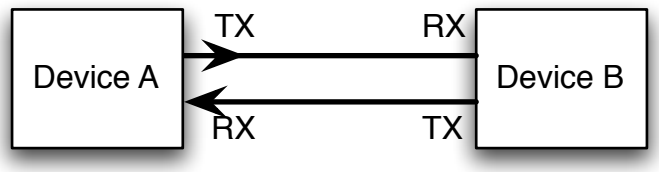
Serial Communication

Asynchronous communication

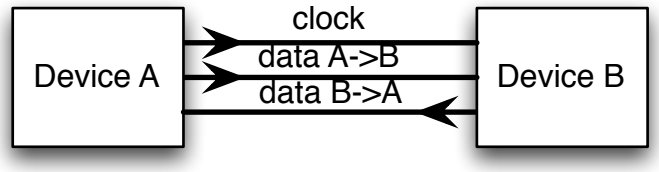


asynchronous – no clock
Data represented by setting HIGH/LOW at given times

Separate wires for transmit & receive

Each device must have good "rhythm"

Synchronous communication



Synchronous – with clock
Data represented by setting HIGH/LOW when "clock" changes

A single clock wire & data wire for each direction like before

Neither needs good rhythm, but one is the conductor

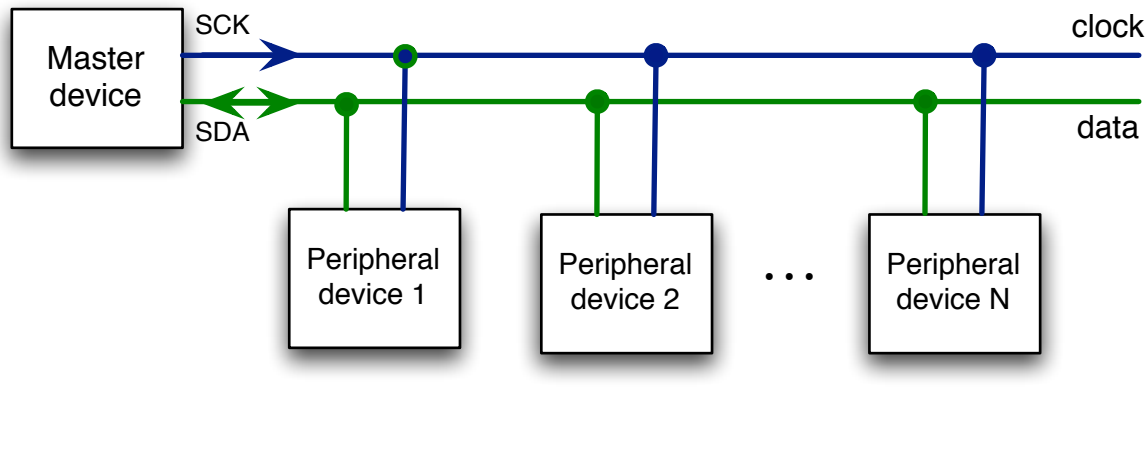
Is one better than the other? It depends on your application. Async is good if there are only two devices and they're both pre-configured to agree on the speed (like your Arduino sketches)

Synchronous is generally better for faster speeds (because you don't need an accurate clock, just the ability to watch the clock wire).

I2C, aka "Two-wire"

Synchronous serial bus with shared a data line

a little network for your gadgets

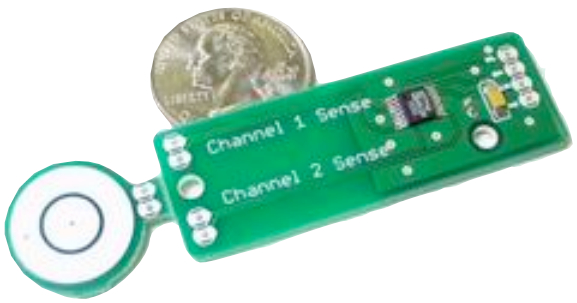


- Up to 127 devices on one bus
- Up to 1Mbps data rate
- Really simple protocol (compared to USB,Ethernet,etc)
- Most microcontrollers have it built-in

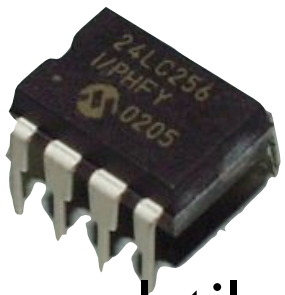
The shared data line means the devices have to agree on when they should "talk" on it. Like how on CBs you say "over" and "over & out" to indicate you're finished so the other person talk.

See "Introduction to I2C": http://www.embedded.com/story/OEG20010718S0073
"I2C" stands for "Inter-Integrated Circuit", but no one calls it that
And if your microcontroller doesn't have I2C hardware built-in, you can fake it by hand in software (for master devices anyway)

