



University of Toronto Coders

Gadgets and hardware: Introduction to Programming Small Devices

Brian Sutherland

Faculty of Information/Institute for Medical Sciences

University of Toronto

Thursday, November 23, 2017



Open Development Platform

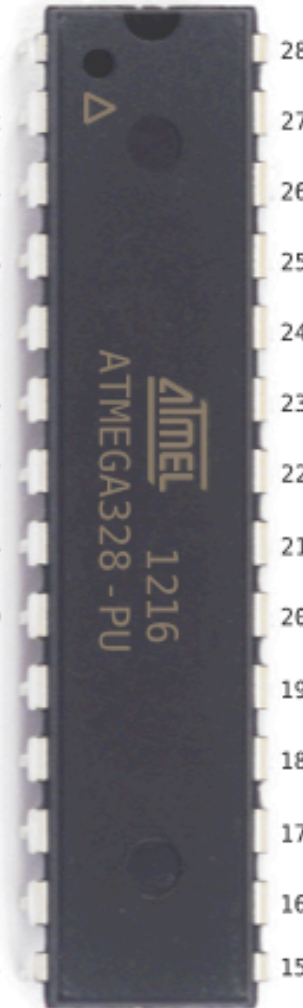
What is Arduino?

- < \$50 microcontroller board (nominal or advanced features)
- [Programming Language](#) based on 'Wiring', electronics variant of Processing (python and C also popular)
- **I**ntegrated **D**evelopment **E**nvironment IDE for developing software (core capabilities +libraries) based on '[Processing](#)'
- An Open Source Community








Arduino function

	reset	PC6	1
	digital pin 0	PD0	2
	digital pin 1	PD1	3
	digital pin 2	PD2	4
	digital pin 3	PD3	5
	digital pin 4	PD4	6
	VCC	VCC	7
	GND	GND	8
	crystal	PB6	9
	crystal	PB7	10
	digital pin 5	PD5	11
	digital pin 6	PD6	12
	digital pin 7	PD7	13
	digital pin 8	PB0	14

