PAT 451 INTERACTIVE MEDIA DESIGN I

ARDUINO_INTRO

MICROCONTROLLER

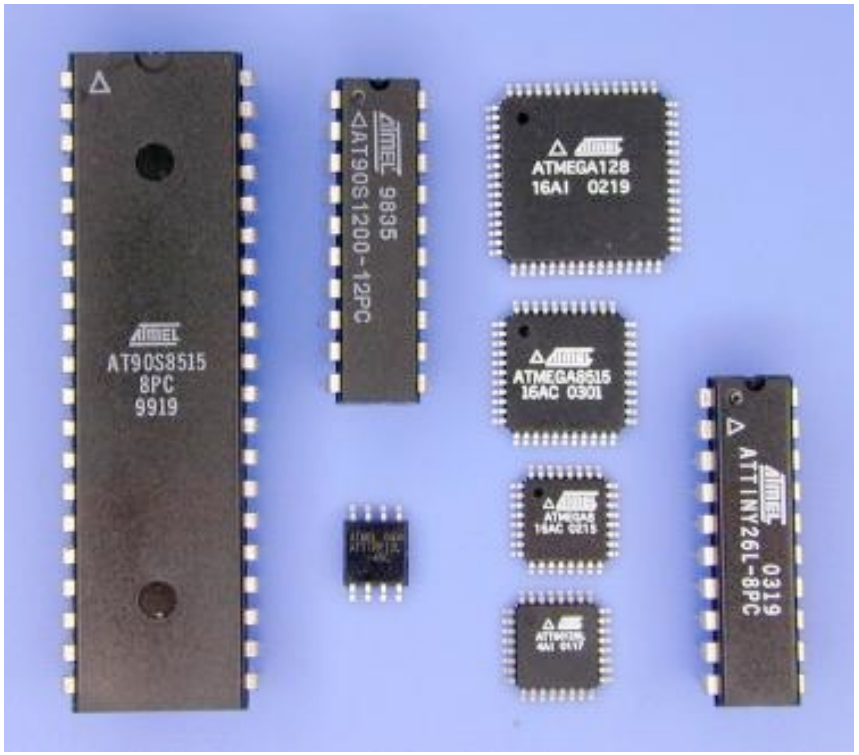
"Computer" on a chip

Different shapes and sizes

Different capabilities

Different architectures

Found in?



MICROCONTROLLER = COMPUTER

CPU

- Instruction set

Clock

- Execute "instructions" at a regular fixed rate

Input

Output

Memory

Storage

MICROCONTROLLER ≠ COMPUTER

	Microcontroller	Computer
Form	Single Integrated Circuit (IC)	Many components connected together
Clock	48 MHz	~2 GHz
Storage	Flash memory: 256kB	Hard disk: 1 TB
Memory	RAM: 32 kB	RAM: 8GB
Bus	32-bit	64 bit
Math	Fixed point	Floating point
Program flow	Single program, runs continuously	Operating System; Multitasking
Peripherals	Low-level: Sensors, switches, LEDs, motors, LCD	High-level: Mouse, printer, camera, sound card

But...

ATARI 800 (1979)

1.8 MHz

8 kB RAM

90 kB Storage (external floppy)

$999.95

RENESAS RA4M1 (2022)