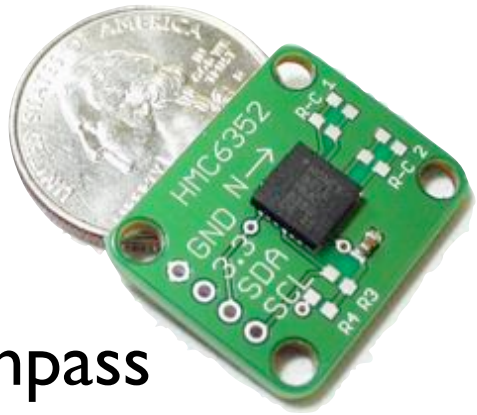
Many I2C devices



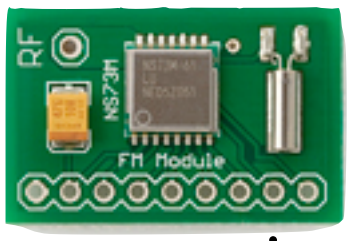
touch sensor



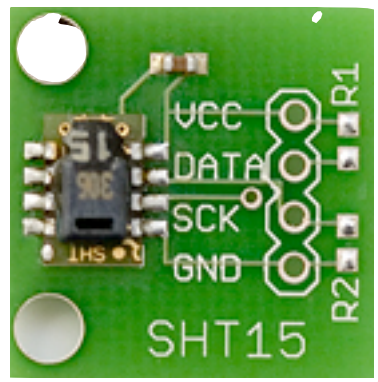
non-volatile memory



compass



fm transmitter



temperature & humidity sensor

LCD display

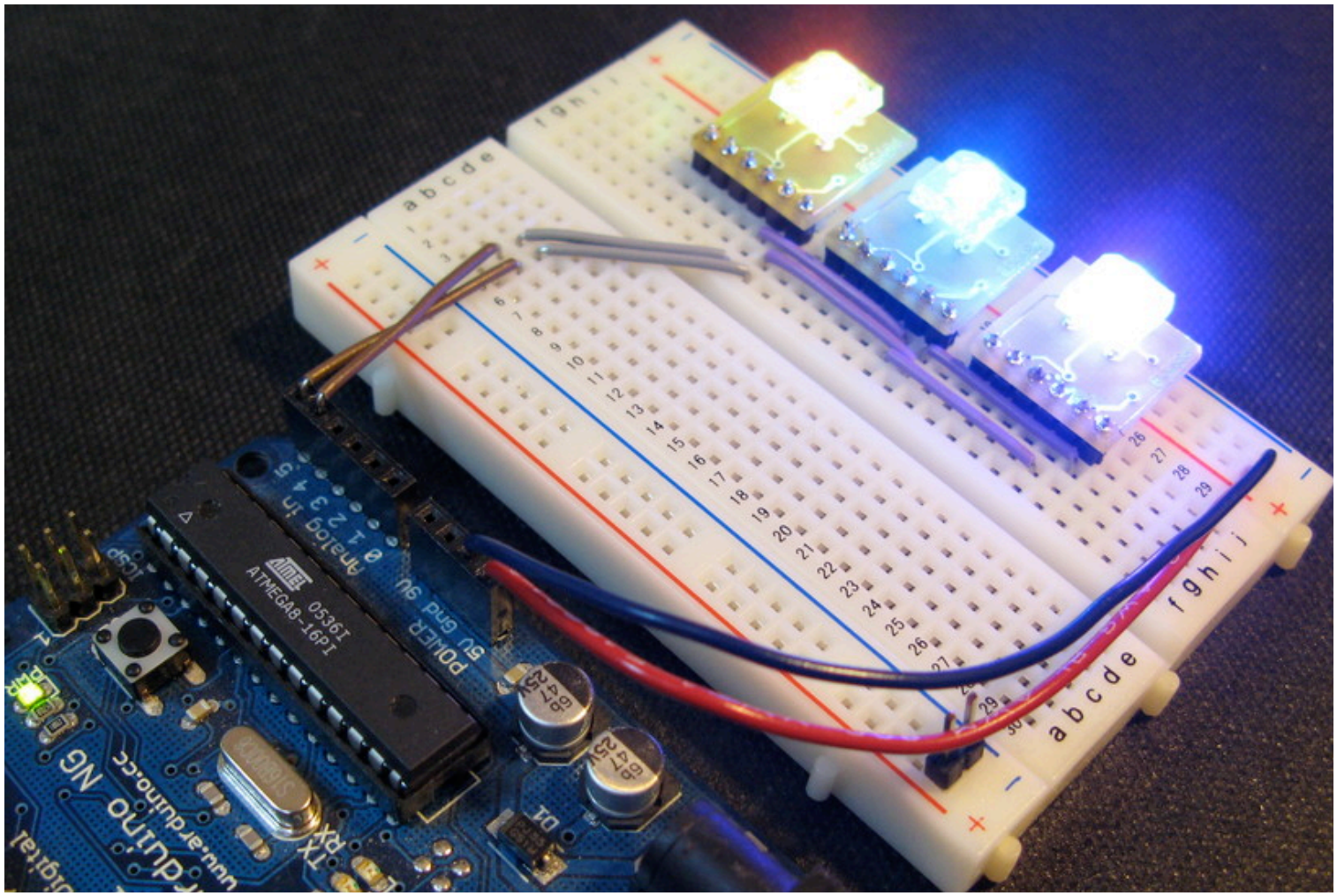
And many others

(gyros,keyboards, motors,...)

Images from Sparkfun.com,except LCD from matrixorbital.com

Obligatory BlinkM Promo

I2C Smart LED



Does all the hard PWM & waveform generation for you

You should be able to buy these from Sparkfun.com in a month or so.

Nintendo Wii Nunchuck

- Standard I2C interface
- 3-axis accelerometer with 10-bit accuracy
- 2-axis analog joystick with 8-bit A/D converter
- 2 buttons
- $20

If you look at the architecture for the Nintendo Wii and its peripherals, you see an almost un-Nintendo adherence to standards. The Wii controllers are the most obvioius examples of this. The Wii controller bus is standard I2C. The Wii remote speaks Bluetooth HID to the Wii (or your Mac or PC)

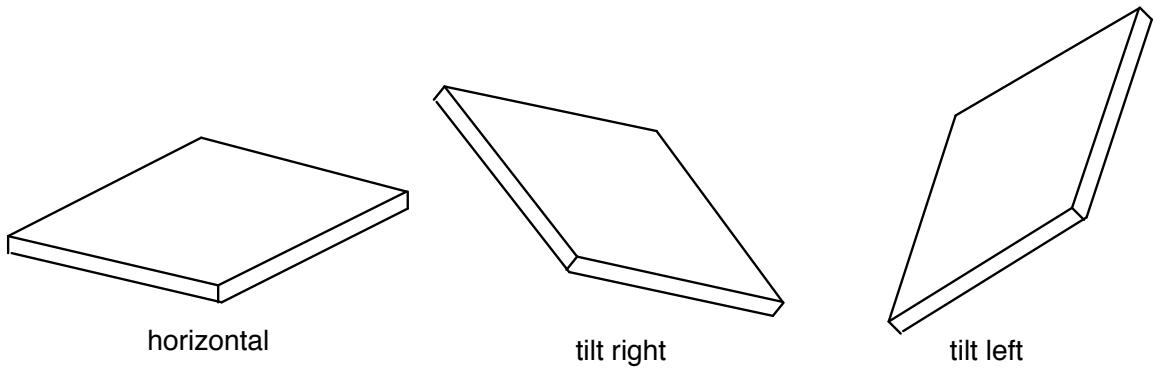
Because it uses standard I2C, it's easy to make the Nunchuck work with Arduino, Basic Stamp or most other microcontrollers.