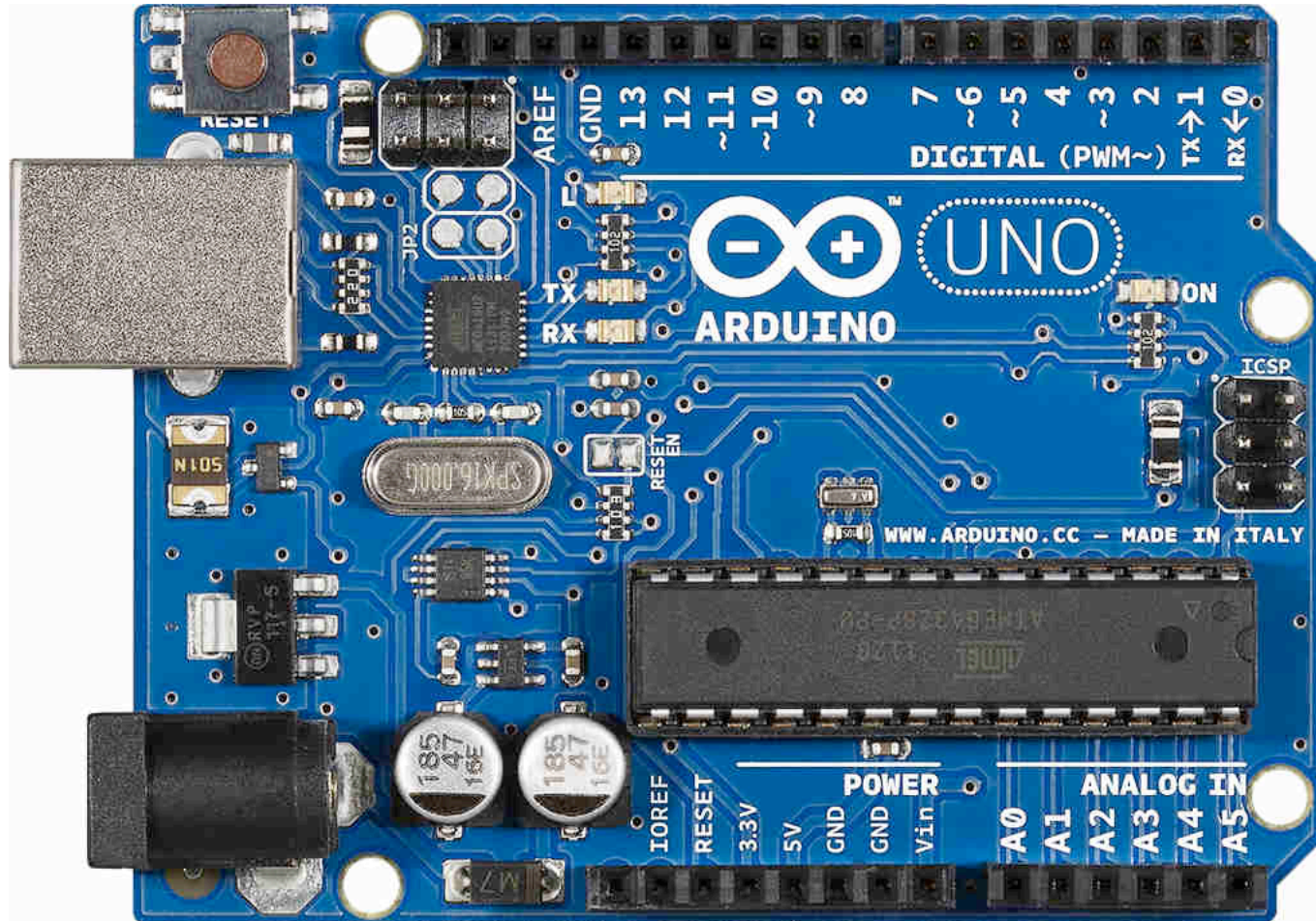
**ATmega328
pin mapping**



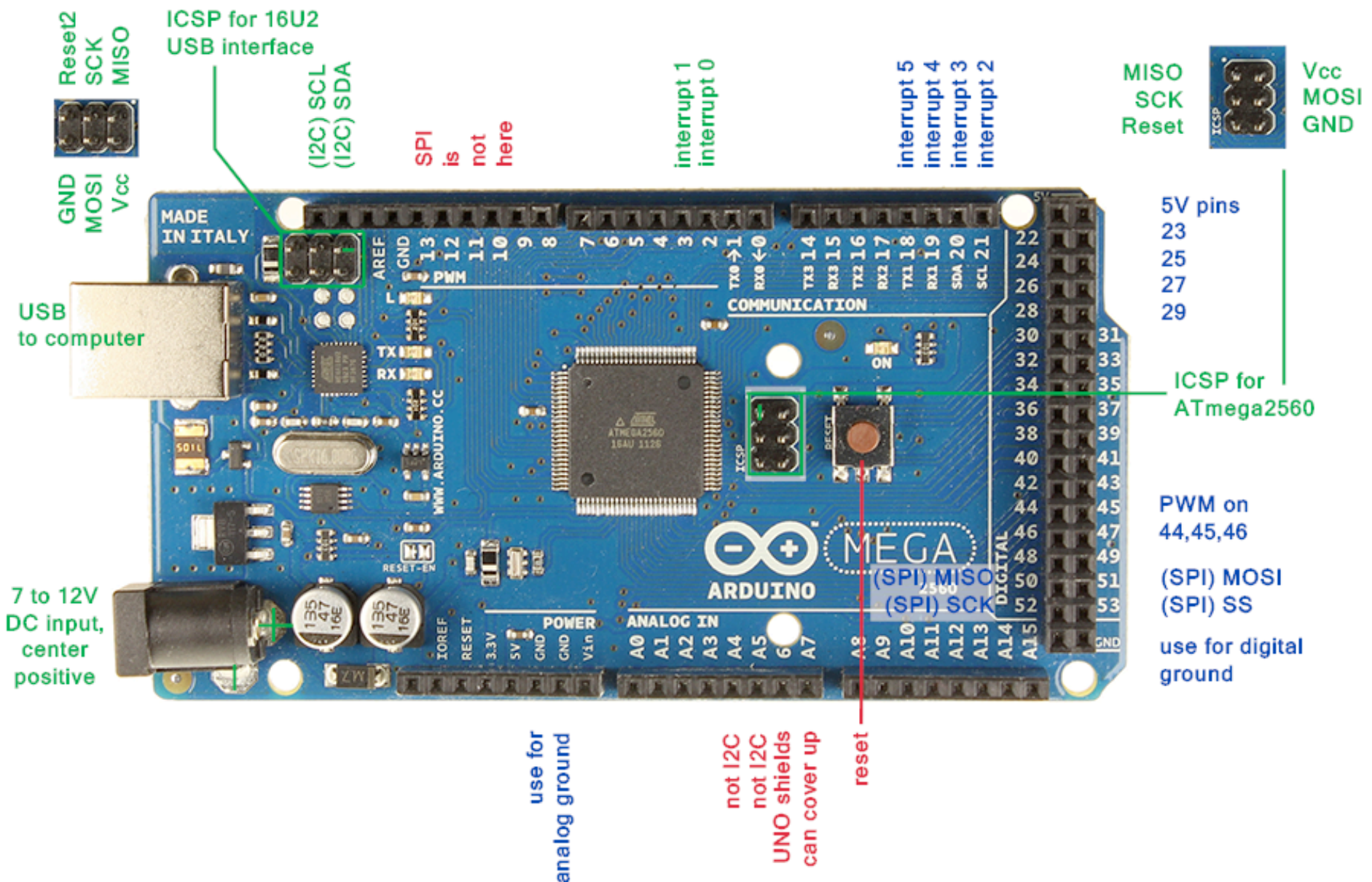

Arduino function

28	PC5	analog input 5
27	PC4	analog input 4
26	PC3	analog input 3
25	PC2	analog input 2
24	PC1	analog input 1
23	PC0	analog input 0
22	GND	GND
21	AREF	analog reference
20	AVCC	AVCC
19	PB5	digital pin 13 
18	PB4	digital pin 12 
17	PB3	digital pin 11  
16	PB2	digital pin 10 
15	PB1	digital pin 9 