48 MHz

32 kB RAM

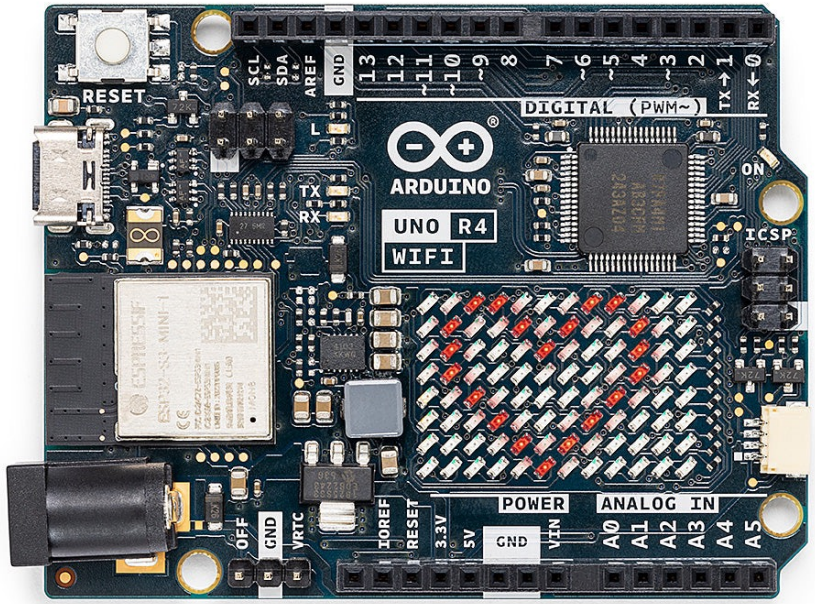
256 kB storage (internal Flash)

~$5.00

ARDUINO

Microcontroller development kit

- convenient hardware access to RA4M1 I/O and programming



"Open hardware" project

- Schematics are developed by a community
- Anyone can build (and sell) one

Brand New (Summer 2023)

Uno R4

- Wifi / Bluetooth chip onboard
- 32-bit ARM architecture processor
- Same footprint and basic features as R3
- USB-C
- LED Matrix
- DAC

PROGRAMMING AN ARDUINO

Write code in C++ on your computer

Efficient compiler

- Designed to translate C++ code into ARM instructions

Compiler generates ARM instructions in a .hex file

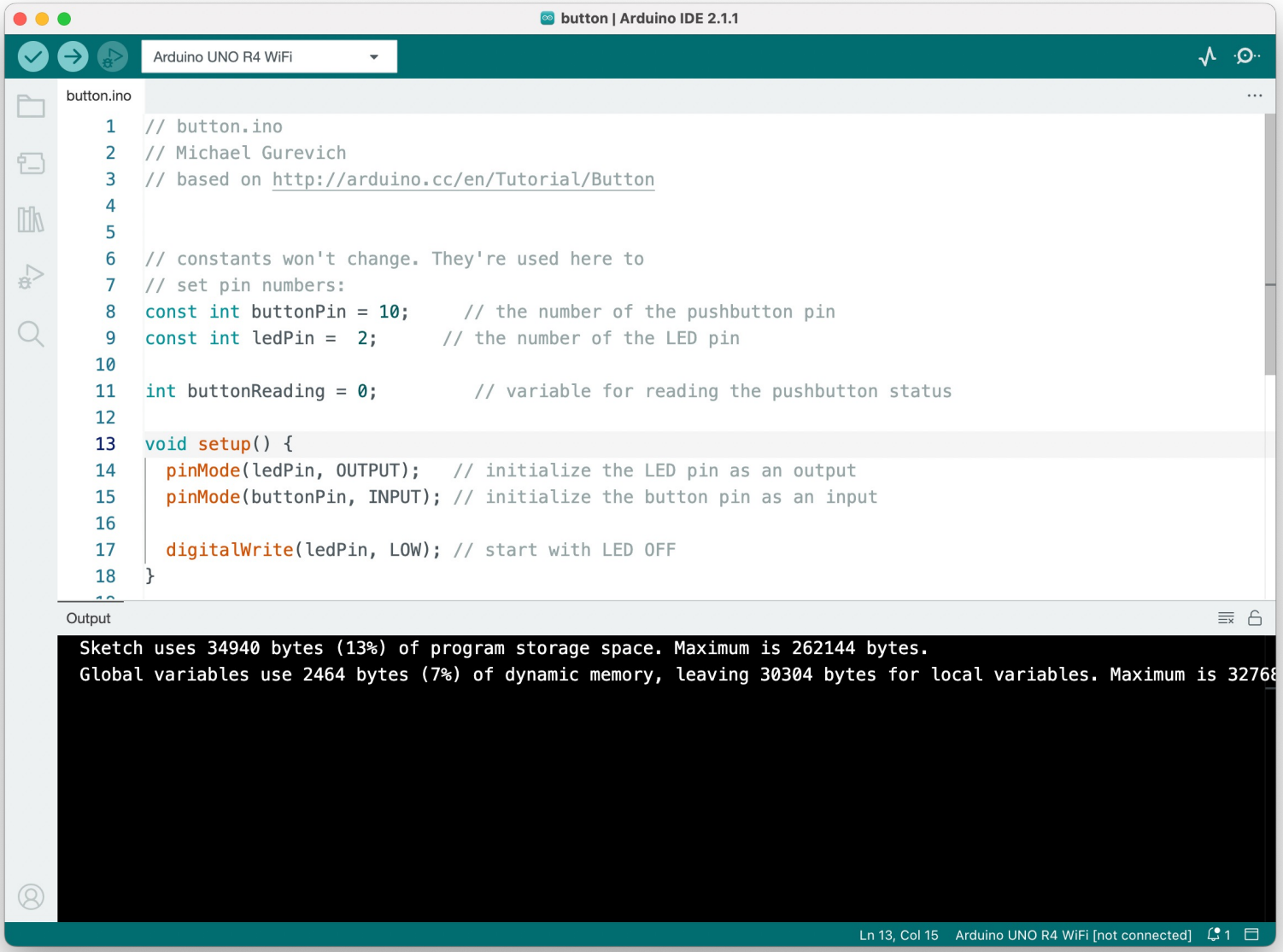
- Actually a map of the ARM's Flash program memory