See: http://www.wiili.org/index.php/Wiimote/Extension_Controllers/Nunchuk
and: http://www.windmeadow.com/node/42
and: http://todbot.com/blog/2007/10/25/boarduino-wii-nunchuck-servo/

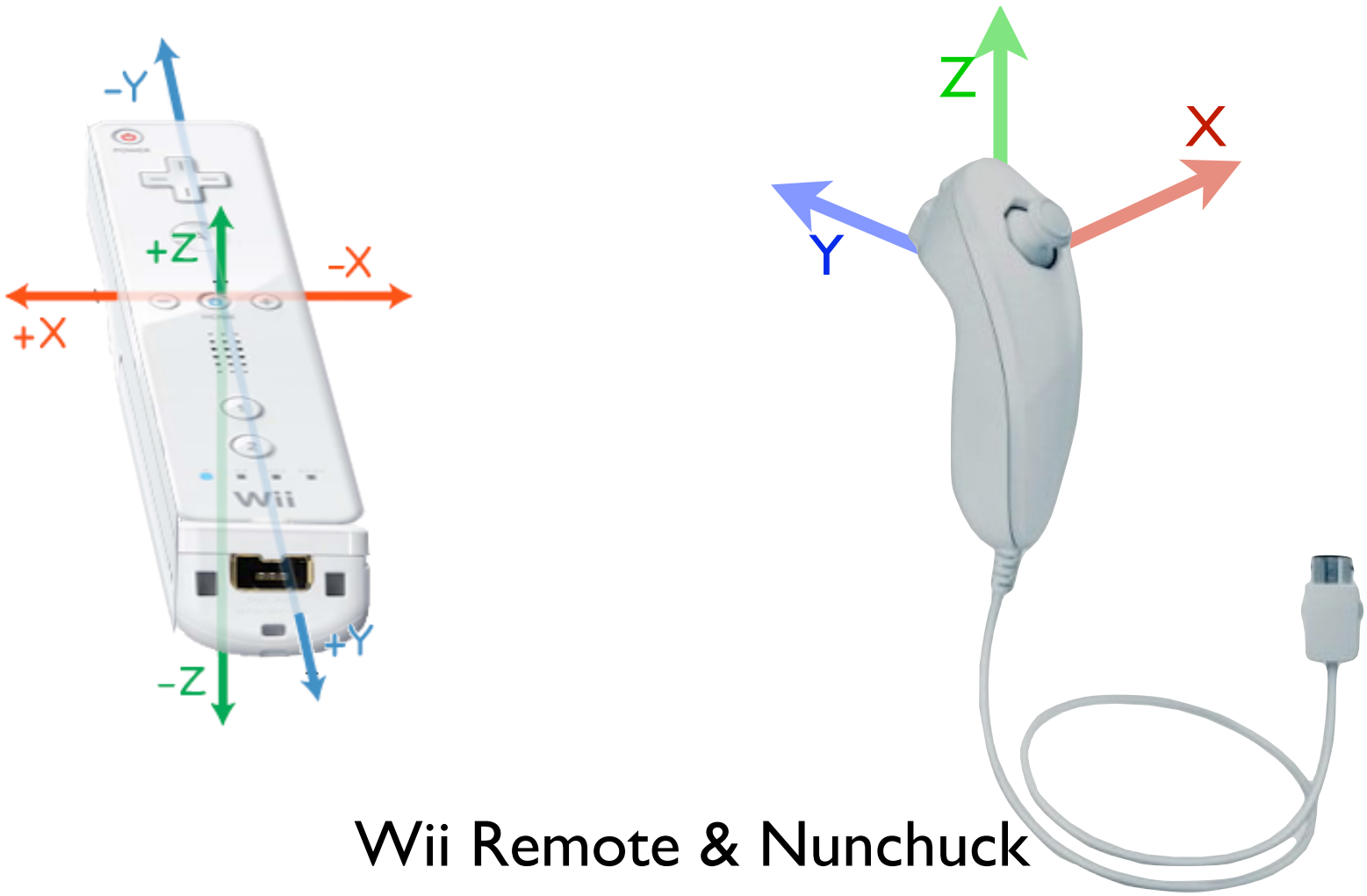
And then there's the Wii Remote, besides Bluetooth HID, it also has accelerometers, buttons, speaker, memory, and is I2C master.

Accelerometer?

- Measures acceleration (changes in speed)
- Like when the car pushes you into the seat
- Gravity is acceleration
- So, also measures tilt



Nunchuck Accelerometer



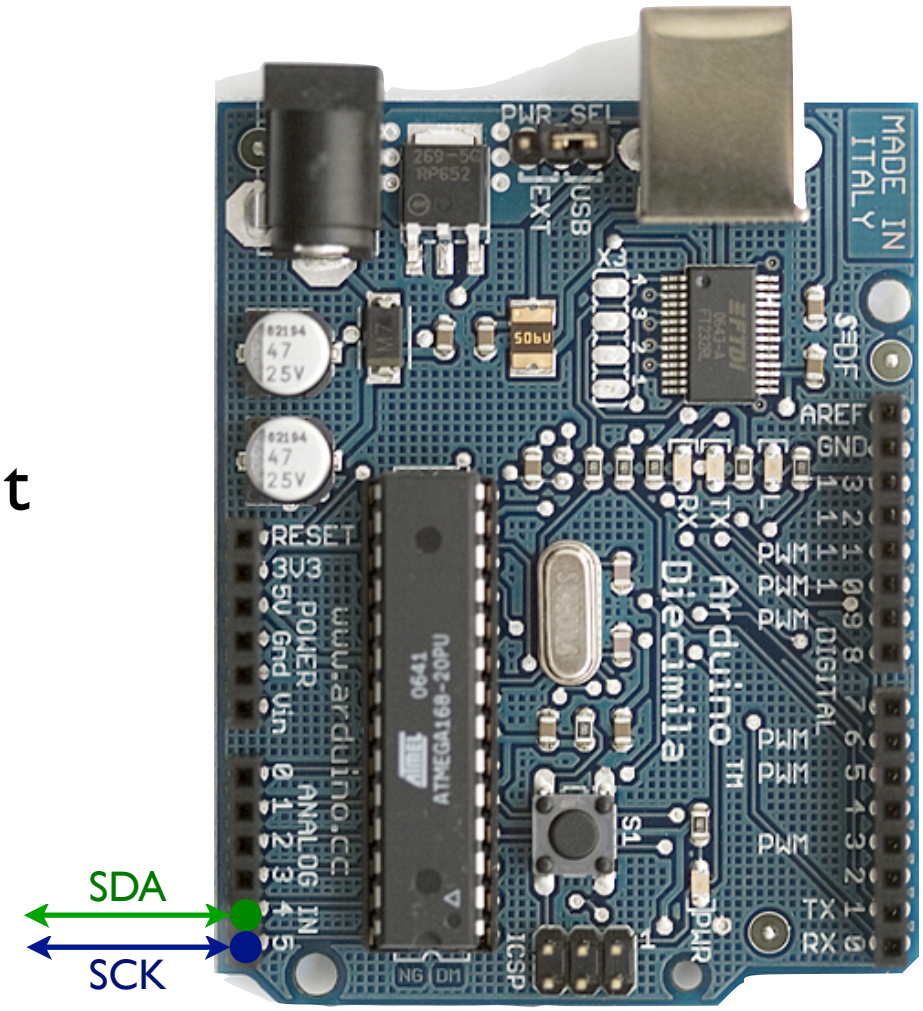
Wii Remote & Nunchuck
accelerometer axes

I'm not sure if I have the Nunchuck one right.