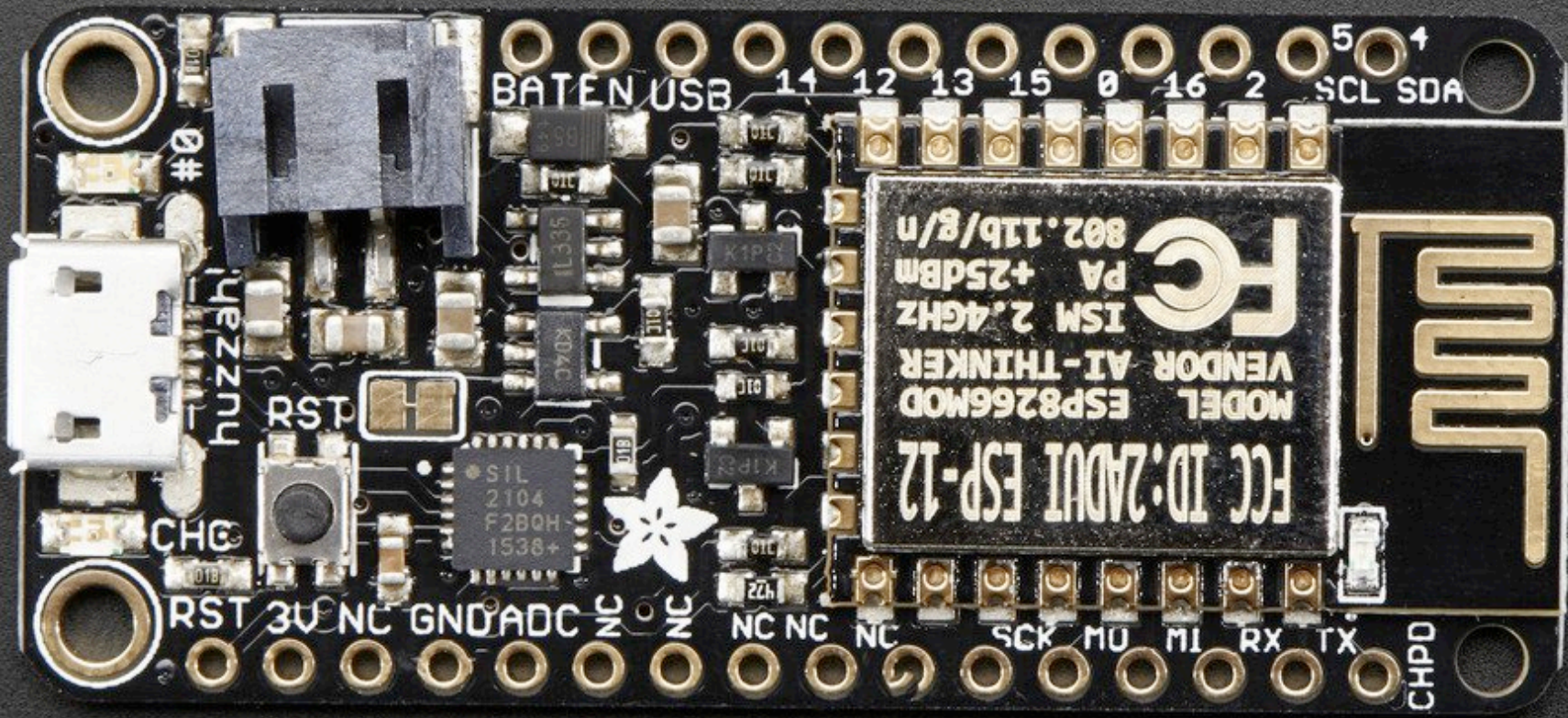
Microcontroller = CPU + RAM + I/O



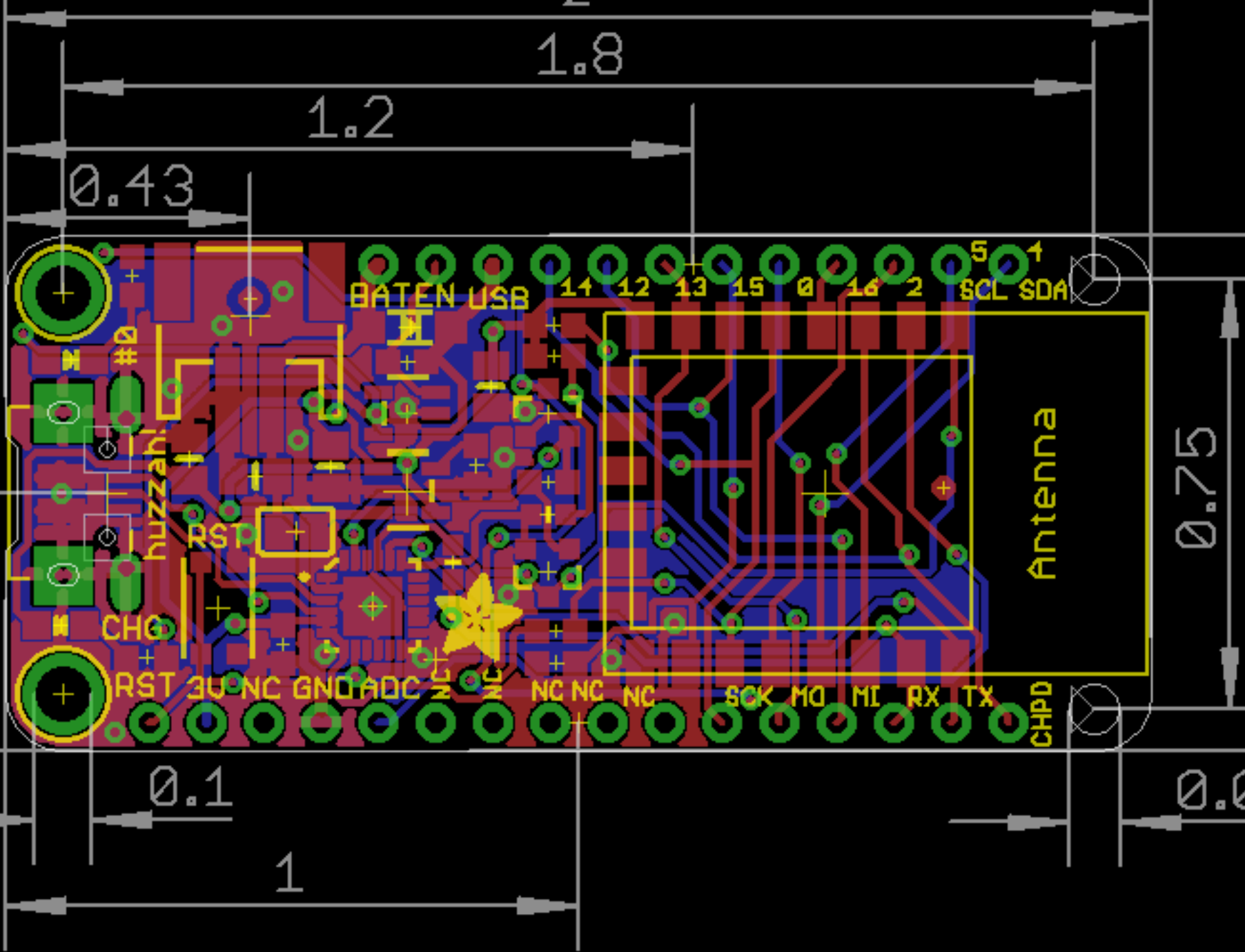
Microcontroller Board: Arduino Uno (comparable to IBM XT)

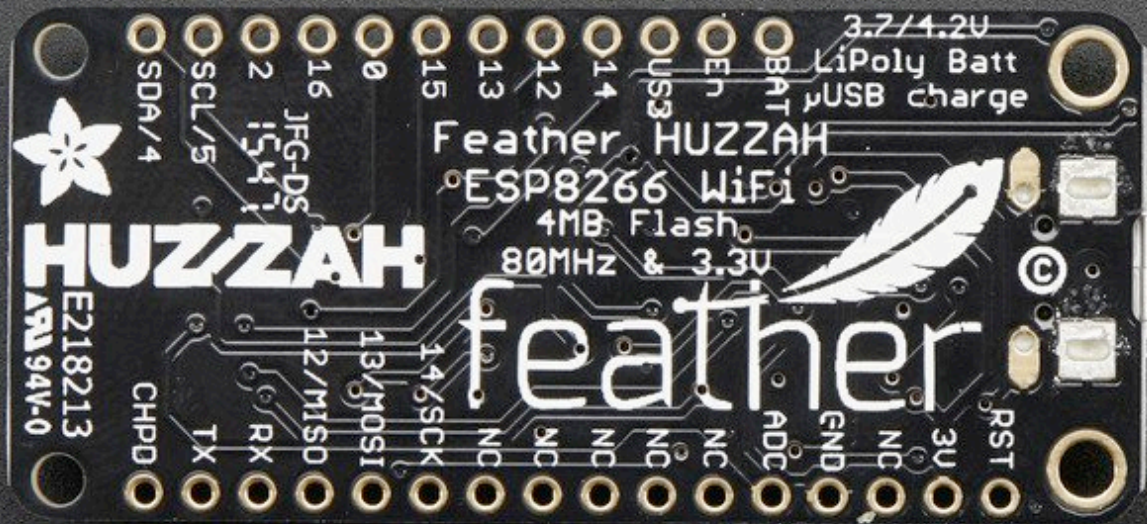


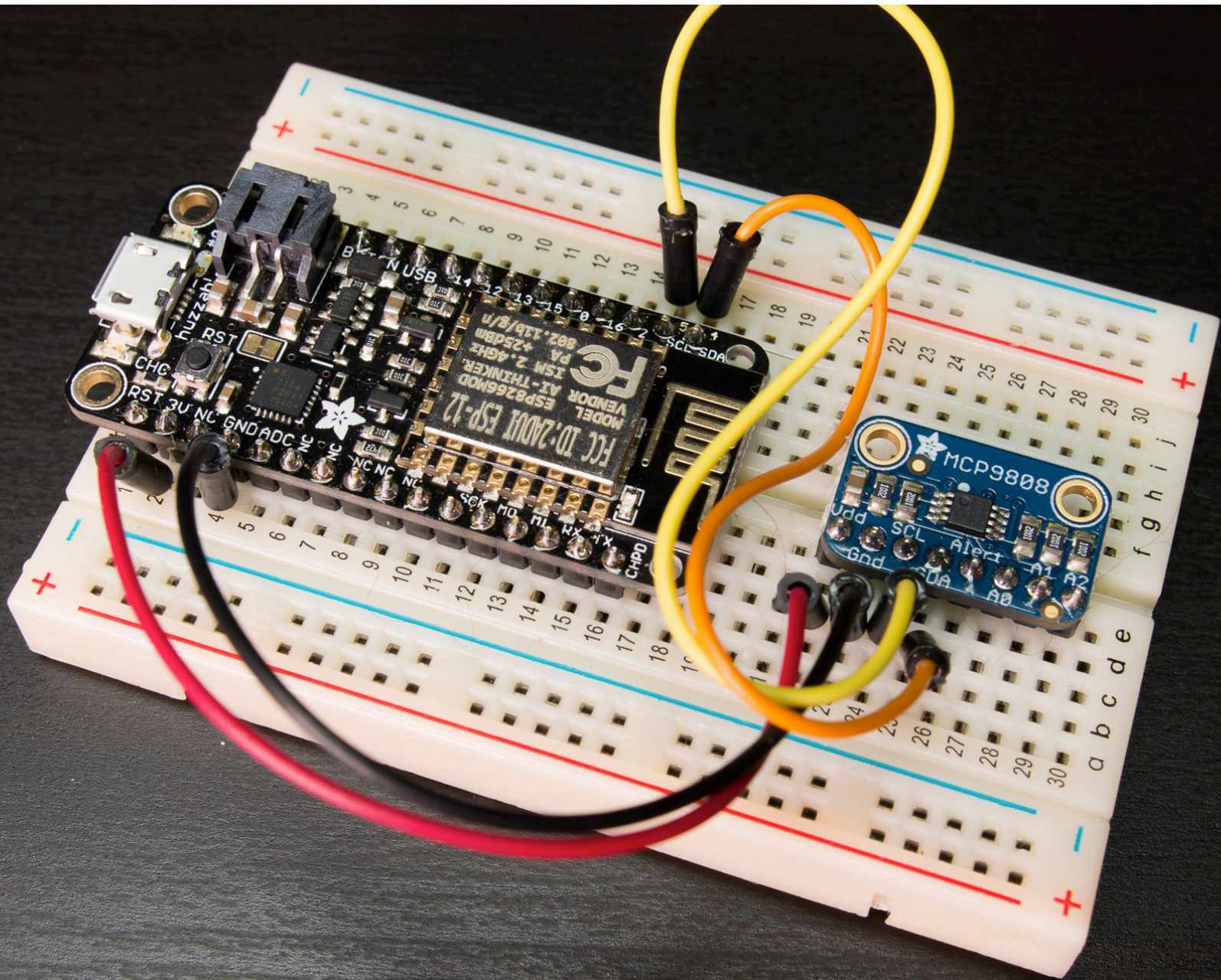
Microcontroller Board - Arduino Mega: same speed, more connectors, memory

