended: different input(s), output(s), etc.
- or invent your own project using parts that you have available

ask the instructors about *anything* you are having difficulty with

- time is short, and help is always available
- do not become blocked because you cannot understand something
- use e-mail to ask for help or advice: `ian.piumarta@kuas.ac.jp`

  (or one of the channels in our MS Teams team)

next week

monday:

- project work
- consultation with instructors
- ask questions!

friday:

- project demonstrations
- approximately 5 minutes per project
  - what the project is (especially if you invented it)?
  - why you chose that project?
  - live demonstration (or make a video that you can share)
  - what was difficult?
  - what did you learn?