Wiimote axis image from http://www.wiili.org/index.php/Wiimote

I2C on Arduino

- I2C built-in on Arduino's ATmega168 chip
- Use "Wire" library to access it
- Analog In 4 is SDA signal
- Analog In 5 is SCK signal



Arduino "Wire" library

Writing Data

Load Wire library

Join I2C bus (as master)

Start sending

Send data

Stop sending

```
#include <Wire.h>

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
}

byte x = 0;

void loop() {
  Wire.beginTransmission(4); // transmit to device #4
  Wire.send("x is ");        // sends five bytes
  Wire.send(x);              // sends one byte
  Wire.endTransmission();    // stop transmitting

  x++;
  delay(500);
}
```

And what the various commands do are documented in the instructions / datasheet for a particular device.

Arduino “Wire” library

Reading Data

Join I2C bus
(as master)

Request data from device

Get data

```
#include <Wire.h>

void setup() {
  Wire.begin();        // join i2c bus (address optional for master)
  Serial.begin(9600);  // start serial for output
}

void loop() {
  Wire.requestFrom(2, 6);    // request 6 bytes from slave device #2

  while(Wire.available()) {   // slave may send less than requested
    char c = Wire.receive(); // receive a byte as character
    Serial.print(c);         // print the character
  }

  delay(500);
}
```

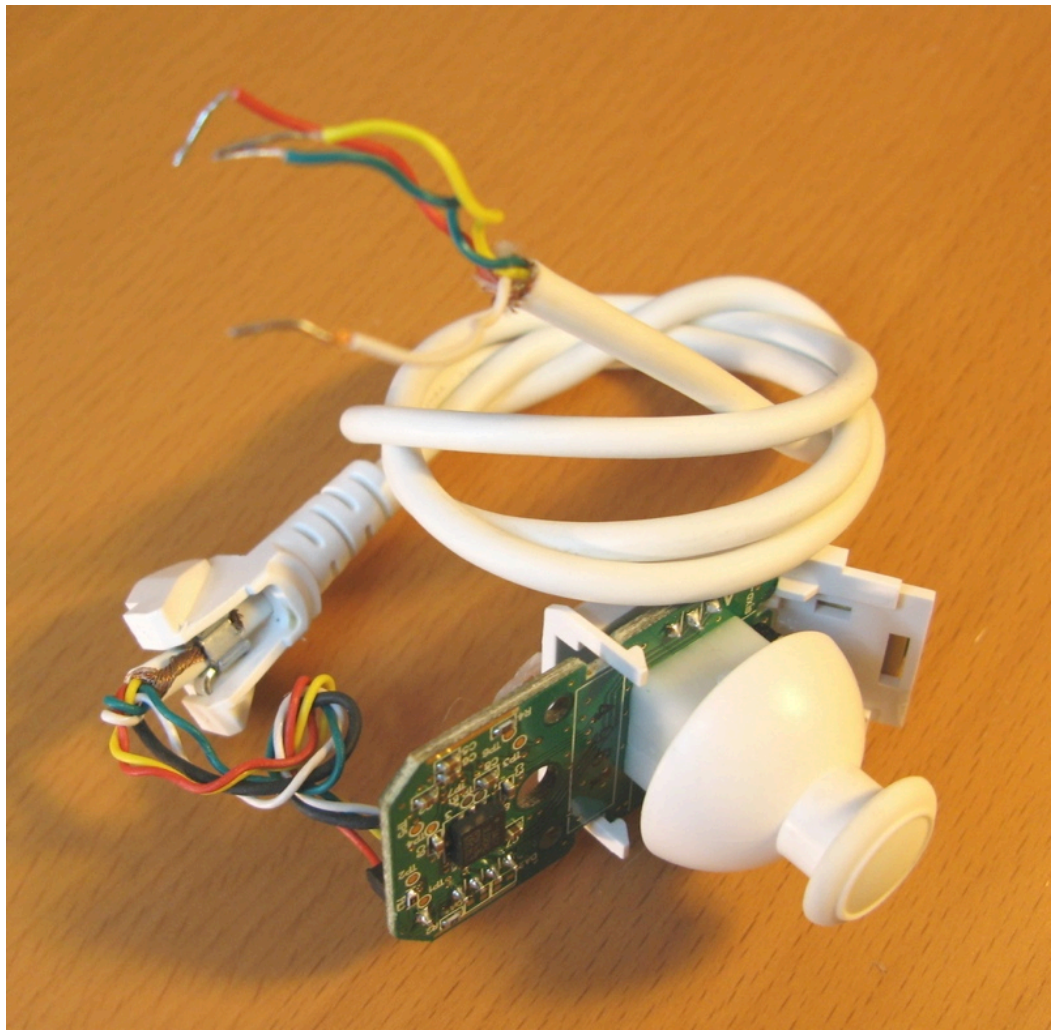
What kinds of interactions you can have depends on the device you’re talking to

Most devices have several “commands”

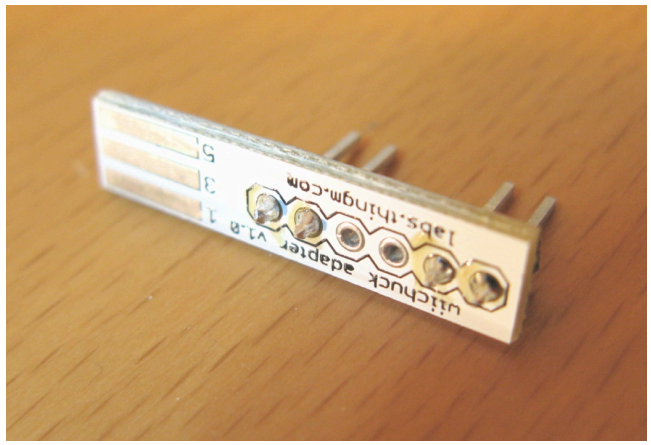
And what the various commands do are documented in the instructions / datasheet for a particular device.