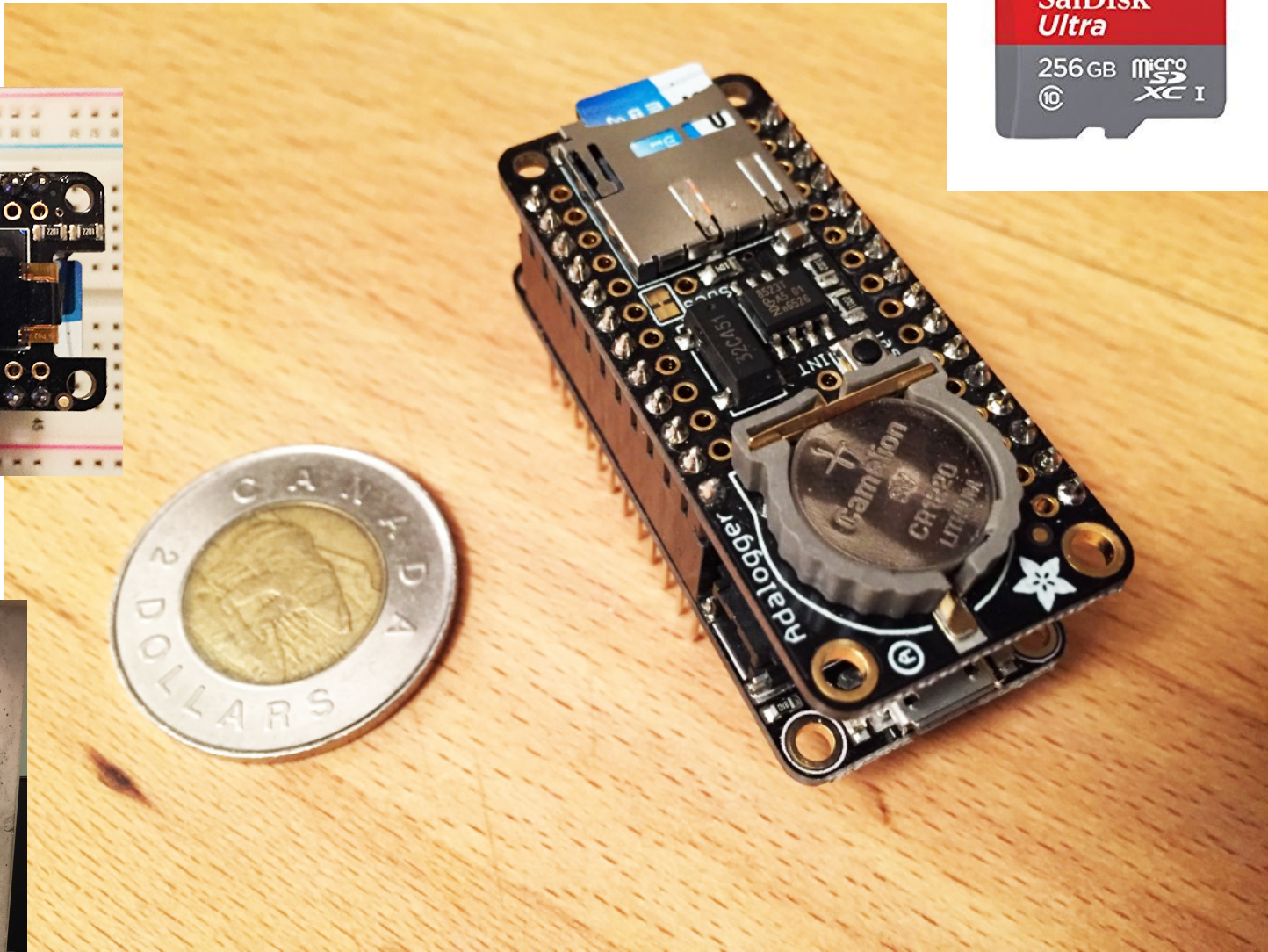
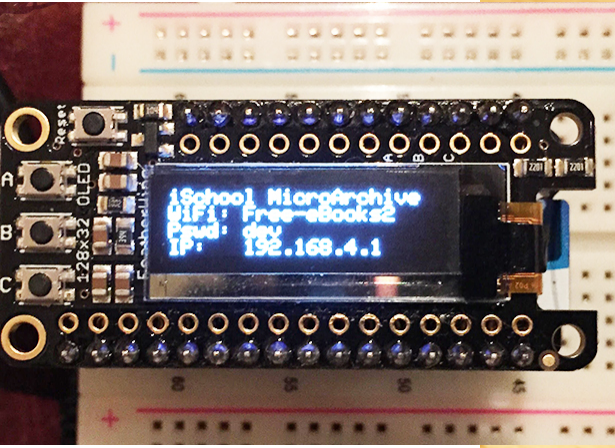