Another program copies .hex file into the ARM's Flash program memory

- Via USB cable

After this, the Arduino runs on its own – no need to connect it to a computer

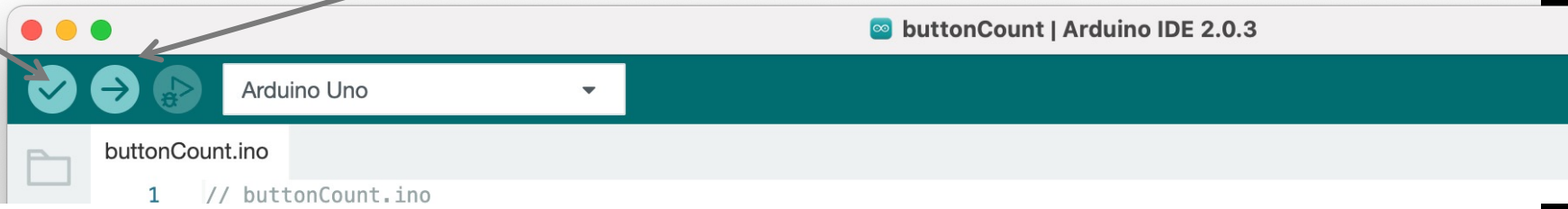
ARDUINO IDE


button | Arduino IDE 2.1.1
Arduino UNO R4 WiFi
button.ino
1 // button.ino
2 // Michael Gurevich
3 // based on http://arduino.cc/en/Tutorial/Button
4
5
6 // constants won't change. They're used here to
7 // set pin numbers:
8 const int buttonPin = 10; // the number of the pushbutton pin
9 const int ledPin = 2; // the number of the LED pin
10
11 int buttonReading = 0; // variable for reading the pushbutton status
12
13 void setup() {
14 pinMode(ledPin, OUTPUT); // initialize the LED pin as an output
15 pinMode(buttonPin, INPUT); // initialize the button pin as an input
16
17 digitalWrite(ledPin, LOW); // start with LED OFF
18 }
Output
Sketch uses 34940 bytes (13%) of program storage space. Maximum is 262144 bytes.
Global variables use 2464 bytes (7%) of dynamic memory, leaving 30304 bytes for local variables. Maximum is 32768
Ln 13, Col 15
Arduino UNO R4 WiFi [not connected]
1