Wiring up the Nunchuck

We could hack off the connector and use the wires directly

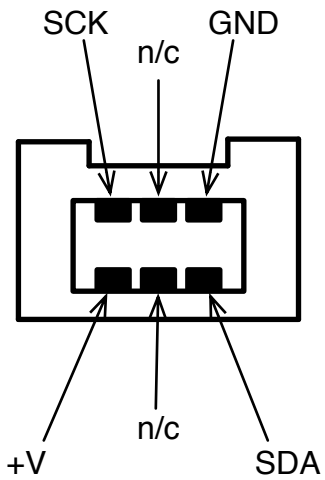


But instead let's use this little adapter board



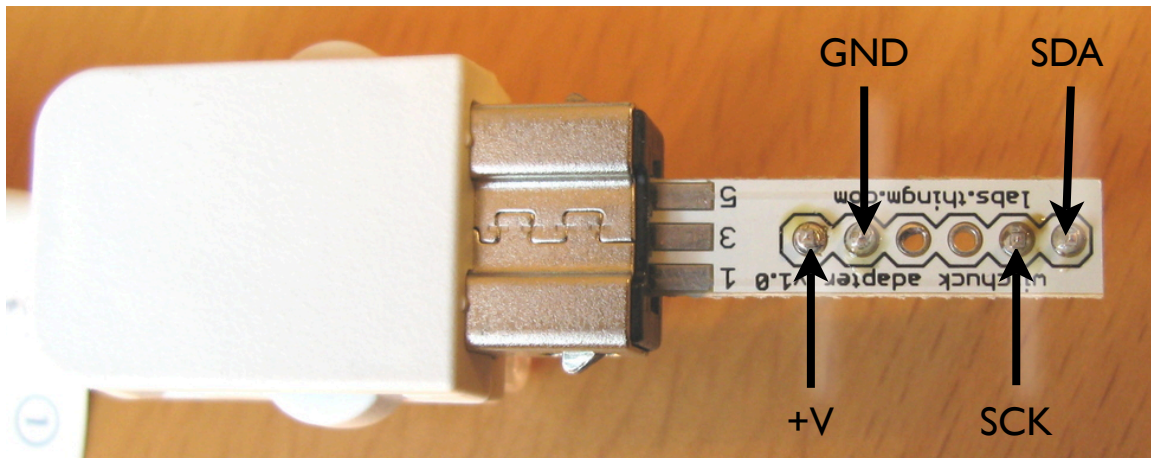
Wii Nunchuck Adapter

Nunchuck Pinout



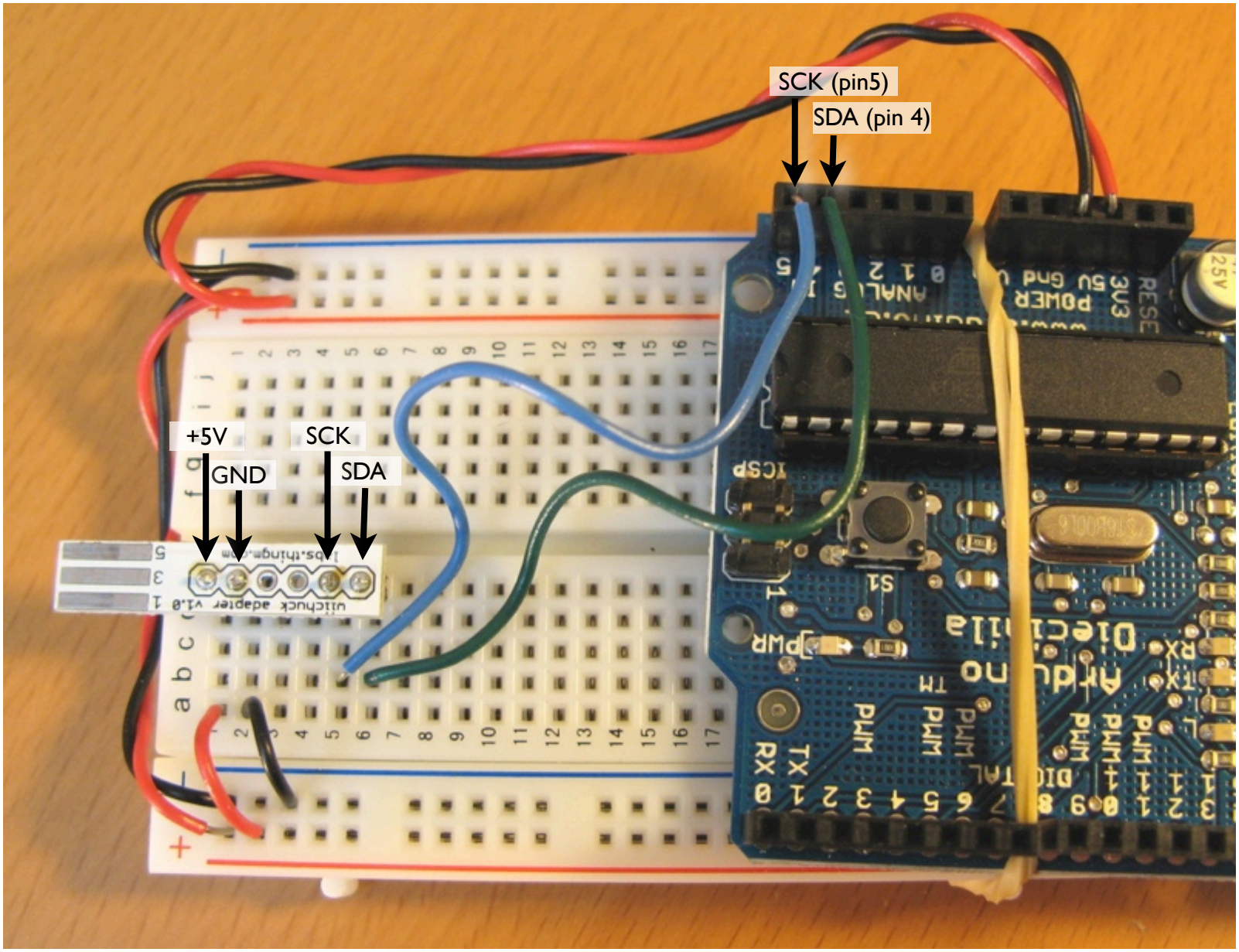
(looking into Nunchuck connector)