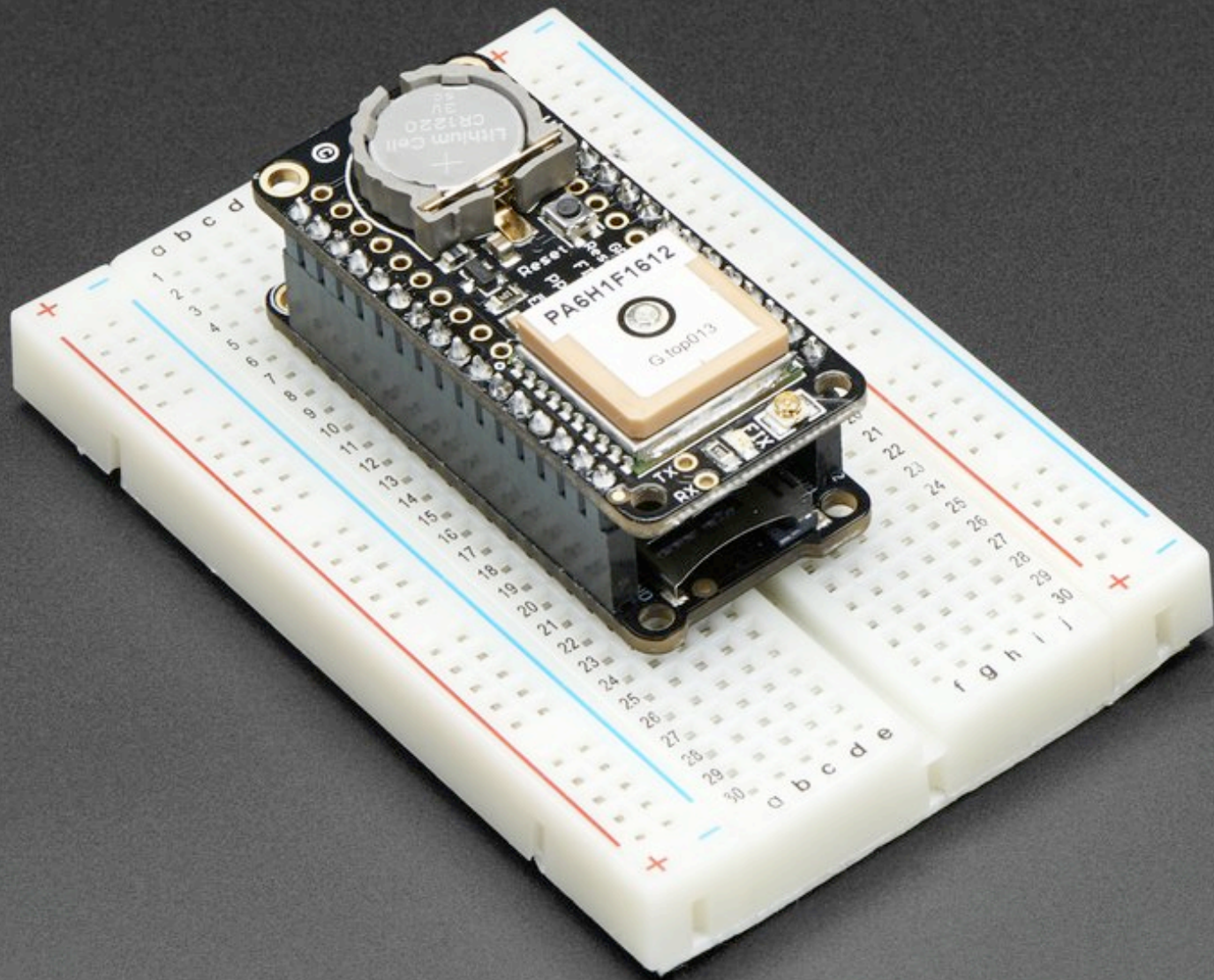
Microcontroller Board - Feather Huzzah ESP8266: postage stamp sized, 10x faster, 128x memory, 2.4GHz WiFi!





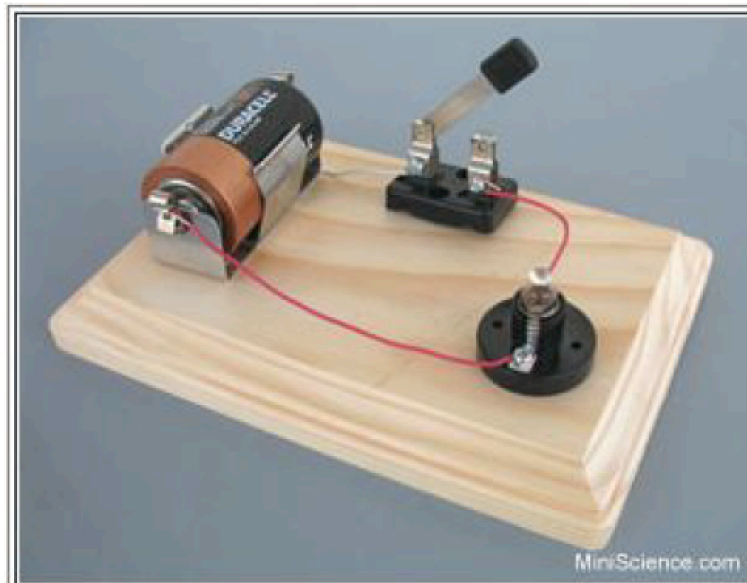




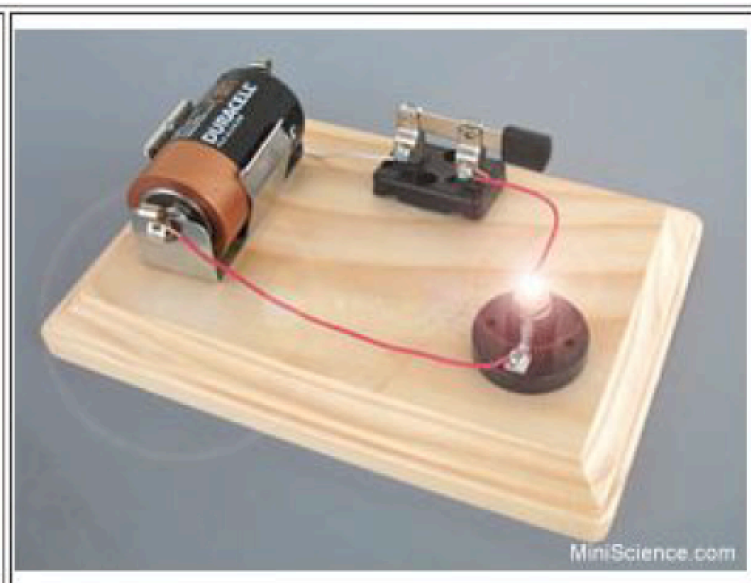


Electric Circuits (Review)

- Conductive loops from power (battery) to component (light bulb)



The circuit or switch is open, The light is off.



The circuit or switch is closed, The light is on.



Use Arduino IDE on our Ubuntu PCs (skip next slide)
Set up Arduino IDE on your own computer (next slide)

Set up Arduino. <http://arduino.cc/> > Software > Download the IDE
Add our board specifications and code samples to the development environment.

Step One: Tell the IDE where the specification is

Arduino > Preferences

Look for the Additional Boards Manager URLs field

Paste: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Select 'OK'

Step Two: Load the specification from the URL

Tools > Board – Boards Manager (at the top)

Scroll to the bottom where it reads

ESP8266 by ESP8266 community.

