Arduino for Neurobiologists

December 4, 2018

Ofer Mazor, Pavel Gorelik,
Brett Graham, Joe Negri

Instrumentation Core Facility

- Develop new and custom technology for neuroscience research
- Work with neuroscientists in and around HMS
- Provide :
  - engineering/design consultations
  - design & fabrication services
  - workshop space, tools, & training for researchers
- Contact us:

Armenise 406
http://instrumentation.hms.harvard.edu
Ofer Mazor (ofer@hms.harvard.edu)
Pavel Gorelik (pavel_gorelik@hms.harvard.edu)

Course Goals & Methods

Goals:

- Get hands-on experience building small projects with an Arduino
- Feel comfortable building and experimenting with the Arduino in the future

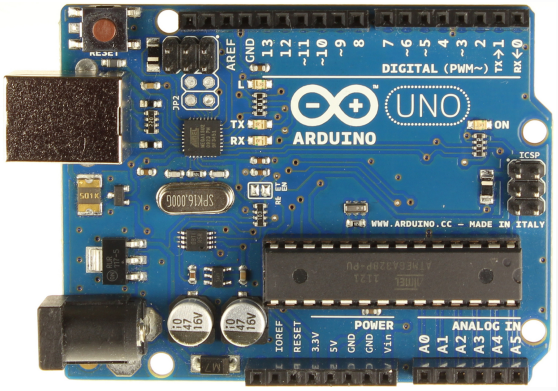
Methods:

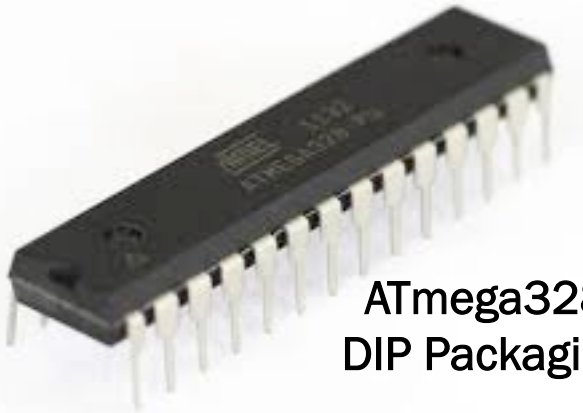
- Learn just enough about Arduinos to get started
- Start building things! Work at your own pace.
- Ask around (instructors, Google) when you get stuck
- Have fun

Today's lecture

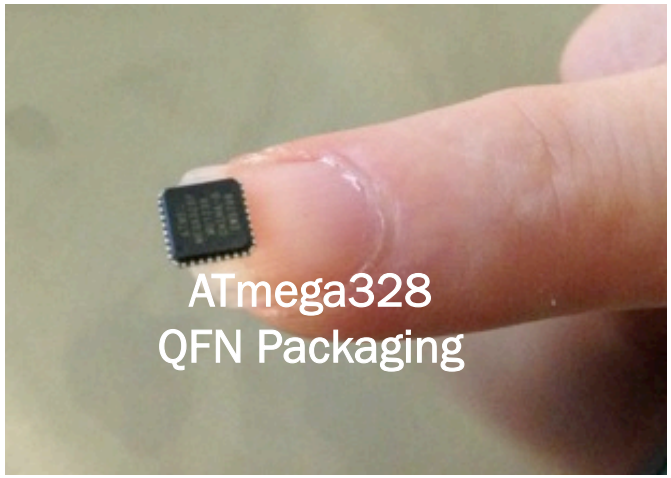
- **Introduction**
  - What is Arduino and when to use it
- **Components of an Arduino project**
  - Capabilities of the Arduino
  - Inputs
  - Outputs
  - Serial communication
  - Programming
- **Demonstration of building a simple project**

Microcontrollers:
The “brains” of Arduinos





ATmega328
DIP Packaging



ATmega328
QFN Packaging

- Small, inexpensive computer on a single microchip
- Limited in power/memory/resources vs. desktop PC
- Runs one program (no OS)
- Requires external hardware to program and integrate into new designs

Microcontroller Examples

Camera MCU

MCUs for keyboard & trackpad

Microcontrollers: also known as MCUs or embedded devices

Arduinos in custom neurobiology instruments

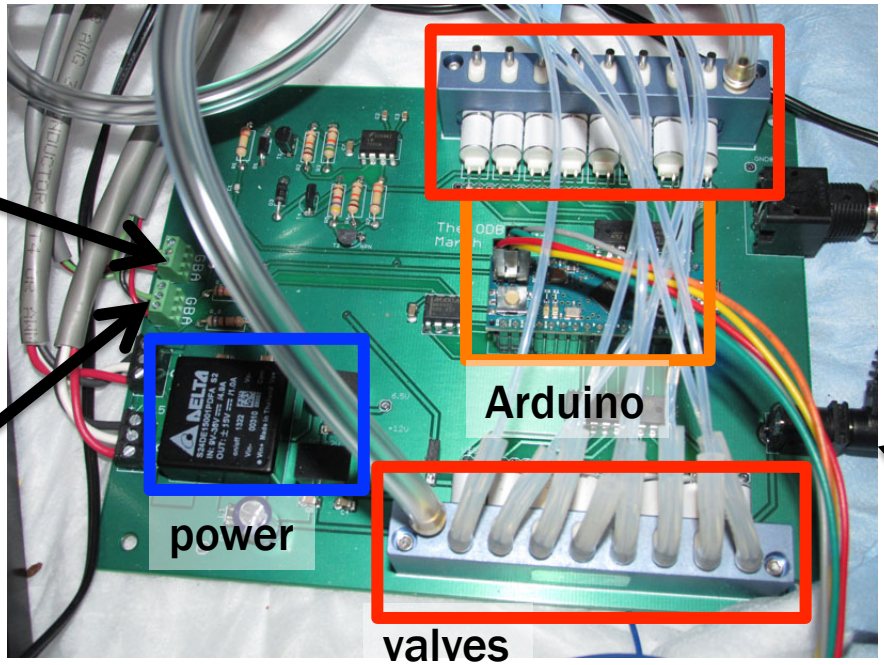
- Stimulus generation
  - olfactometers
  - LED/laser control
- Animal behavior
  - detecting animal position
  - delivering rewards (valve for water/juice)
  - delivering tones/shock
- General instrument control
  - timed laser pulses
  - move a mirror/shutter in a microscope
  - communicate with syringe pump, mass flow controller, etc

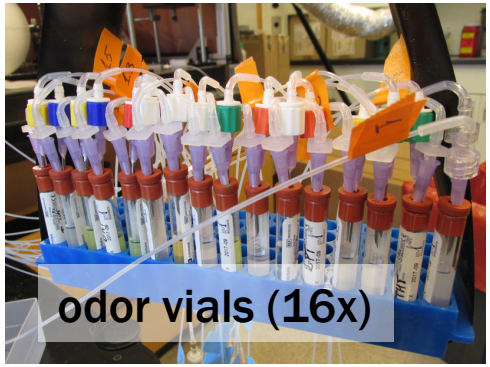
Microcontrollers in custom neurobiology instruments

- 16 channel olfactometer (odor delivery device)
  - 16 different odors
  - adjustable air flow rate
  - USB communication with PC

mass flow controller

PC



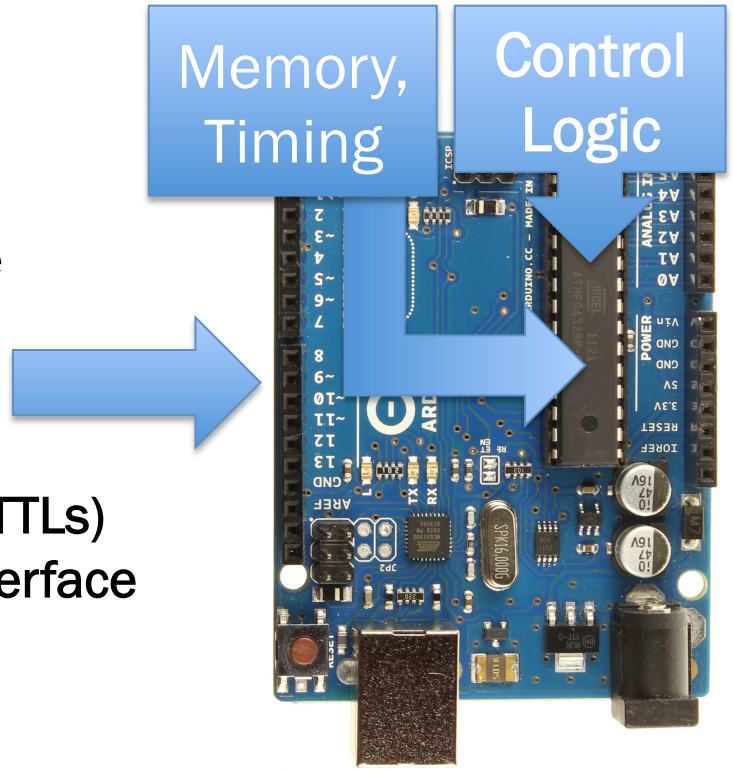


timing signals to NI acquisition board

Architecture of a typical project

Inputs:

- Sensors
  - Temperature
  - Position
  - Pressure
  - Light
- Control Signals (TTLs)
- Buttons/User Interface



Outputs:

- Actuators/Transducers
  - Motors
  - LEDs
  - Switches/transistors
  - Speakers
- Timing/Control Pulses
- User Feedback

USB/Serial Communication

- PCs, Lab instruments
- Other digital devices

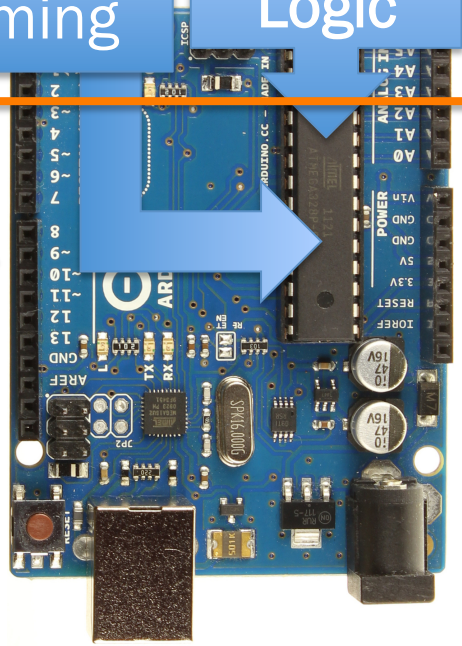
Architecture of a typical project

Inputs:

- Sensors
  - Temperature
  - Position
  - Pressure
  - Light
- Control Signals (TTLs)
- Buttons/User Interface

Memory, Timing

Control Logic



Outputs:

- Actuators/Transducers
  - Motors
  - LEDs
  - Switches/transistors
  - Speakers
- Timing/Control Pulses
- User Feedback