Adapter Pinout

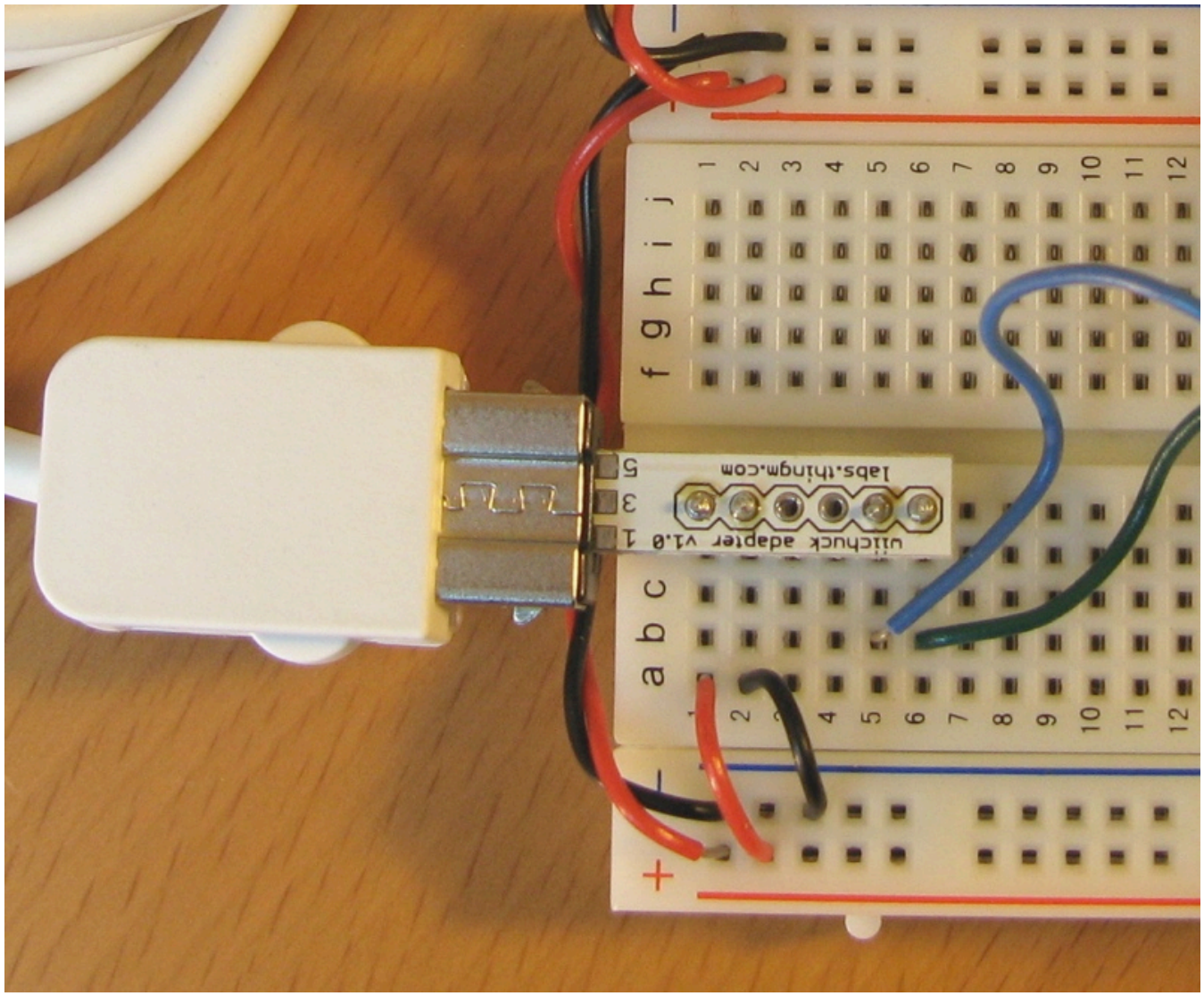


Note there *are* labels on the adapter, but they're wrong. So you'll have to trust the diagrams above

Wiring it Up



Pluggin' in the 'chuck

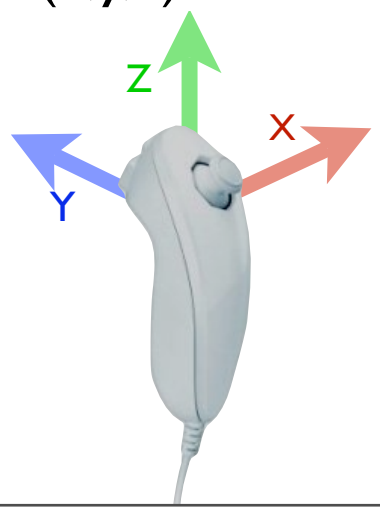


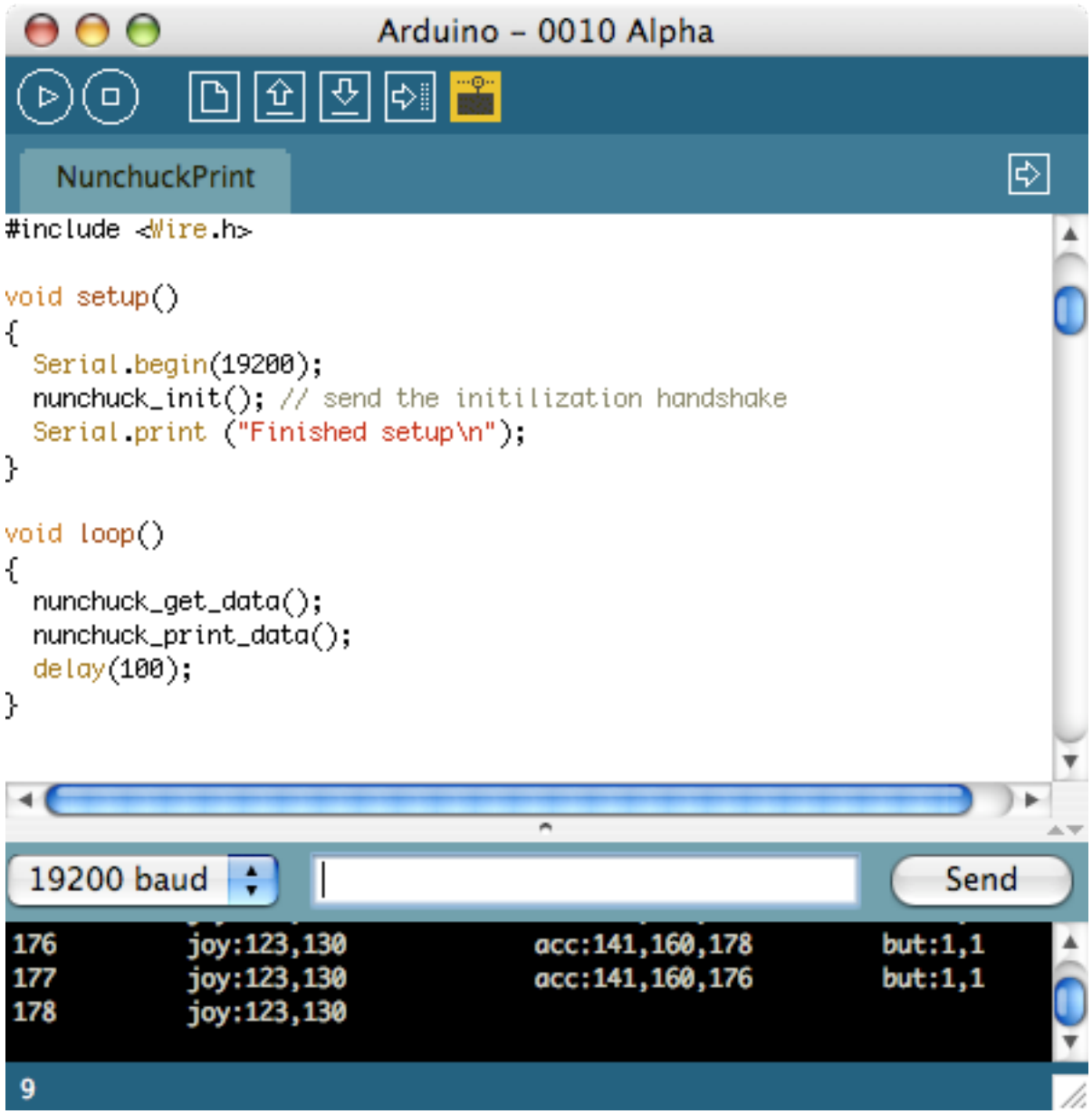
Trying the Nunchuck

"NunchuckPrint"