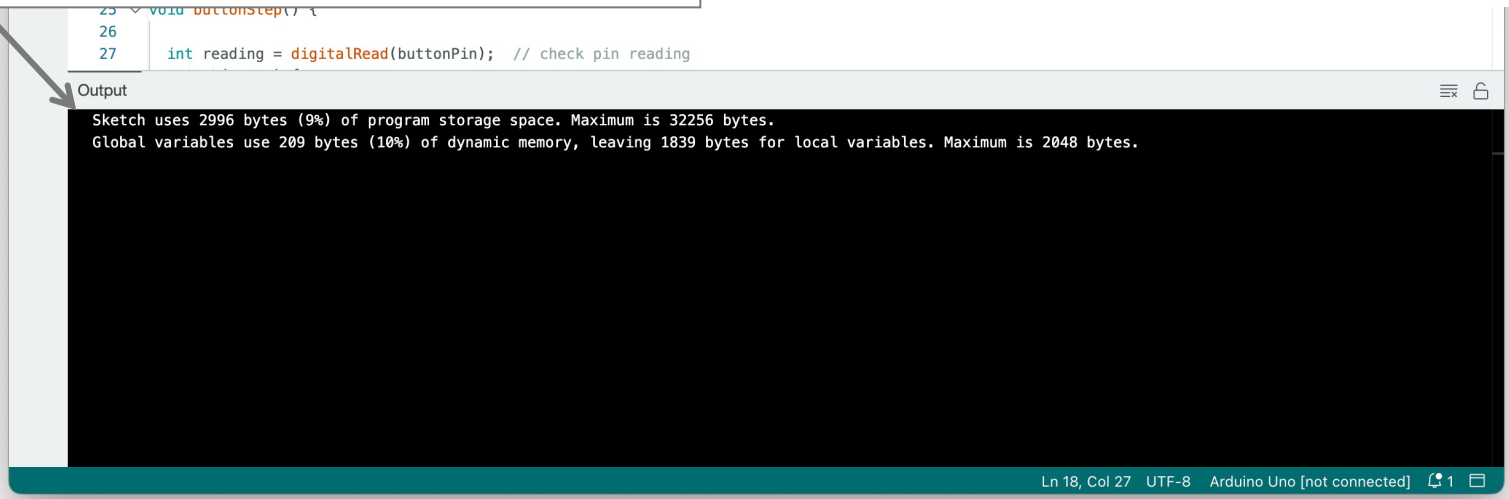
ARDUINO IDE

"Verify"
* Compiles Code

"Upload"
* Sends compiled code to Arduino board



How you know your code was compiled successfully. If not, errors are displayed here.



ARDUINO PROGRAM STRUCTURE

```
void setup() {
  // code executed once, at beginning
}


void loop() {
  // code executed continuously, as fast as possible
}
```

ELECTRICAL REPRESENTATION OF DIGITAL LOGIC

The principle of digital systems is to represent bits as discrete voltage levels.

The "logic level" of the system dictates what the high voltage is (we use 5V).

The low level is ground (0V).

With the microcontroller, we can interact with the outside world in 2 very basic ways:

- Read the voltage of an input
- Set the voltage of an output

We have 2 voltage possibilities: +5V and 0V

DIGITAL I/O

Arduino has 14 pins for doing digital I/O

Pins can be configured as input or output with the `pinmode` function

```
pinMode(int pin,MODE);
        // Set pin (integer) to INPUT or OUTPUT
```

Initially, we may want to use input pins to read the state of switches or buttons

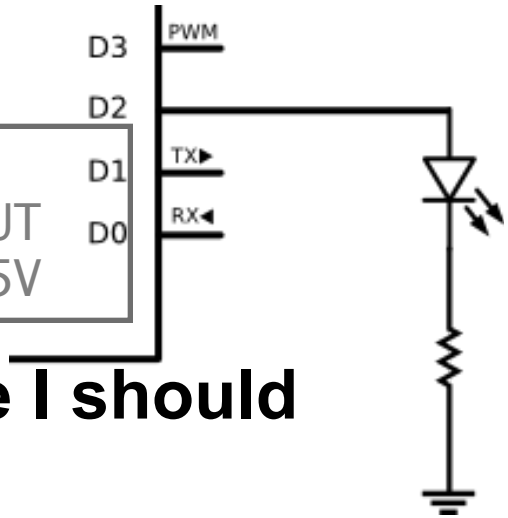
We may want to use output pins to turn on or off LEDs