Class 1: Overview

Class 2: Electronics
inputs, outputs, power

Class 3: Arduino Internals
programming, timing

Class 4: Testing & Debugging

USB/Serial Communication

- PCs, Lab instruments
- Other digital devices

MCUs in custom neurobiology instruments

Mouse Nose Poke

- Demo

MCUs in custom neurobiology instruments

Mouse Nose Poke

- Inputs:
  - Nose sensor
- Outputs:
  - Speaker
  - Valve
- Processing (Control software):
  - Controls timing of trials (lick window, inter-trial interval)

MCUs in custom neurobiology instruments

Mouse Nose Poke

- Inputs:
  - Nose sensor
  - Digital signals (for external timing of trials)
- Outputs:
  - Speaker
  - Valve
- Processing (Control software):
  - Controls timing of trials (lick window, inter-trial interval)
  - Logs correct and incorrect responses, response time
  - Sends logs to PC
  - Timing parameters can be changed by PC

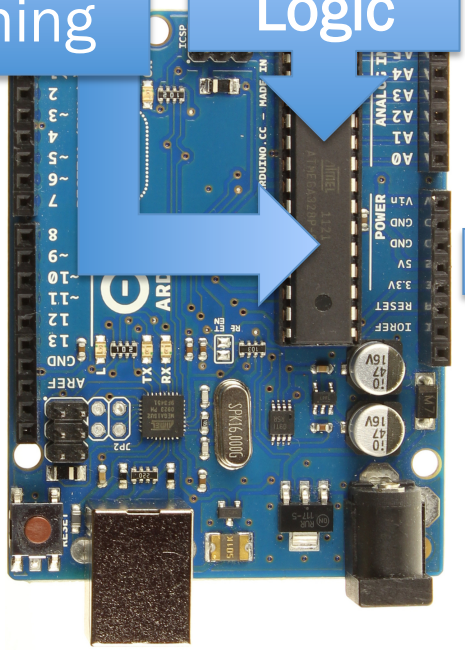
Architecture of a typical project

Inputs:

- Sensors
  - Temperature
  - Position
  - Pressure
  - Light
- Control Signals (TTLs)
- Buttons/User Interface

Memory, Timing

Control Logic



Outputs:

- Actuators/Transducers
  - Motors
  - LEDs
  - Switches/transistors
  - Speakers
- Timing/Control Pulses
- User Feedback

USB/Serial Communication

- PCs, Lab instruments
- Other digital devices

Inputs: Sensors

- Sensors
  - Accelerometers
  - Biometrics
  - Capacitive
  - Current
  - Flex / Force
  - Gyros
  - ID
  - IMU
  - Infrared
  - Light / Imaging
  - Magneto
  - Proximity
  - Radiation
  - Sound
  - Temperature
  - Vernier
  - Weather

HOME / PRODUCT CATEGORIES / SENSORS (203 PRODUCTS)

sort by Most Popular showing 48 per page UPDATE

1 2 3 4 5 ALL

Humidity and Temperature Sensor - RHT03
SEN-10167
$9.95
22

MyoWare Muscle Sensor
SEN-13723
$37.95
1

Flex Sensor 4.5"
SEN-08606
$12.95
3

SparkFun Triple Axis Accelerometer Breakout - ADXL345
SEN-09836
$17.95
7

Force Sensitive Resistor 0.5"
SEN-09375
$6.95
6

SparkFun Sound Detector
SEN-12642
$10.95
7

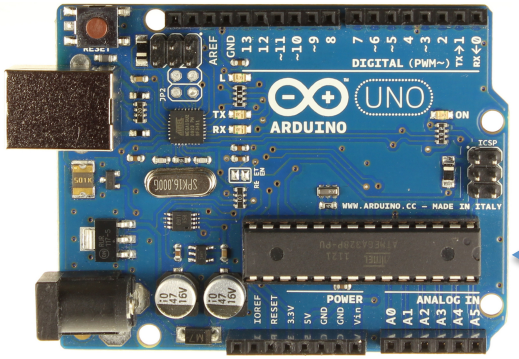
Fingerprint Scanner - TTL (GT-511C3)
SEN-11792
$49.95
8

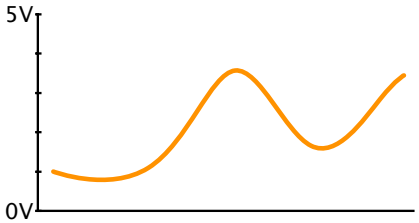
SparkFun 6 Degrees of Freedom IMU Digital Combo Board - ITG3200/ADXL345
SEN-10121
$39.95
4

www.sparkfun.com

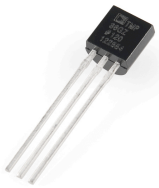
Analog Input

- Arduino has special input pins that accept analog voltage inputs between 0 V and +5 V
  (or 0–3.3V, depending on the model).
- Analog-to-digital conversion:
  - Voltage input is converted to a 10-bit integer
  - ~1000 discrete values can be represented (5mV step size)





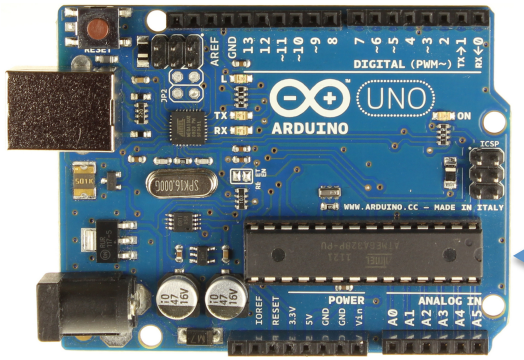
potentiometer

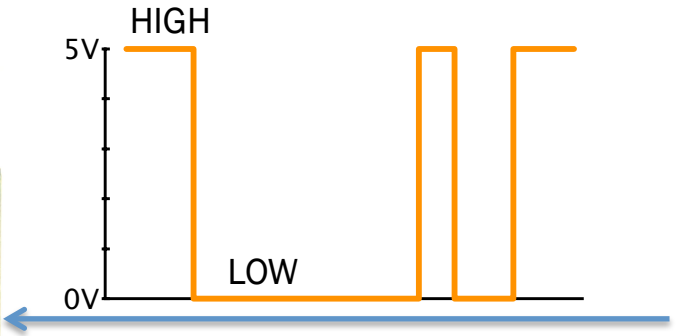


Analog temperature sensor

Digital Input

- Digital input pins:
  - 5 V: HIGH (1) signal
  - 0 V: LOW (0) signal
- Robust to noise: any input > 2.6 V counts as HIGH



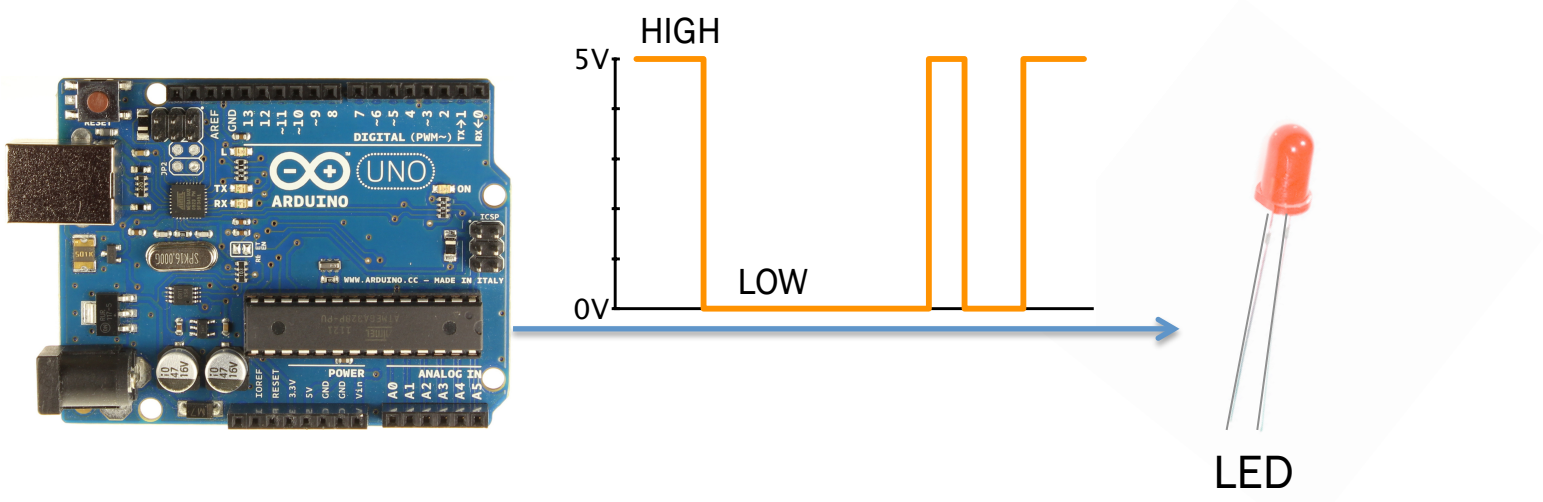


(some models use 3.3V logic instead)

pushubtton

Digital Output

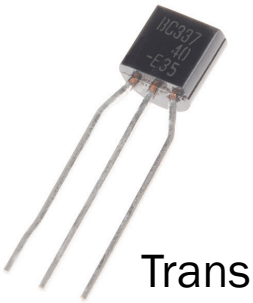
- Digital output pins:
  - 5 V: HIGH (1) signal
  - 0 V: LOW (0) signal