Read the Nunchuck every 1/10th of a second & print out all the data:

- joystick position (x,y)
- accelerometer (x,y,z)
- buttons Z,C





Uses the beginnings of an Arduino library I'm writing.

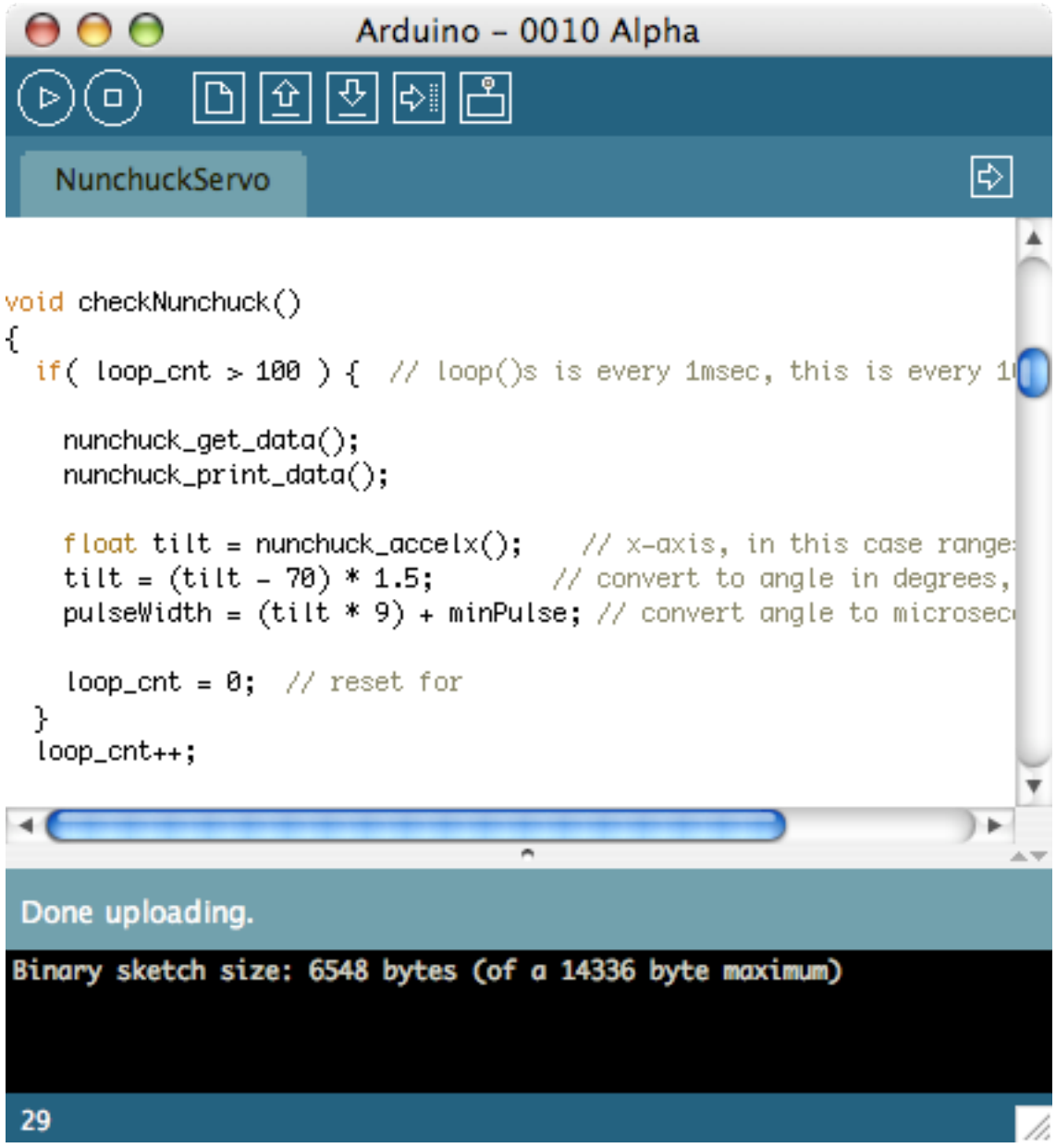
Adding a Servo

"NunchuckServo"