WRITING AN OUTPUT PIN

When we set an Output pin HIGH, we can think of it as a 5V battery

- It can source a small amount of current (8 mA)
- We can use this to light up an LED:

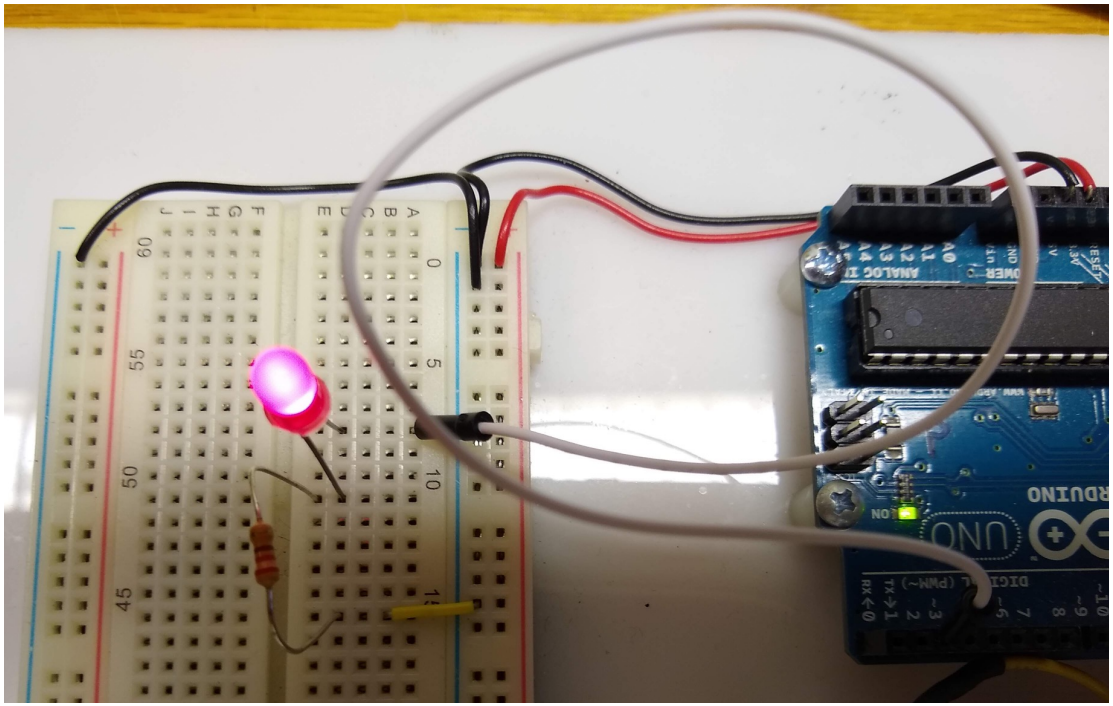
```
int ledPin = 2;
pinMode(ledPin,OUTPUT); // Set pin 2 to OUTPUT
digitalWrite(ledPin,HIGH); // Set pin 2 to +5V
```

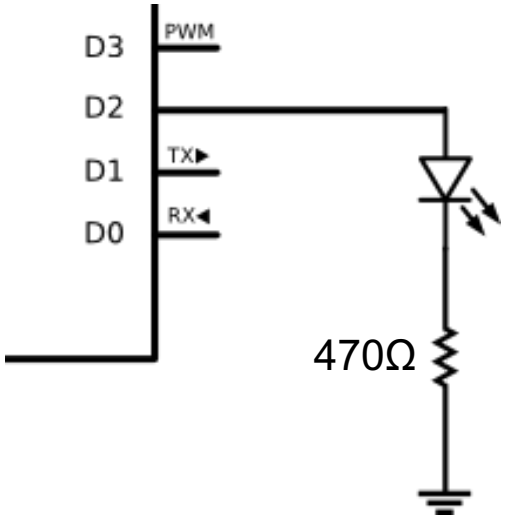


Q: What is the smallest resistor value I should use?