small LEDs and other low-power devices can be driven directly by the Arduino

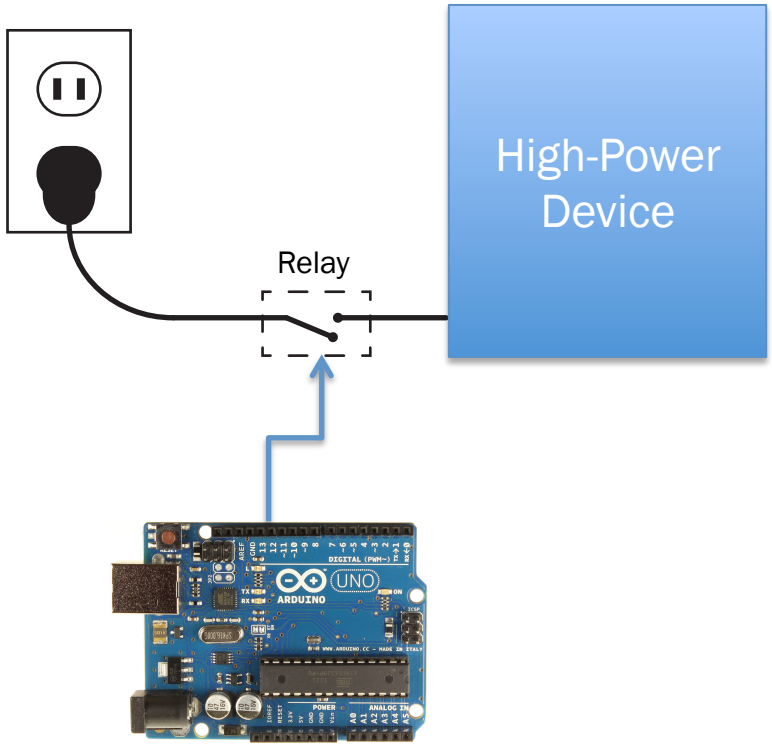
Outputs: Switches

Relays and Transistors act as switches for any powered device



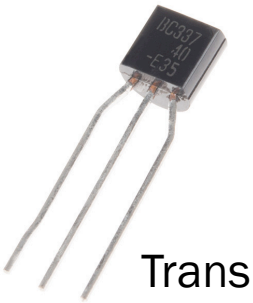
Transistor

Solid State Relay



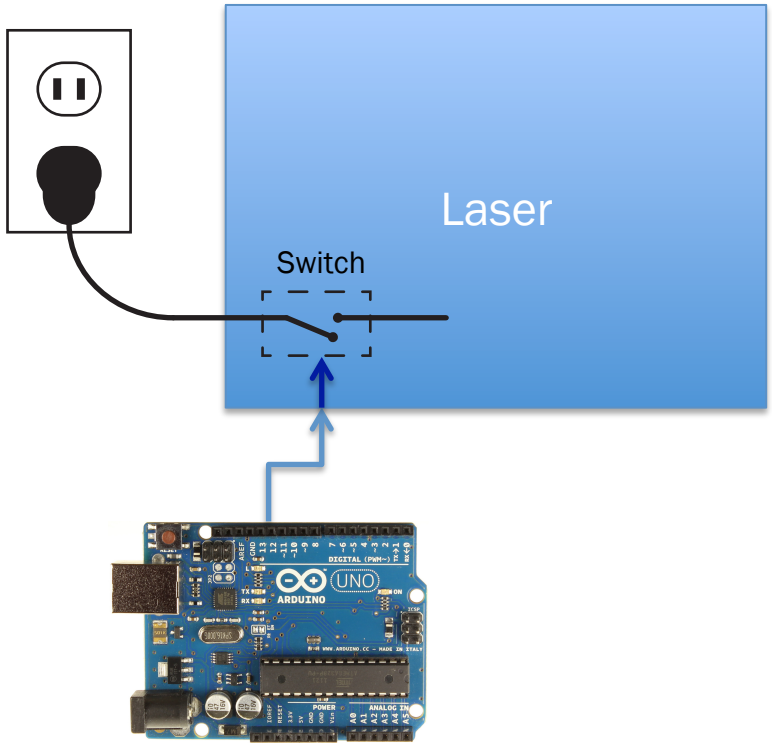
Outputs: Switches

Relays and Transistors act as switches for any powered device



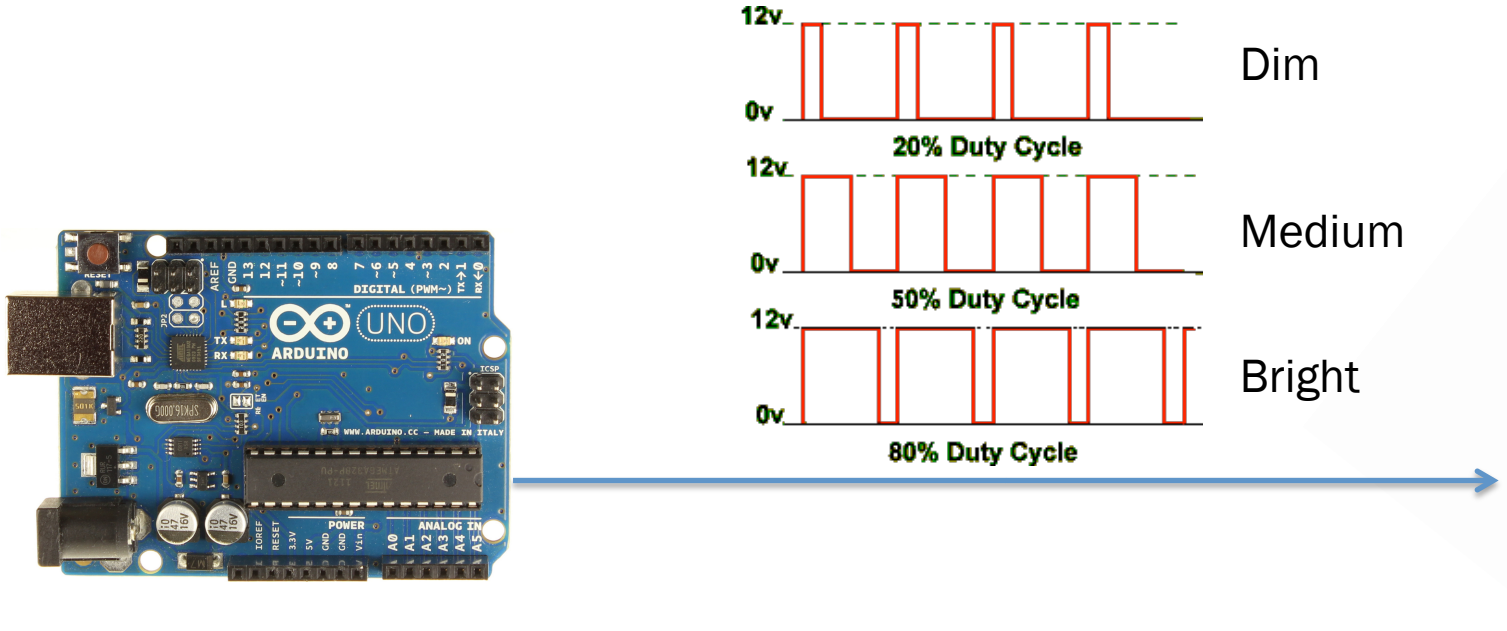
Transistor

Solid State Relay



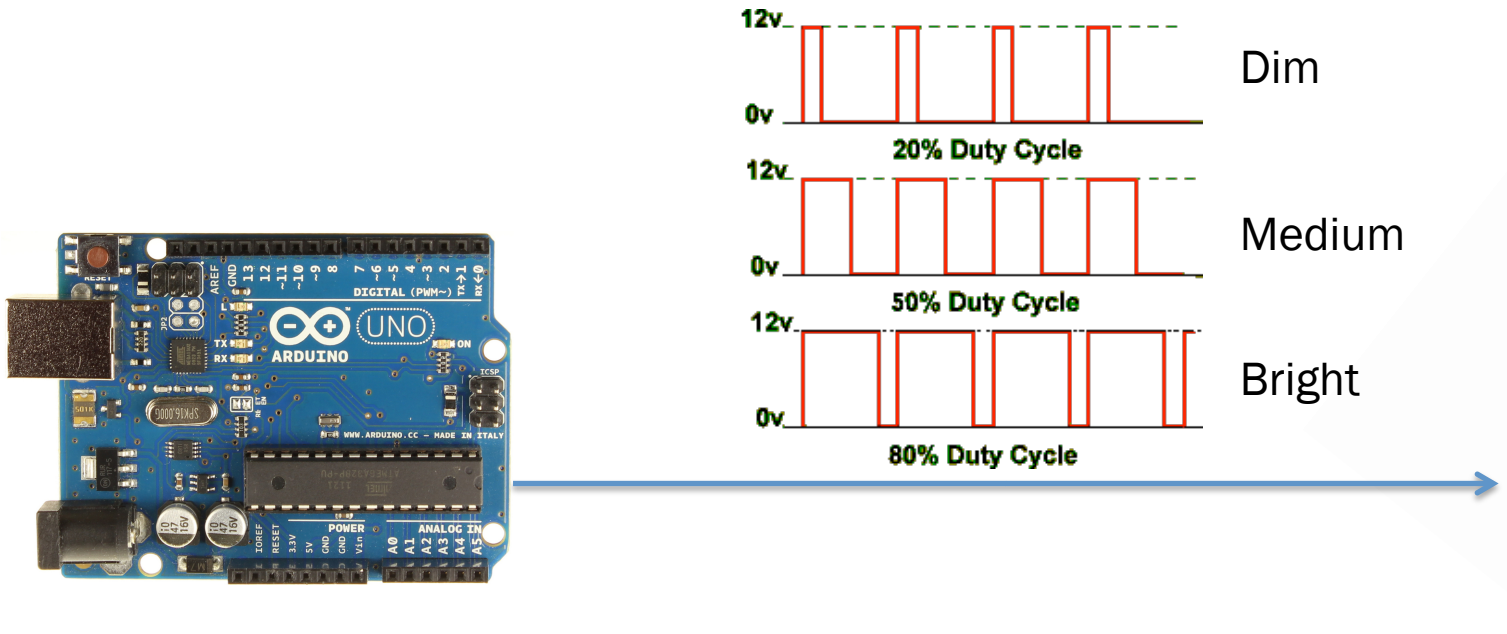
Pseudo-Analog Output: Pulse-Width Modulation (PWM)

- Digital approximation of analog output:
  - Generate digital pulses with constant period
  - Vary the duration of pulse from 0–100% of period



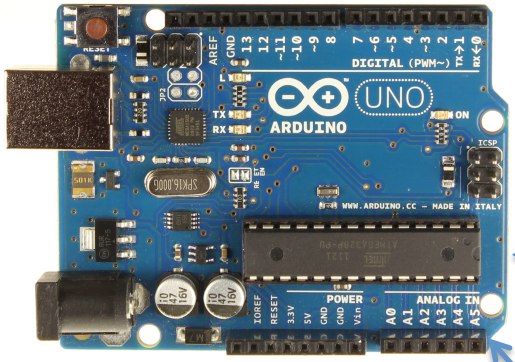
Pseudo-Analog Output: Pulse-Width Modulation (PWM)

- Digital approximation of analog output:
  - Remember, it's not true analog
    - So where does filtering happen?