Select and Install

Step Three: Restart Arduino IDE

Close Arduino

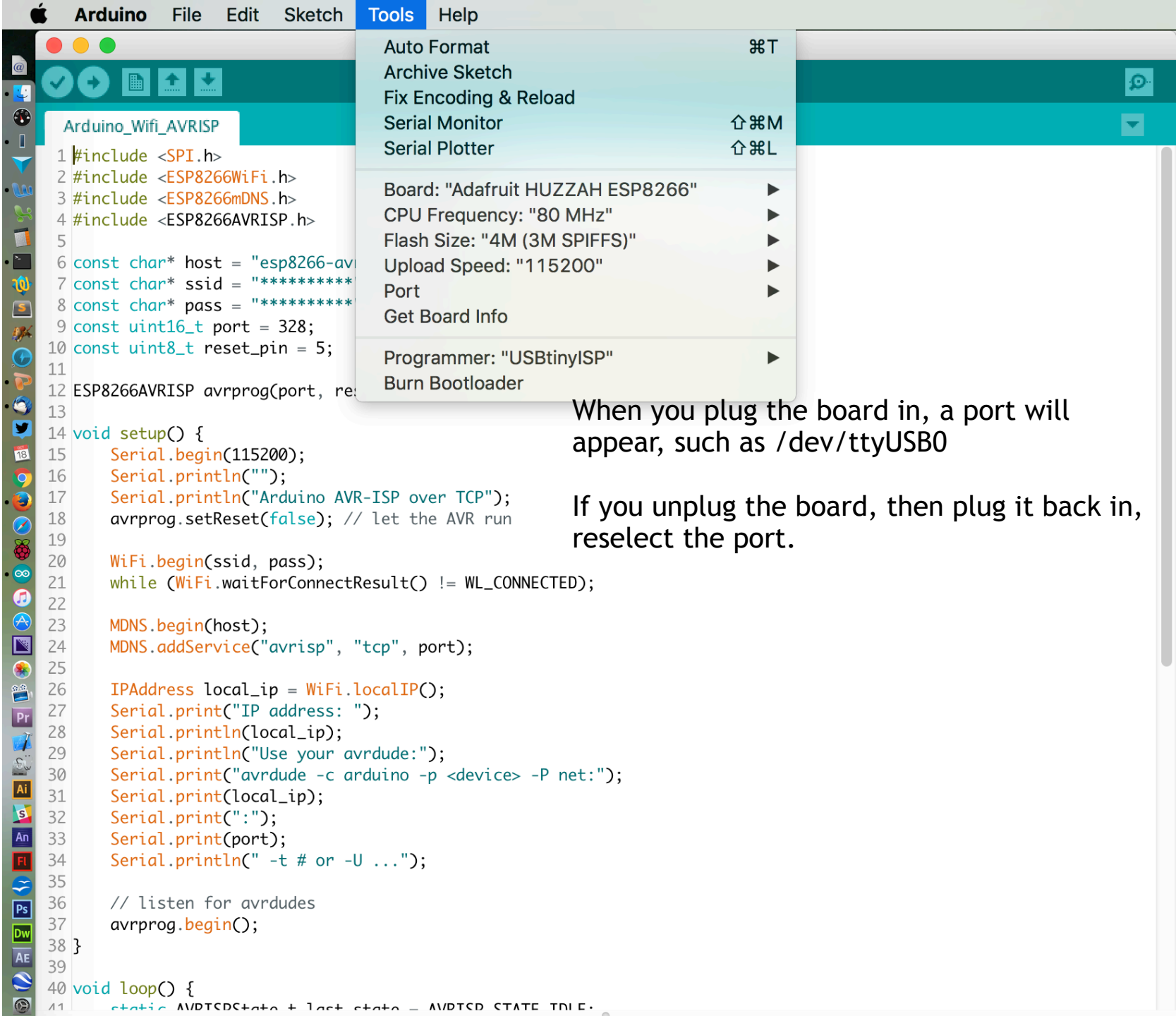
Open Arduino again, and select Board

From the drop down a series of ESP8266 boards should now appear

Set board to Adafruit Huzzah ESP8266

When connecting the board, set port (e.g. COM4 on Windows; /dev/ttyUSB0 on Unix)

The example code for ESP8266 should appear, and your board is ready to send programs to.



- Auto Format ⌘T
- Archive Sketch
- Fix Encoding & Reload
- Serial Monitor ⇧⌘M
- Serial Plotter ⇧⌘L
- Board: "Adafruit HUZZAH ESP8266" ▶
- CPU Frequency: "80 MHz" ▶
- Flash Size: "4M (3M SPIFFS)" ▶
- Upload Speed: "115200" ▶
- Port ▶
- Get Board Info
- Programmer: "USBtinyISP" ▶
- Burn Bootloader

When you plug the board in, a port will appear, such as /dev/ttyUSB0

If you unplug the board, then plug it back in, reselect the port.

```

Arduino_Wifi_AVRISP
1 #include <SPI.h>
2 #include <ESP8266WiFi.h>
3 #include <ESP8266DNS.h>
4 #include <ESP8266AVRISP.h>
5
6 const char* host = "esp8266-avr";
7 const char* ssid = "*****";
8 const char* pass = "*****";
9 const uint16_t port = 328;
10 const uint8_t reset_pin = 5;
11
12 ESP8266AVRISP avrprog(port, reset_pin);
13
14 void setup() {
15   Serial.begin(115200);
16   Serial.println("");
17   Serial.println("Arduino AVR-ISP over TCP");
18   avrprog.setReset(false); // let the AVR run
19
20   WiFi.begin(ssid, pass);
21   while (WiFi.waitForConnectResult() != WL_CONNECTED);
22
23   MDNS.begin(host);
24   MDNS.addService("avrisp", "tcp", port);
25
26   IPAddress local_ip = WiFi.localIP();
27   Serial.print("IP address: ");
28   Serial.println(local_ip);
29   Serial.println("Use your avrdude:");
30   Serial.print("avrdude -c arduino -p <device> -P net:");
31   Serial.print(local_ip);
32   Serial.print(":");
33   Serial.print(port);
34   Serial.println(" -t # or -U ...");
35
36   // listen for avrdudes
37   avrprog.begin();
38 }
39
40 void loop() {
41   static AVRISPState_t last_state = AVRISP_STATE_IDLE;

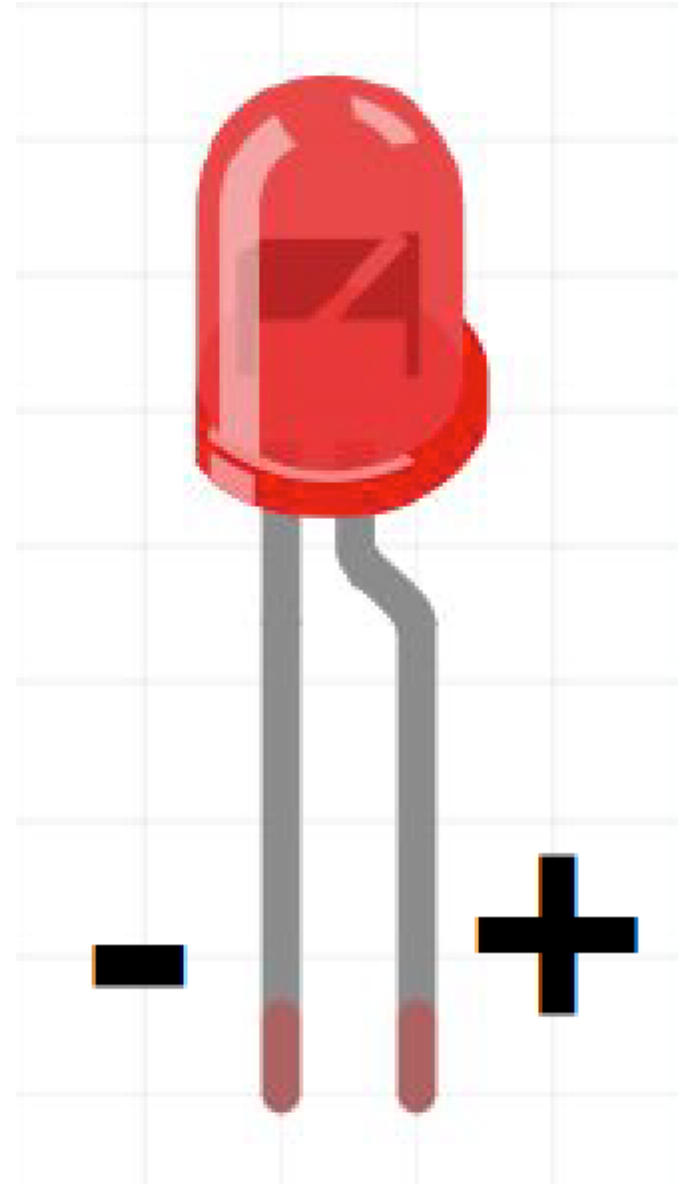
```