WRITING AN OUTPUT PIN

See **led_source.ino** in accompanying examples.



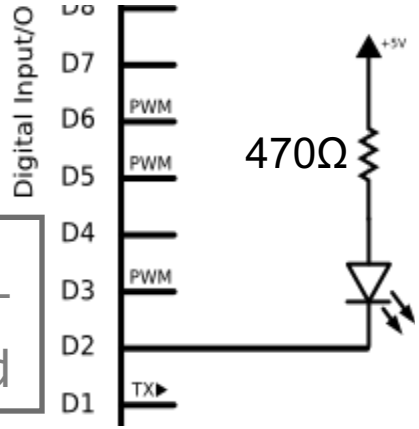



```
int ledPin = 2;
pinMode(ledPin,OUTPUT); // Set pin 2 to OUTPUT
digitalWrite(ledPin,HIGH); // Set pin 2 to +5V
```

WRITING AN OUTPUT PIN

When we set an Output pin LOW, we can think of it as ground

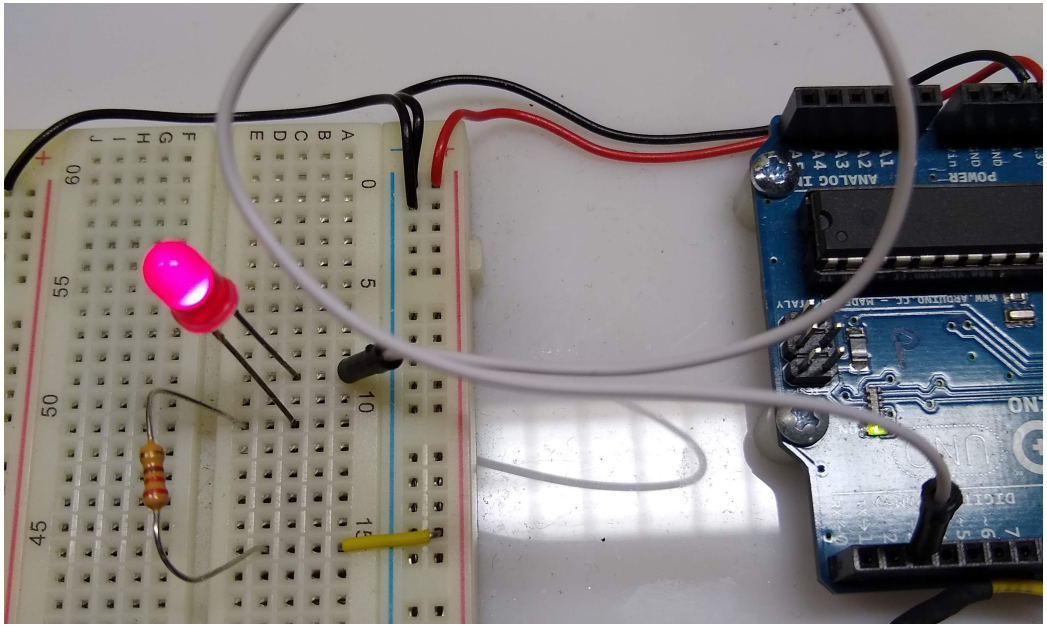
- It can sink a small amount of current (8 mA)
- We can also use this to light up an LED, with a different circuit:

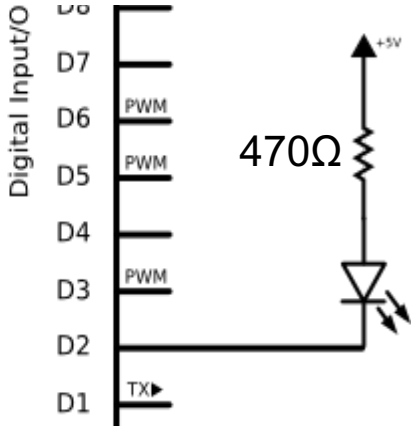



```
int ledPin = 2;
pinMode(ledPin,OUTPUT);   // Set pin 2 to OUTPUT
digitalWrite(ledPin,LOW); // Set pin 2 to ground
```

WRITING AN OUTPUT PIN

See **led_sink.ino** in accompanying examples.