Controlling "complex" input/output devices

- Serial communication to control instruments



Syringe Pump
Send commands to:
start, stop, set rate, etc

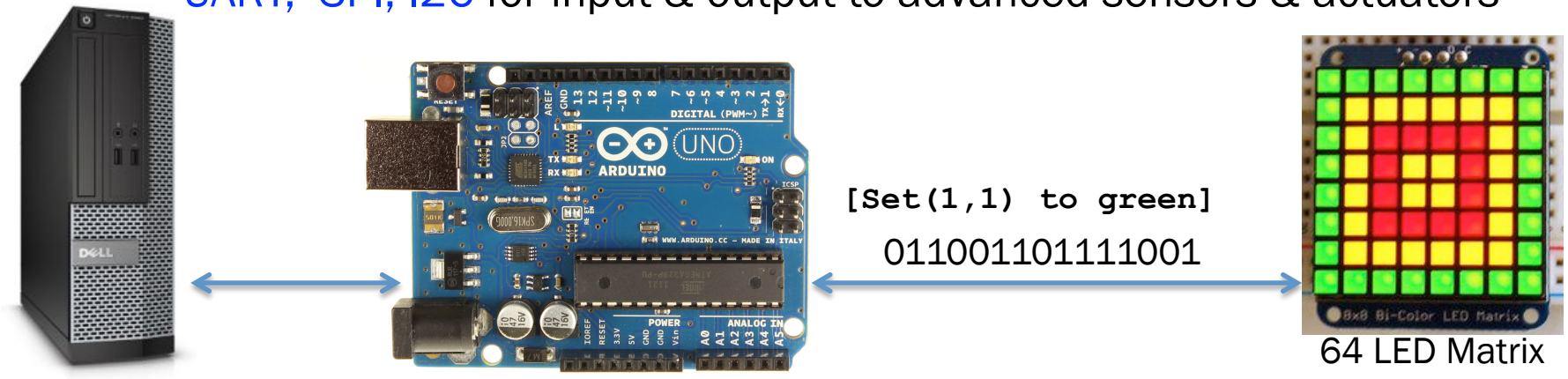
LED Array
64 output channels

Digital Thermometer
4 channels of sensor input

Controlling "complex" input/output devices

Serial Communication

- Bi-directional digital communication protocol
- Sends bits (0 or 1) *serially* over a single data wire
- Devices must agree on a common language for interpreting the raw bit sequence
- Serial protocols:
  - USB communication, typically with a PC
  - UART, SPI, I2C for input & output to advanced sensors & actuators



High-level measurements

What if we want to sense something more complicated:

- How fast is our model organism moving?
- How much food did it eat?
- Which side of the cage is it in?

High-level measurements

What if we want to sense something more complicated:

- **How fast is the mouse running?**

"Art of engineering": Translate high-level feature into a property we can measure.

- Similar translation process for building up high level actions (e.g. deliver a reward).

High-level measurements

- **How fast is the mouse running?**

- **Video tracking** of free-moving animal

  (Arduino can't do this)

- **Measure rotation** of running wheel, ball
  - use rotary encoder for 1 axis of rotation
  - use optic flow sensor (computer mouse) for 2 axes of rotation

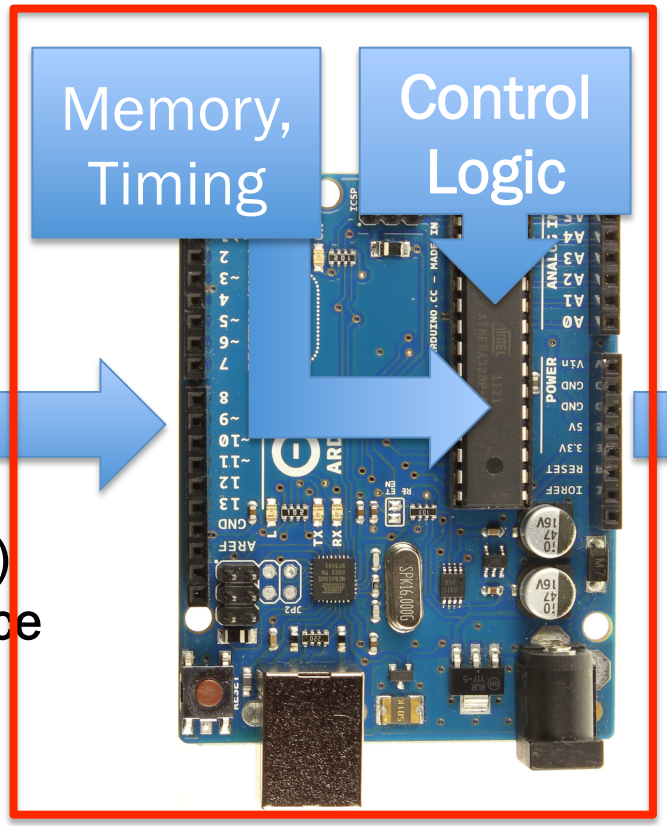
- **Detect animal's location** in linear arena using array of proximity sensors

- **Measure acceleration & rotation** directly using accelerometer & gyroscopic sensor affixed to mouse

Architecture of a typical project

Inputs:

- Sensors
  - Temperature
  - Position
  - Pressure
  - Light
- Control Signals (TTLs)
- Buttons/User Interface



Outputs:

- Actuators/Transducers
  - Motors
  - LEDs
  - Switches/transistors
  - Speakers
- Timing/Control Pulses
- User Feedback

USB/Serial Communication

- PCs, Lab instruments
- Other digital devices

Programming

- Programs are written in the C programming language
- Programs have access to all features of the MCU:

Programming

- Programs are written in the C programming language
- Programs have access to all features of the MCU:

Arduino Uno technical specs

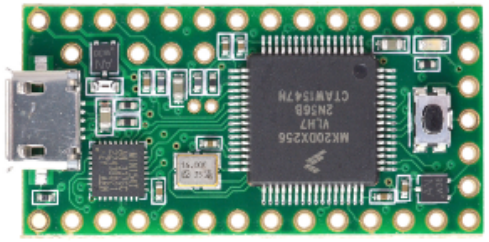
- 16 MHz, 8-bit processor
- 20 Input/output pins
- Memory
  - 32 KB Flash (stores the program, non-volatile)
  - 2 KB RAM ("working memory", volatile)
  - 1 KB EEPROM
- Digital Serial I/O
  - USB, UART, SPI, I2C
- Interrupts
  - Triggered by timers or digital inputs
- Timers/Counters

Many models of Arduino

Different Arduino (-compatible) devices have a range of features & tradeoffs.

Compared to **Arduino Uno**:

Arduino Mega or Due
Faster processor (Due)
More I/O Pins
More expensive



Teensy 3.2
Faster, smaller, more pins
Touch sensors
True analog output
Comparable price

When to use an Arduino

Arduino Capabilities:

- Programmable general-purpose I/O pins
  - Control of several sensors/actuators/ instruments
  - "Programming real things"
- Accurate timing (deterministic)

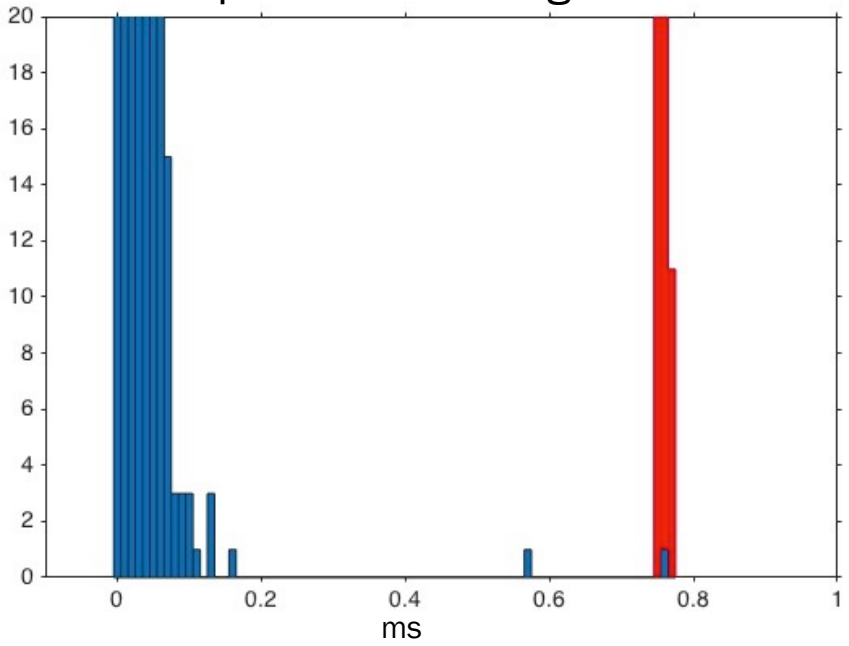
Two typical use cases:

- Communication with PC over USB
  - can provide input to programs running on a PC including Matlab/LabView code
- Standalone operation (no PC)
  - cheap to scale up

When to use an Arduino

Timing Variability:

Compute sum of integers 1-1000



MATLAB

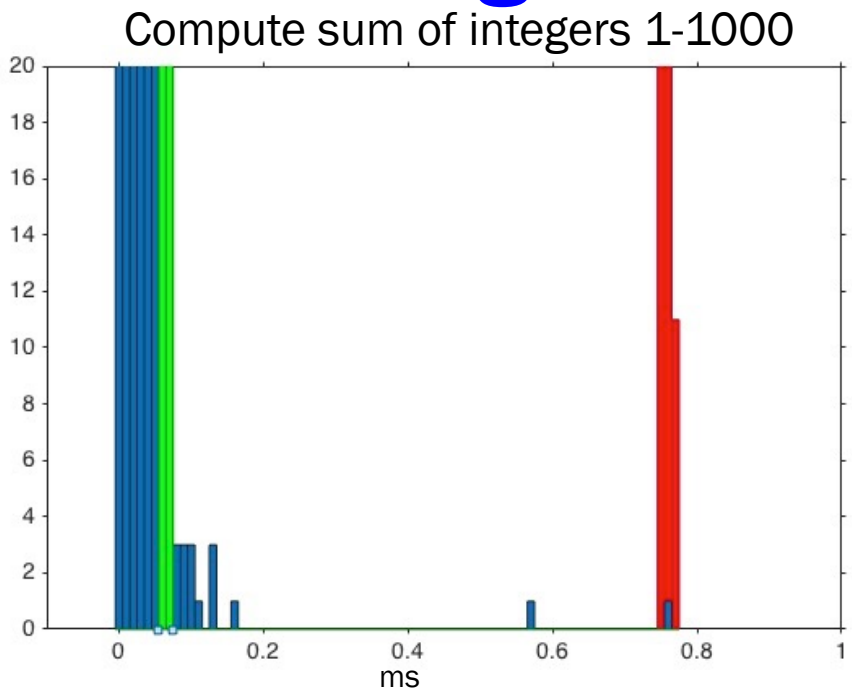
Mean:	3.9 µs
SD:	1.5 µs
Max:	764.0 µs

Arduino Uno

Mean:	759 µs
SD:	2 µs
Range:	20 µs

When to use an Arduino

Timing Variability:



PC:

- Faster on average
- High jitter

Microcontrollers:

- Very reliable timing
  (when programmed right)

MATLAB		Faster MCU (Teensy)		Arduino Uno	
Mean:	3.9 µs	Mean:	63 µs	Mean:	759 µs
SD:	1.5 µs	SD:	1 µs	SD:	2 µs
Max:	764.0 µs	Range:	6 µs	Range:	20 µs

When to use an Arduino/MCU

Arduino Limitations

- Limited input/output and computational bandwidth
  - typically can't handle:
    - video, image analysis
    - arbitrary sound
- Runs only one program (no OS)
- Not designed for high-precision analog input/output

Alternatives

When microcontrollers aren't powerful enough:

Linux single-board-computer
E.g. Raspberry Pi
Low cost (<$100)
Programmable I/O pins

PC with Data Acquisition Board
>$1000
High bandwidth
Precision analog input & output

Time / Cost trade-off

Even the simplest project will take time:

- Find, order and test the hardware components (sensors, actuators)
- Wire up your circuit
- Program the Arduino
- Testing & debugging!!
  - don't assume you'll get it right the first time

Sometimes it's less costly (in time + $$$) to buy a commercial product that "just works."