First Project

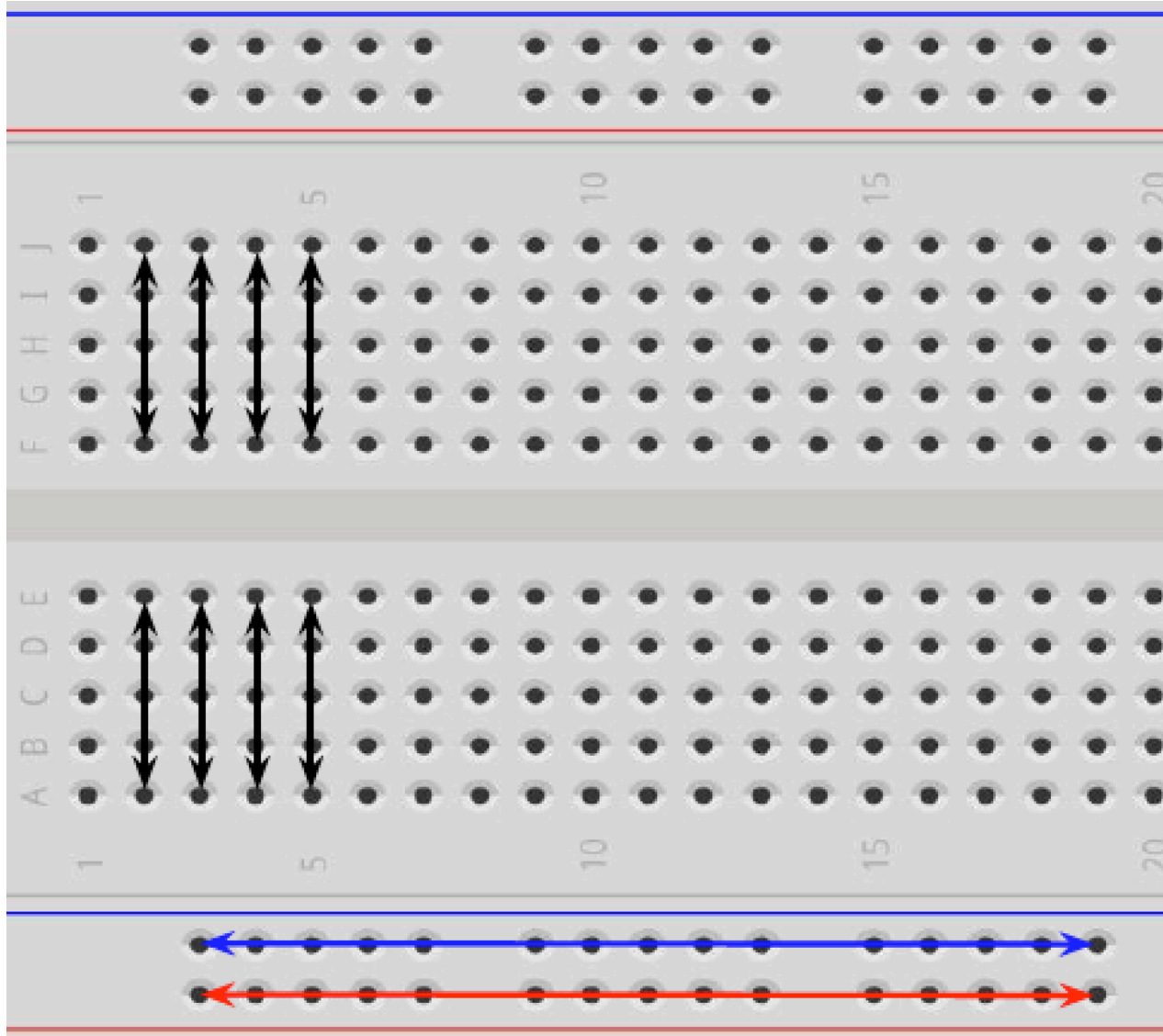
- Arduino IDE and writing programs to the ESP8266
 - File > Examples > ESP8266 > Blink
 - Set the desired blink frequency via the delay
 - Select the checkmark to check your syntax
 - Select the arrow to Flash the ESP8266
 - ESP8266 red led lights dimly, blue LED flickers on and off, then reboots
 - On board LED should blink

Light Emitting Diode

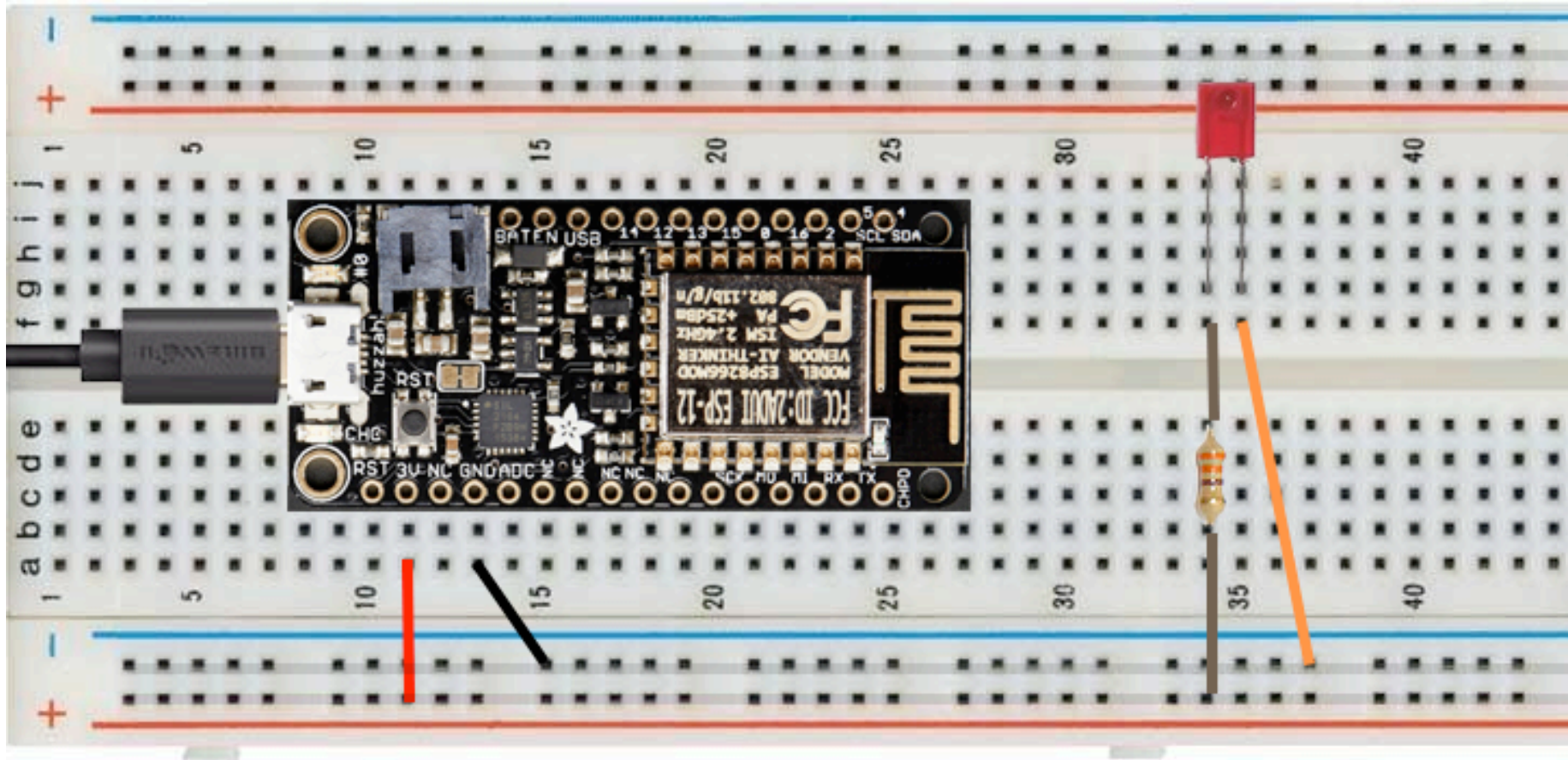
- Longer foot is +ve
3.3V
- Shorter foot is -ve
- Unlike light bulbs, LEDs have polarity