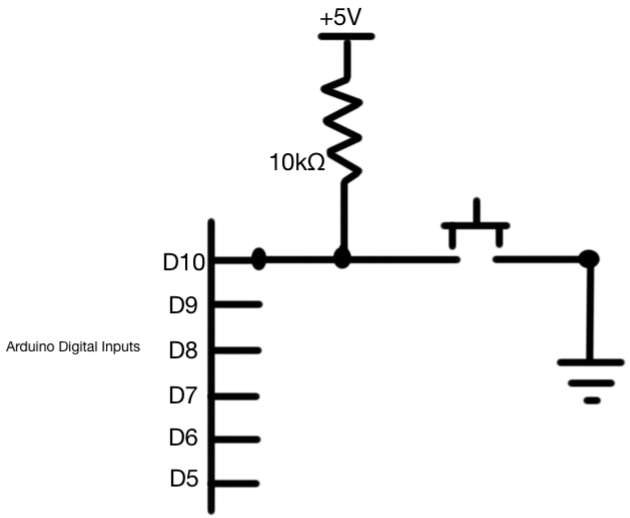




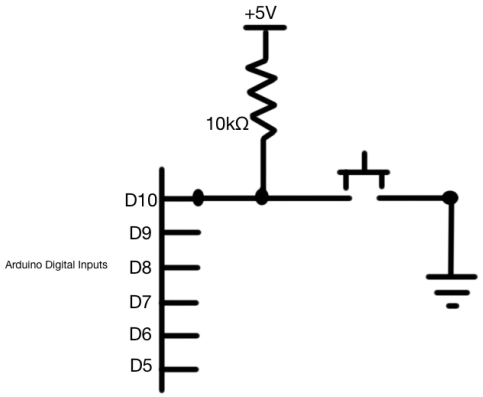

```
int ledPin = 2;
pinMode(ledPin,OUTPUT);    // Set pin 2 to OUTPUT
digitalWrite(ledPin,LOW); // Set pin 2 to ground
```

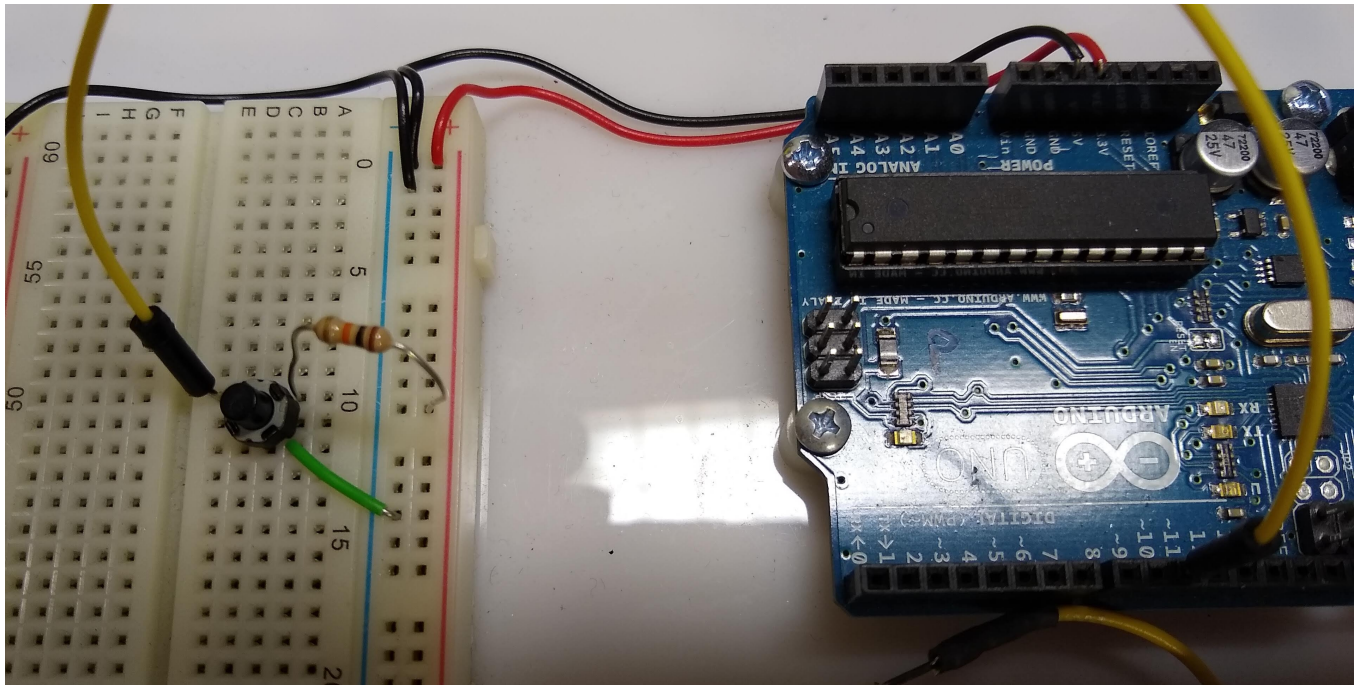
READING AN INPUT PIN

```
int digitalRead(pin)
        // returns the value applied to an input pin
        // either HIGH or LOW
```