Building your first Arduino project

Practical skills:

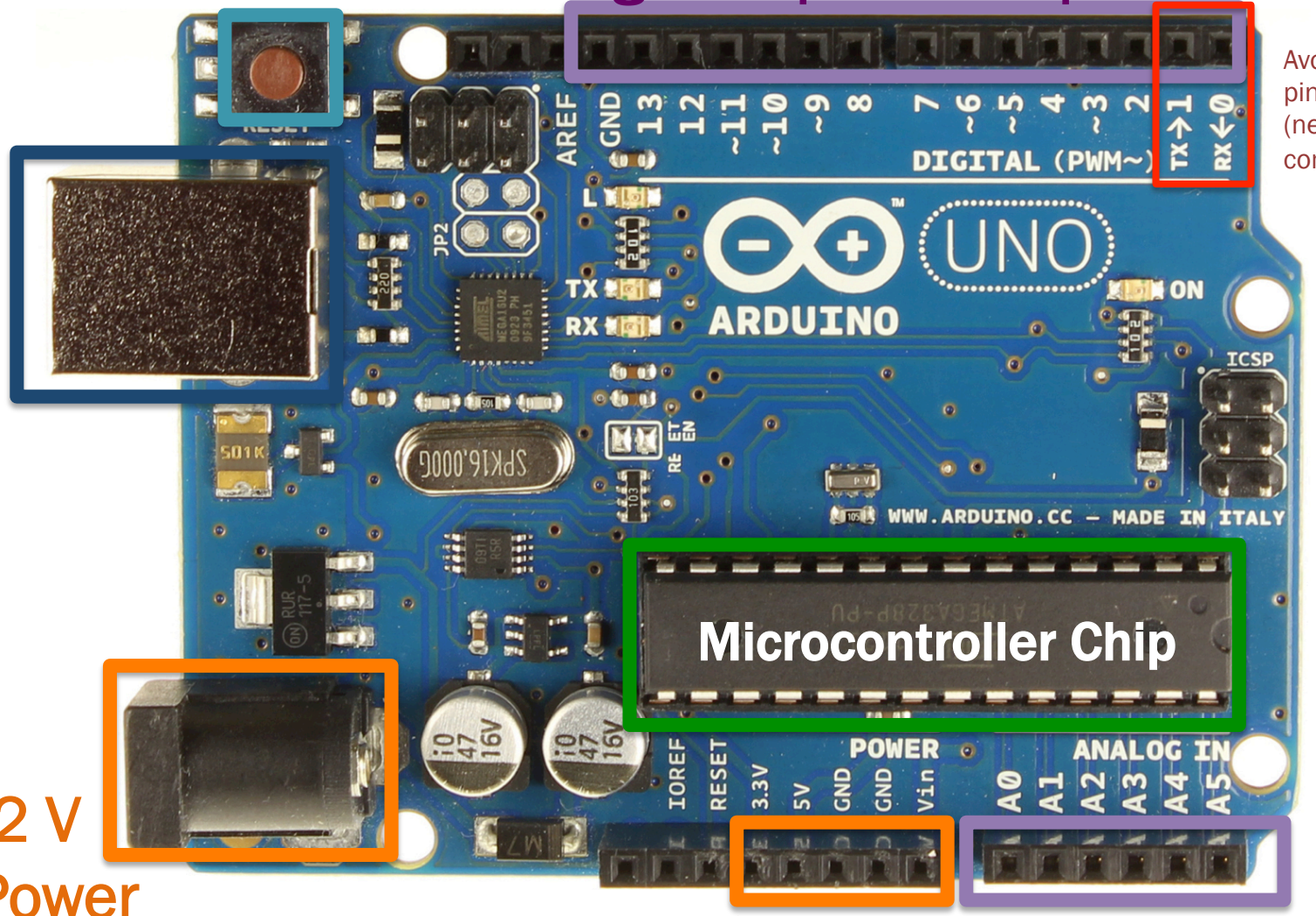
Hardware

- Reading circuit diagrams
- Connecting wires

Software

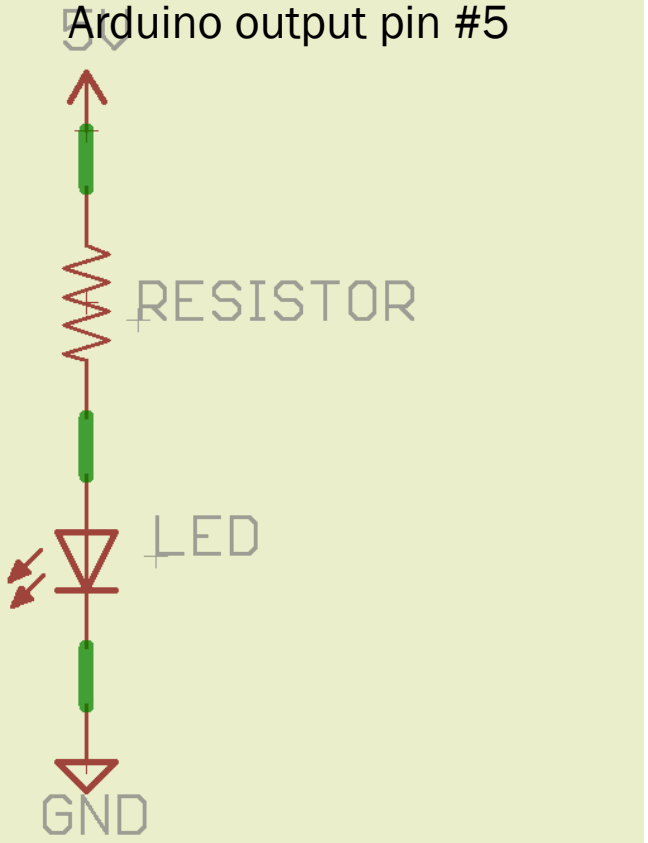
- Arduino programming environment
- Programming in C