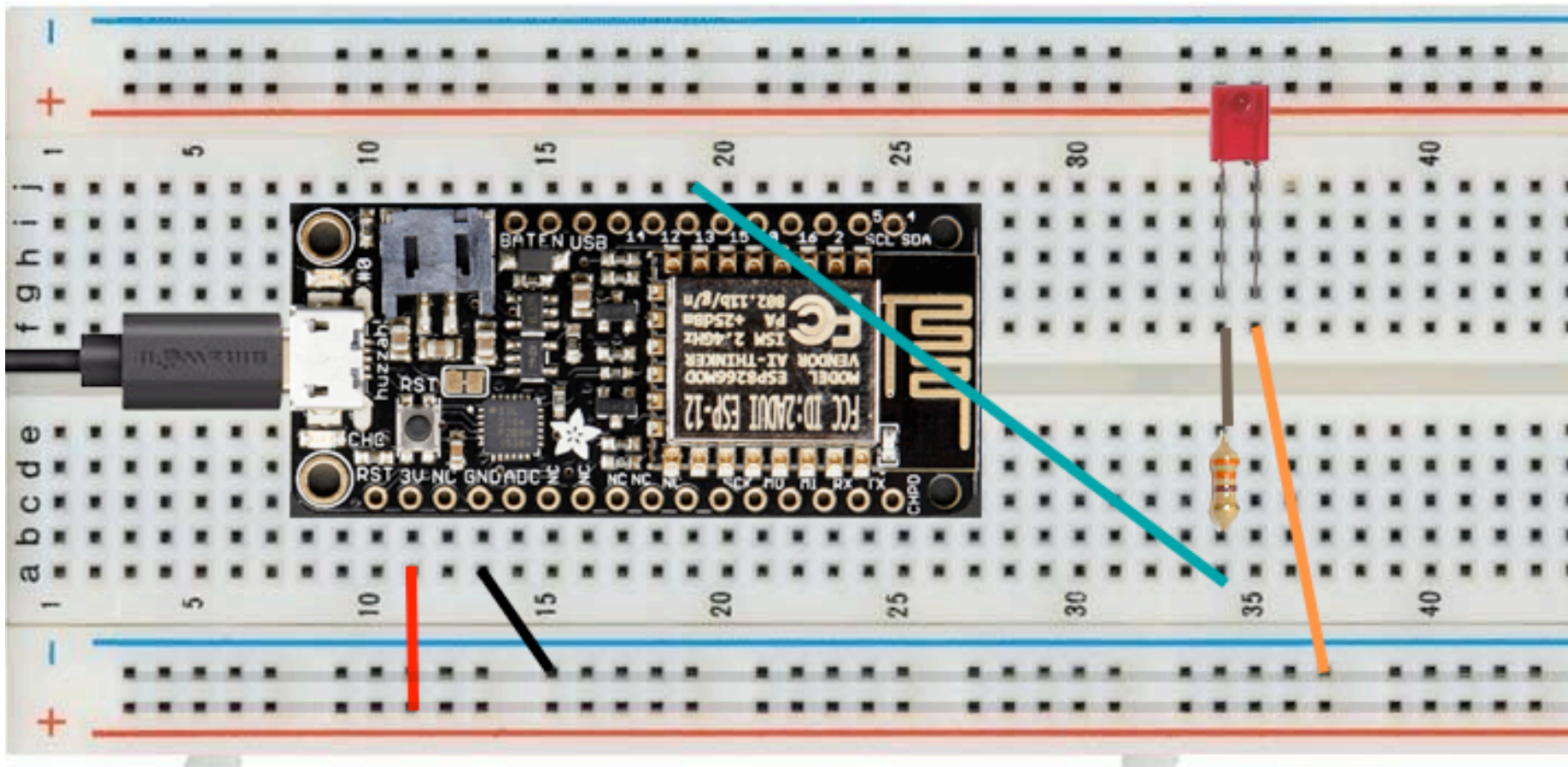
Solderless Breadboard



Power an LED from the Board



Blink an LED from the Board (pin 13)



sketch_oct28_blink

```
1 void setup() {
2   // put your setup code here, to run once:
3   pinMode(13, OUTPUT);
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
9   delay(1000);           // wait for a second
10  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
11  delay(2000);           // wait two seconds
12 }
```

Done uploading.

```
Starting app without reboot
  espcomm_send_command: sending command header
  espcomm_send_command: sending command payload
  espcomm_send_command: receiving 2 bytes of data
closing bootloader
```

Second Project - Heat Sensor

- Need a Seebeck Element (heat absorbing or endothermic side = no writing) + 2 banana clips + 2 wires
- Red wire to the ESP8266 Analog to Digital Converter pin (labelled ADC, next to GND)
- Black wire to the ground rail
- File > Examples > Basics > AnalogReadSerial
- Upload to ESP8266
- Tools > Serial Monitor to view ADC values change when you heat the sensor with your hand
- The serial speed of the monitor must match the serial speed set in the program or it will appear garbled. 115200 baud?
- Can you alter the program to make the heat sensor turn the LED on and off?

Third Project - Light Sensor

- Cadmium Sulphide (CdS) light sensitive resistor (250k - 10M) + 220k resistor (or higher) + one wire
- Attach one end of the resistor to the positive rail, and the other end to one side of the CdS
- The other side of the CdS goes to the ground rail (forming a 'voltage divider' - the ESP8266 ADC limit is 1 Volt)
- Attach one wire from the resistor junction to the ADC
- File > Examples > Basics > AnalogReadSerial
- Upload to ESP8266
- Tools > Serial Monitor to view ADC values
- Can you alter the program to make the light sensor turn the LED on and off?

Fourth Project - Talk to your device

- File > Examples > ESP8266WiFi > WiFiAccessPoint
- Configure SSID and Password (blank or 8 char min)
- Flash to ESP8266
- Try connecting to the hotspot and browse to 192.168.4.1
- Can you read the sensor values or turn the LED on and off by browsing to different hyperlinks?

Thanks!

- Brian Sutherland
- b.sutherland@utoronto.ca