READING AN INPUT PIN





See **button.ino** example

PUTTING IT TOGETHER

```
const int buttonPin = 2;      // the number of the button pin
const int ledPin =  13;       // the number of the LED pin
int reading = 0;     // variable for storing button state

void setup() {
   pinMode(ledPin, OUTPUT);   // initialize LED pin as an output
   pinMode(buttonPin, INPUT); // initialize button pin as an input
   digitalWrite(ledPin, LOW); // begin with LED OFF
}

void loop(){
   reading = digitalRead(buttonPin); // read state of the button

   // check if the button is pressed.
   // if it is, the buttonState is LOW
   if (reading == LOW) {
    digitalWrite(ledPin, HIGH); // turn LED on
   } else {
    digitalWrite(ledPin, LOW); // turn LED off
   }
 }
```

SIMPLE TIMING

```
delay(unsigned long t);
        // pause the processor for t milliseconds
        // t must be an integer
```

When you use `delay`, the Arduino program effectively pauses, and other parts of the program aren't accessed.

This can become an issue if you are trying to read button presses while the program is paused.

But in some cases it's an effective way to sequence events or control timing.

EXERCISE 1

Write a program that flashes an LED.

-no input

-on/off cycle times are the same

EXERCISE 2

Write a program that uses a button to start/stop a flashing LED.

- **Initial state: LED off**
- **Press the button while LED off: LED flashing**
- **Press the button while LED is flashing: LED off**

DIGITAL SYSTEMS

Digital here refers to discrete systems

- Can represent a fixed set of values

Most basic quantity is a bit

- Bit: Binary Digit

Binary numbers are simply base-2 numbers

- We normally use base-10 or decimal numbers
  - digits can represent 10 different values (0-9)
  - Where n is the base, there are n different digits from 0 to (n-1)
  - base 10 numbers are constructed as follows:

$$5*10^3 + 7*10^2 + 2*10^1 + 9*10^0$$

$$5729$$

BITS

Binary numbers use base 2

$1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = ??$

1011

Q: What is the maximum base-2 number we can represent with 4 digits?

Q: What is the maximum base-10 number we can represent with 4 digits?

Q: What is a general formula for the maximum number we can represent with n digits?

BITS & BYTES

With n digits, the maximum base-x number you can represent is:

$$x^n - 1$$

We call an 8-digit binary number a byte

We can represent $2^8 = 256$ different values from a range of 0000 0000 to 1111 1111

- 0 to $2^8 - 1$
- 0 to 255

BYTES IN A MICROCONTROLLER

Microcontroller represents data in bytes

- Memory is a series of 8-bit wide containers called registers
- Think of a shelf with 8 slots

When a bit is a 1 (there is a book in the slot on the shelf), we say it is set or high

When a bit is a 0 (the slot is empty), we say it is cleared or low

Digital logic:

- 1 or 0
- high or low
- set or cleared
- on or off