Anatomy of an Arduino Uno

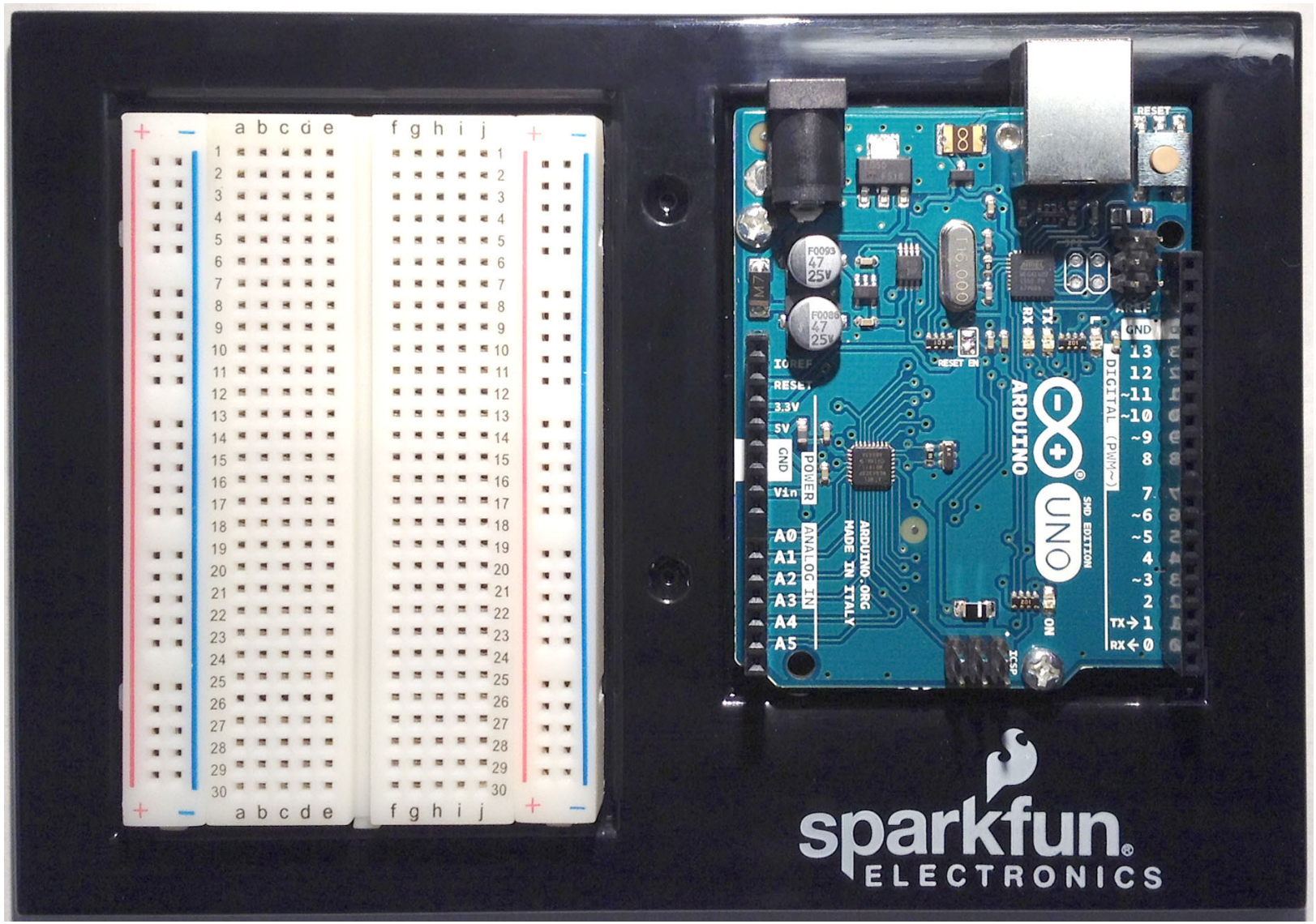


Building your first Arduino project

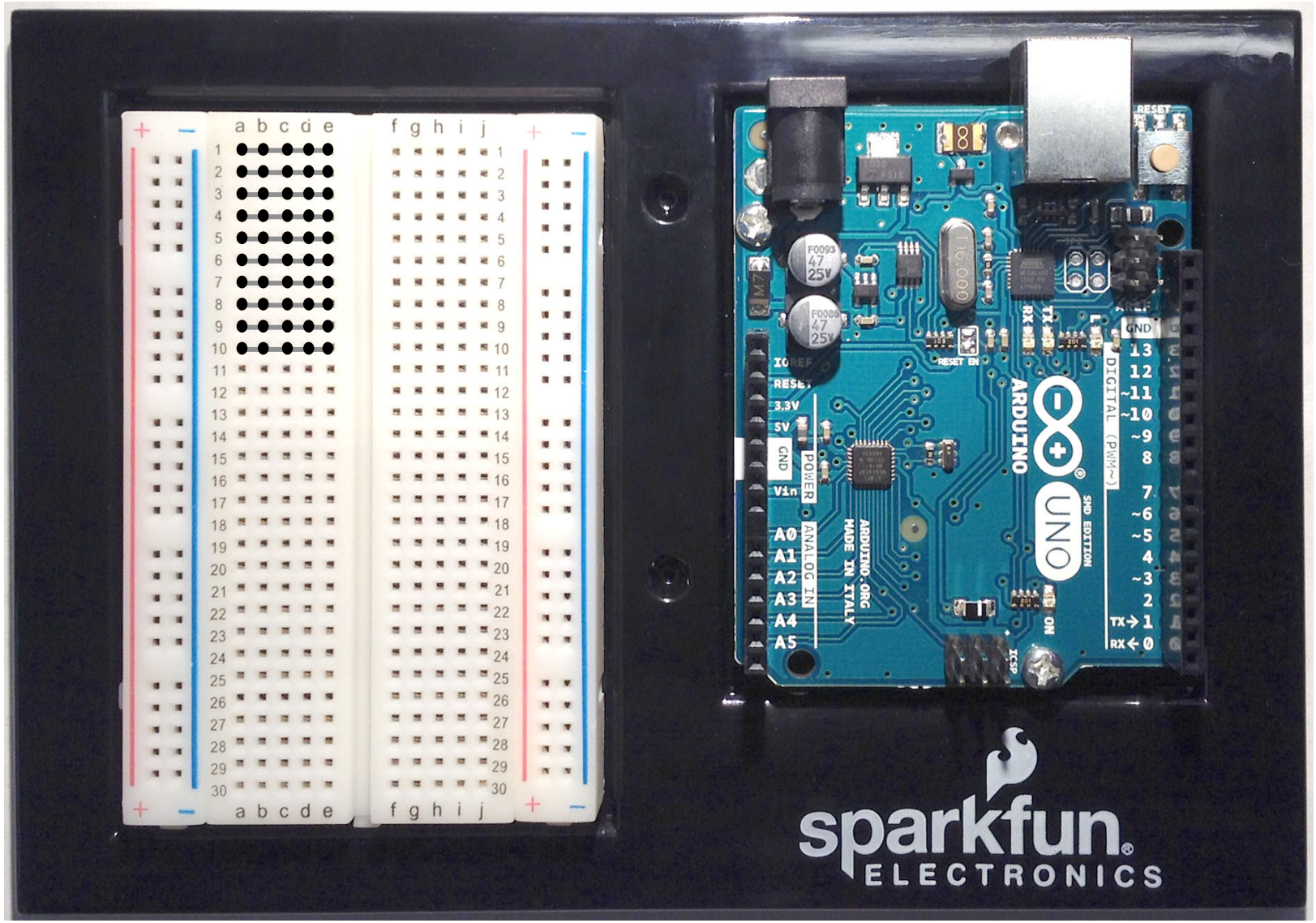
Arduino-controlled LED



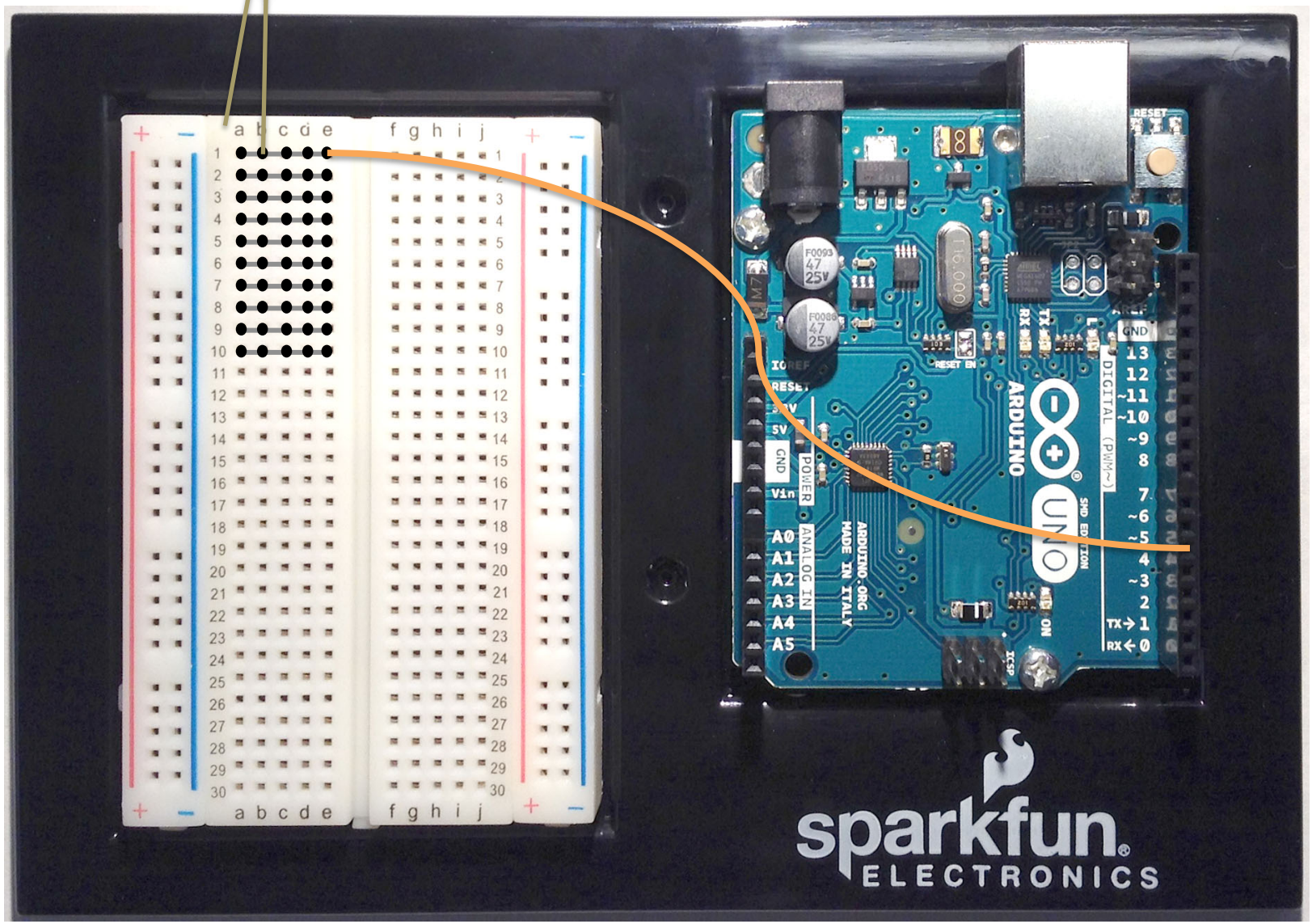
Building your first Arduino project



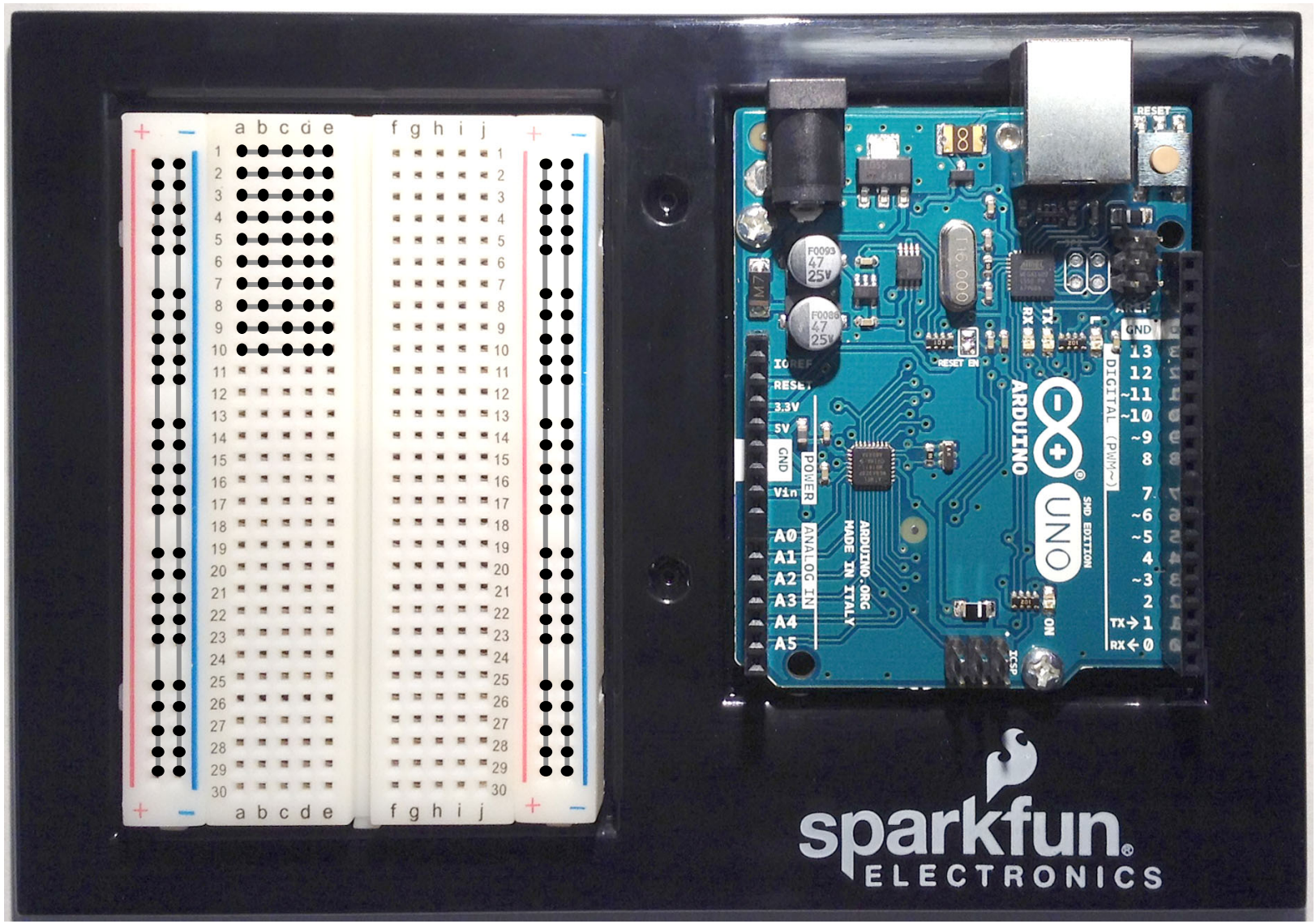
Building your first Arduino project



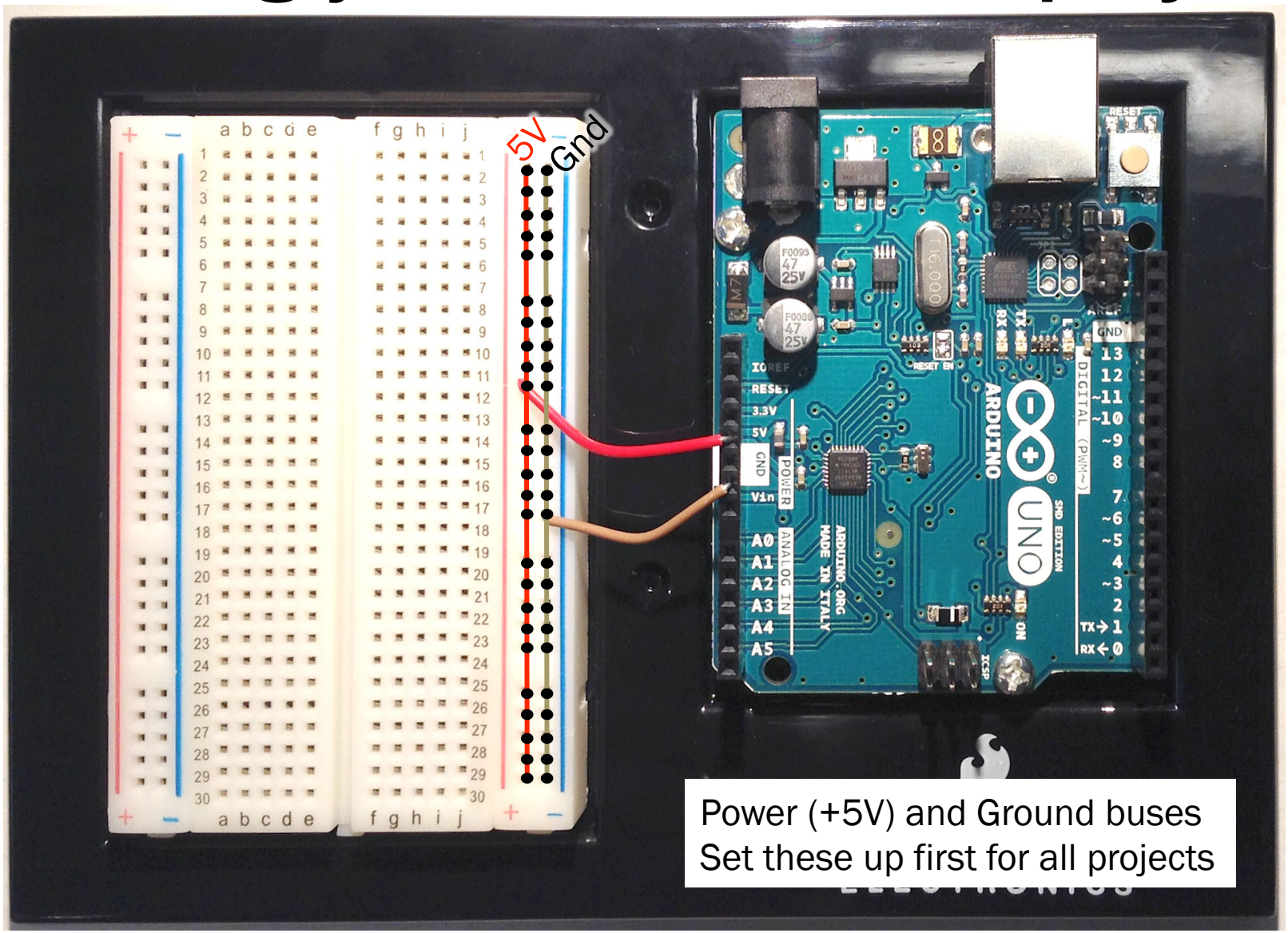
Building your first Arduino project



Building your first Arduino project

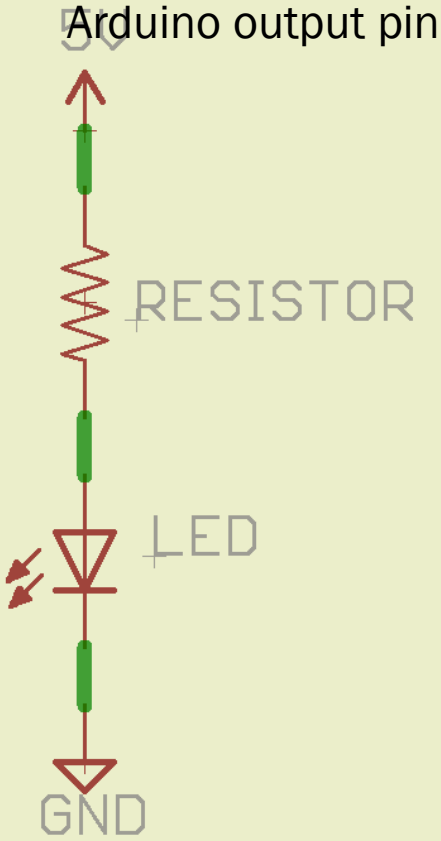


Building your first Arduino project

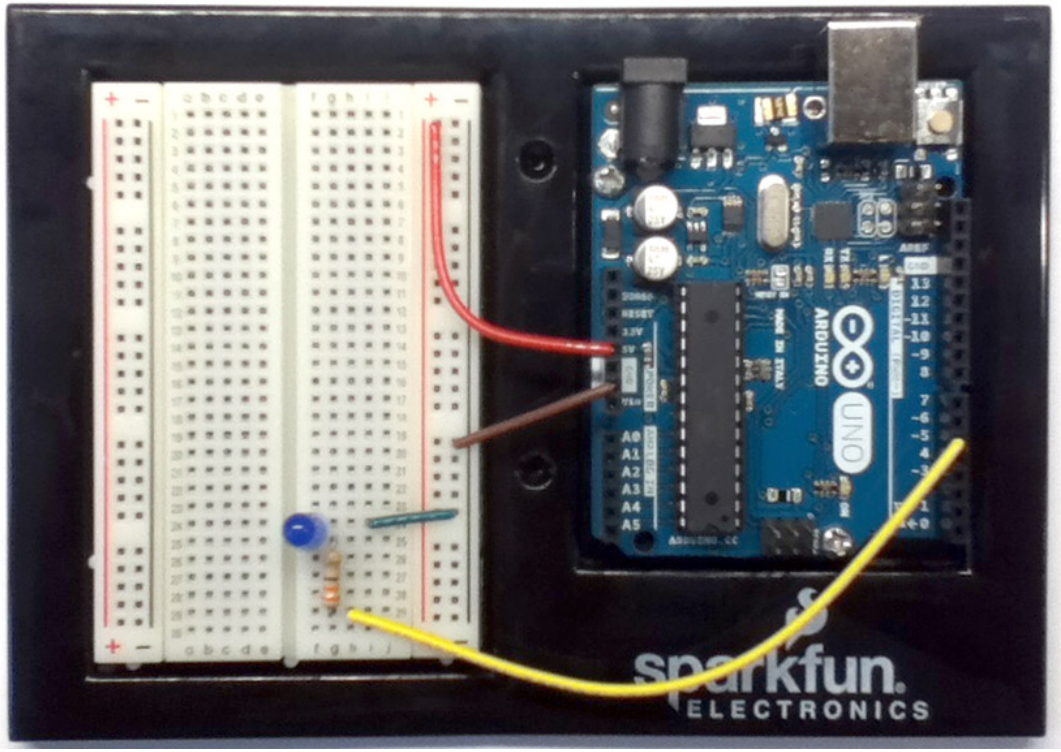


Building your first Arduino project

Circuit Schematic

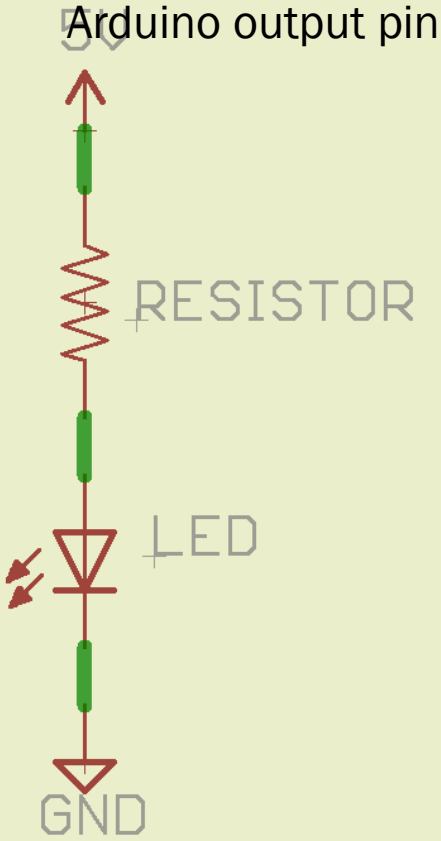


Actual Circuit

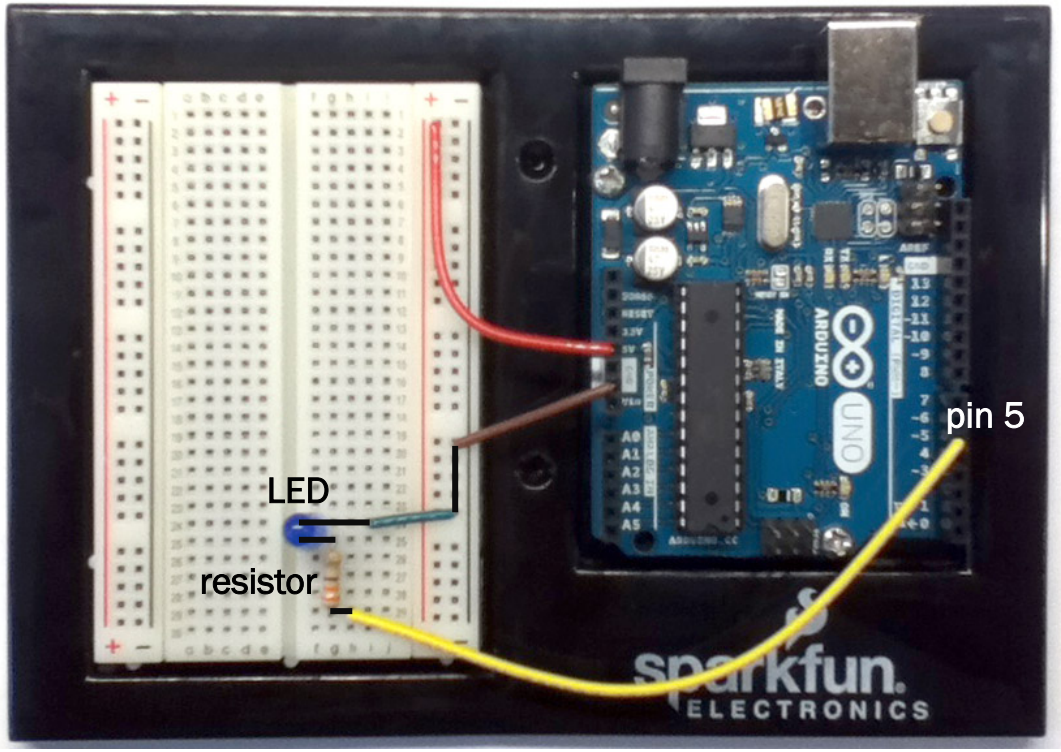


Building your first Arduino project

Circuit Schematic



Actual Circuit

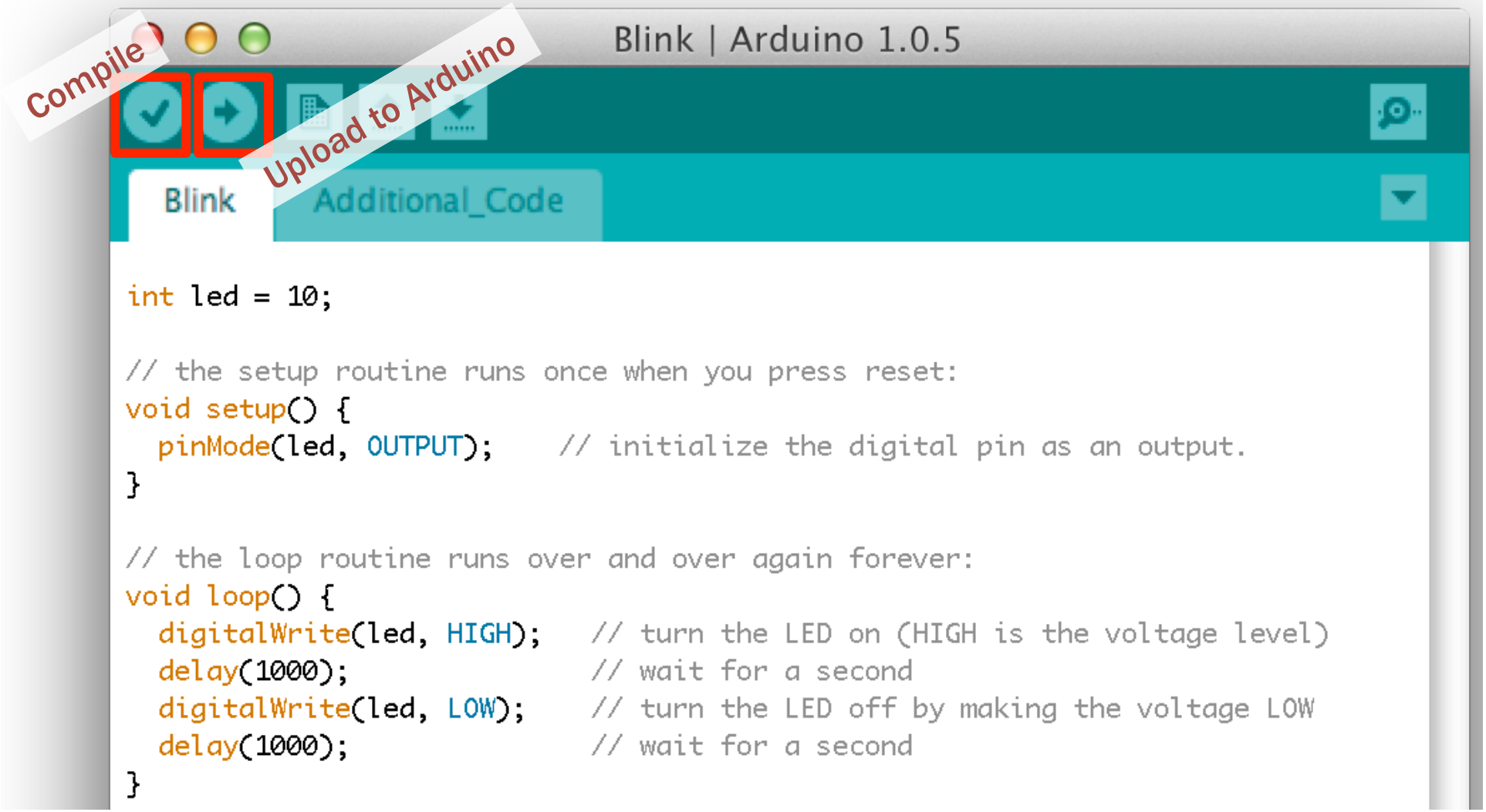


Programming the Arduino

- The Arduino is programmed in C.
- You write and compile a program on your computer and then upload to the Arduino
- Basic C routines to control the Arduino come pre-installed

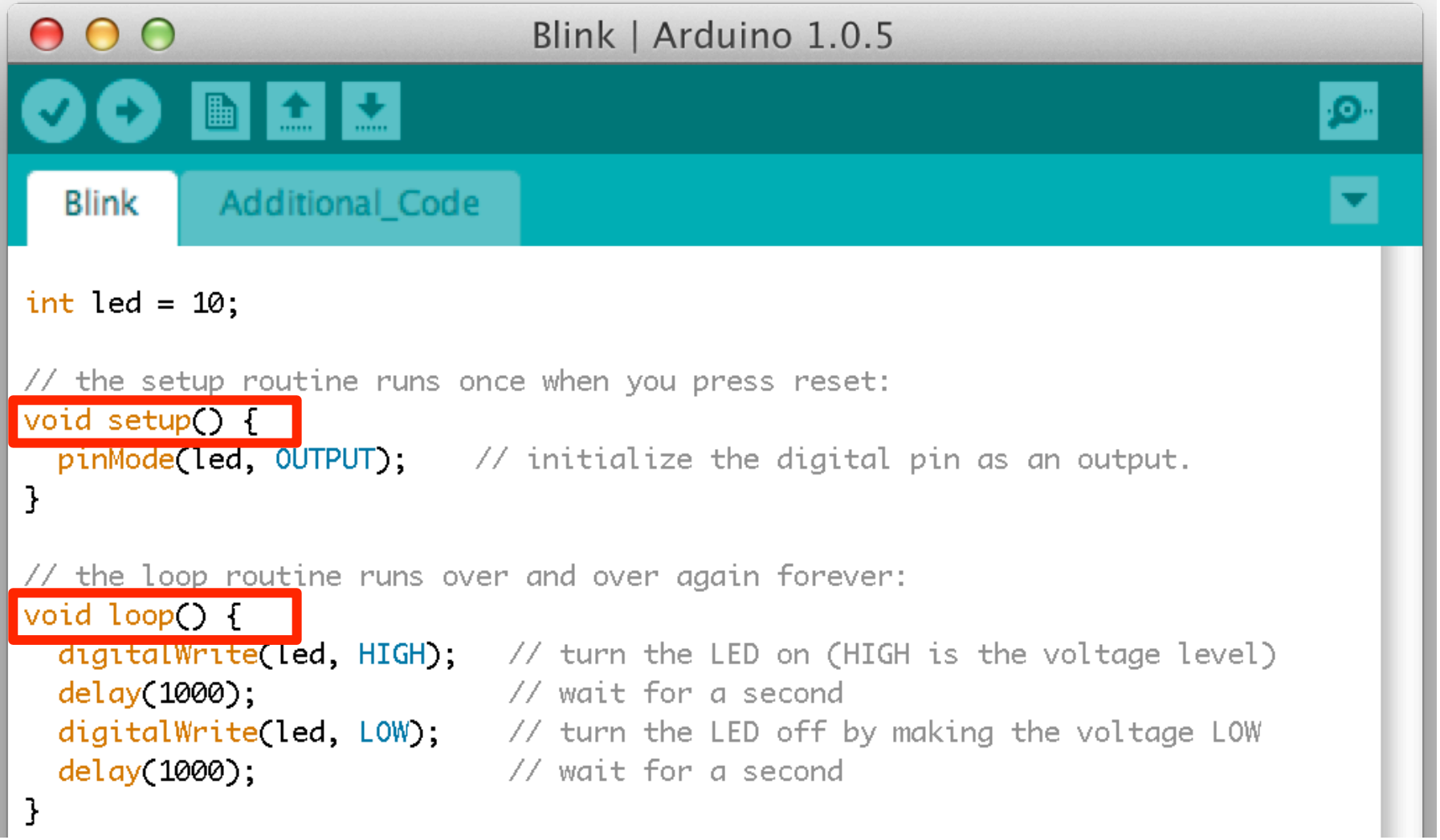
Programming the Arduino

Arduino-programming software runs on your Mac/PC:



Programming the Arduino

Arduino-programming software runs on your Mac/PC:



Programming the Arduino

Programming in C:

- Every line with a command must end with a semicolon (;).

- Variables must be declared before use:

WRONG:

```
x = 3;
```

CORRECT:

```
int x;
x = 3;
int y = 4;
```

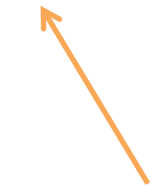
Programming the Arduino

Programming in C:

- Numeric data types:

```
int a, b, c;    // integer type
a = 3;
b = 2;
c = a + b;      // c is 5

c = a / b;      // c is 1  !!
```



Text after "//" is a comment, not executed by Arduino

Programming the Arduino

Programming in C:

- Numeric data types:

```
float a, b, c;   // floating point type

a = 3.0;
b = 2.0;
c = a / b;          // c is 1.5

int i;
i = 5;              // Converting ints to floats:
float f = i;
c = f / 2.0;        // c is 2.5
```

Programming the Arduino

Programming in C:

- Calling functions:

```
Serial.begin(9600);  // initialize USB comm. with PC

Serial.print("Message to sent to PC over USB");
```

Programming the Arduino

Programming in C:

- `if` statement syntax:

```
int a = 3;

if (a < 10) {
    Serial.print("a is less than 10");    // This will print
} else {
    Serial.print("a is 10 or greater");   // This won't
}

if (a == 10) {
    Serial.print("a equals 10");          // This won't
}
```

Note that = and == are different:

= assigns a value to a variable

== tests for equality

Programming the Arduino

Programming in C:

"=" versus "==" :

```
int a = 3;

if (a == 10) {
    Serial.print("a equals 10");          // This won't print
}

if (a = 10) {
    Serial.print("a is now 10");          // This will print!!
}                                         // DON'T DO THIS
```

Note that = and == are different:

= assigns a value to a variable

== tests for equality

Useful Resources

Help/Reference

www.arduino.cc

learn.sparkfun.com

arduino.stackexchange.com/

www.allaboutcircuits.com/textbook

gammon.com.au/forum/bbshowpost.php?bbtopic_id=123

Introductory kit with parts & instructions:

www.sparkfun.com/products/14189

($100)

Parts

www.sparkfun.com

www.adafruit.com

www.pololu.com (robotics)

These 3 stores have a curated selection and information for new users.

www.digikey.com

www.newark.com

Overwhelming selection of all possible components.

Final Thoughts

- Experiment! Don't be afraid to try things out
- Remember that testing & debugging are part of the process
- Have fun!

Class website:

github.com/hms-ric/arduinoNanocourse