Move the servo by moving your arm

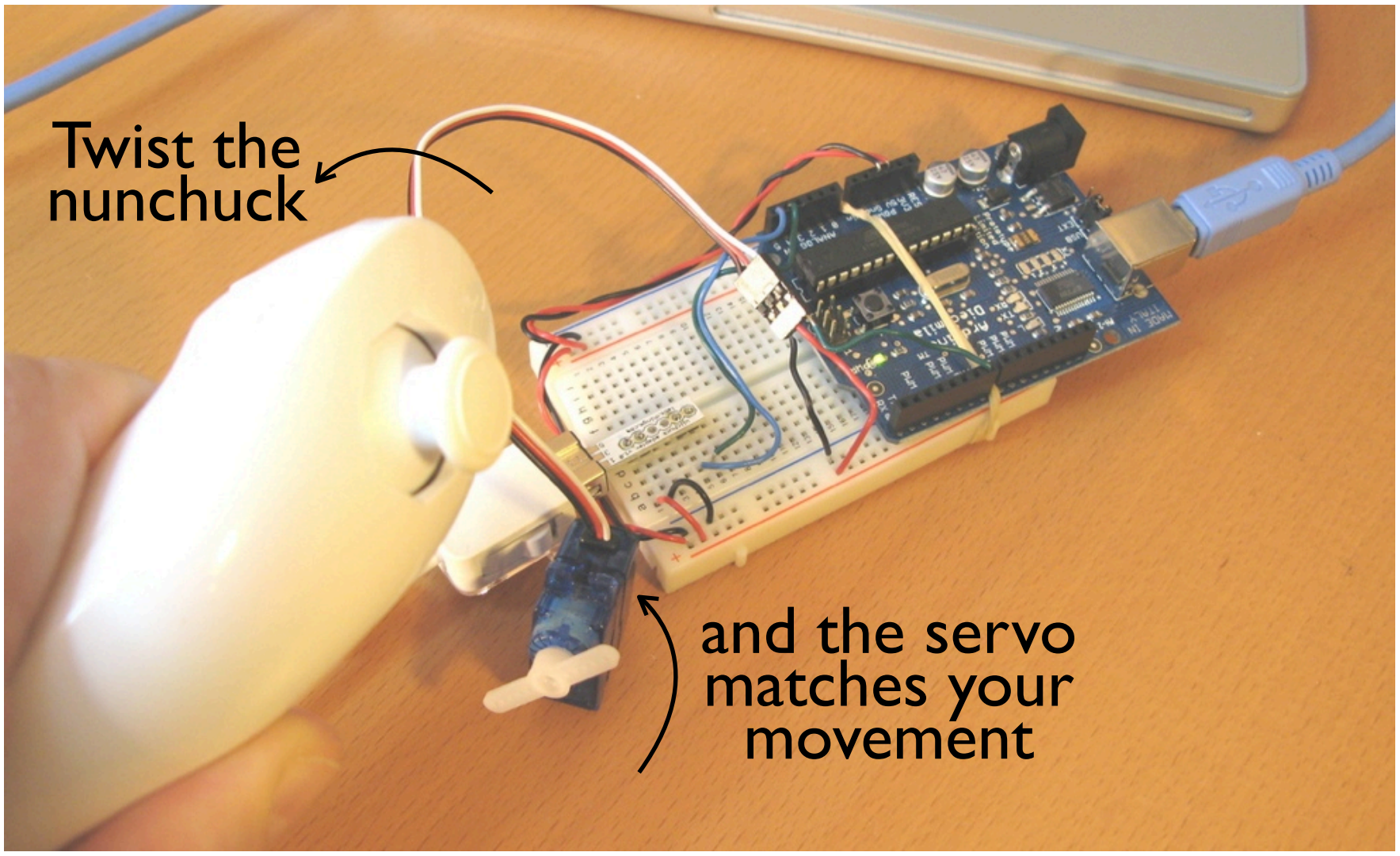
You're a cyborg!

Also press the Z button to flash the pin 13 LED

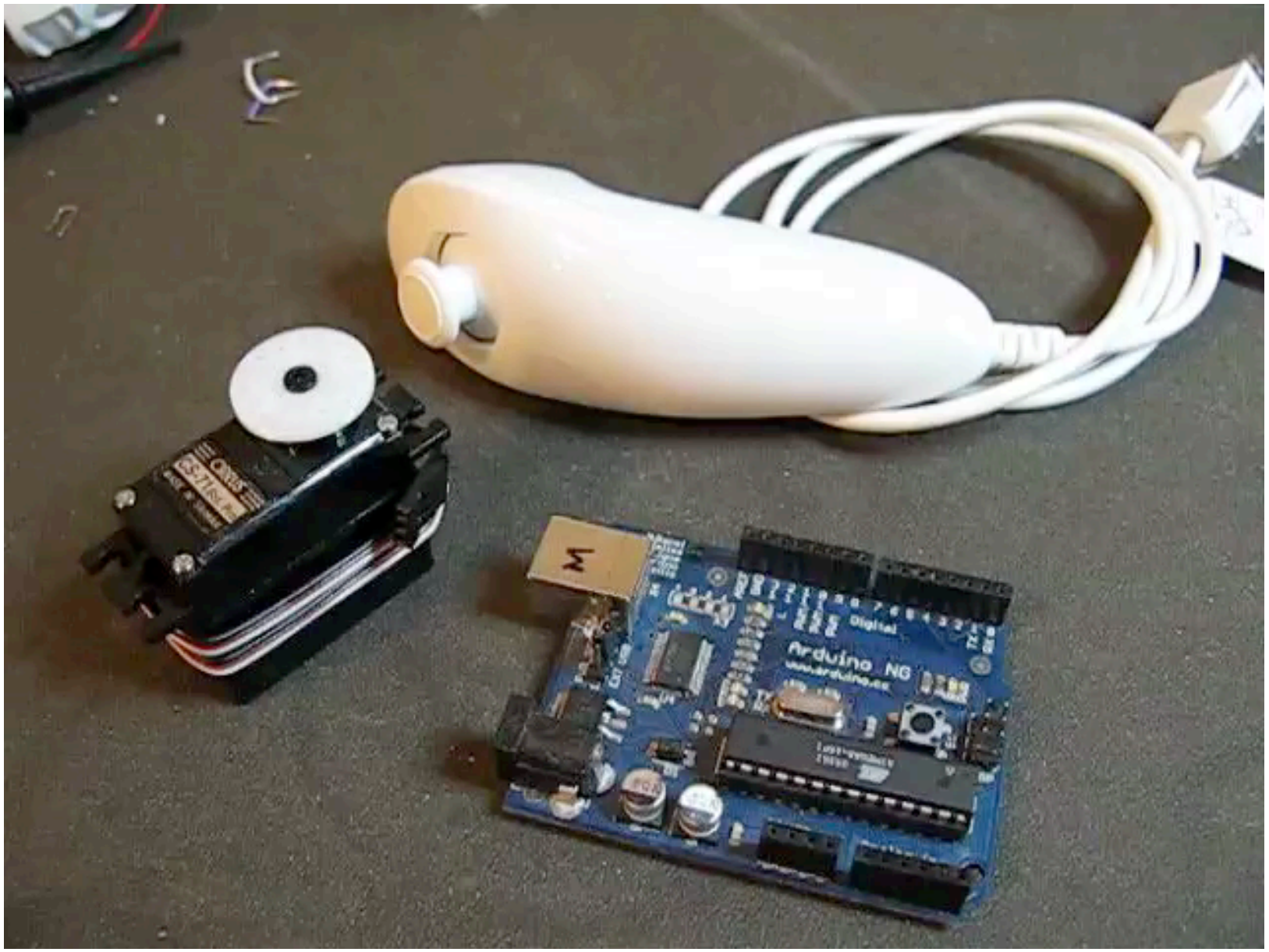


Utilizes the task slicing mentioned before

Nunchuck Servo



Segway Emulator



Same basic code as NunchuckServo.
For details see: http://todbot.com/blog/2007/10/25/boarduino-wii-nunchuck-servo/

Going Further

- Servos
  - Hook several together to create a multi-axis robot arm
  - Make a “servo recorder” to records your arm movements to servo positions and plays them back
  - Great for holiday animatronics

Going Further

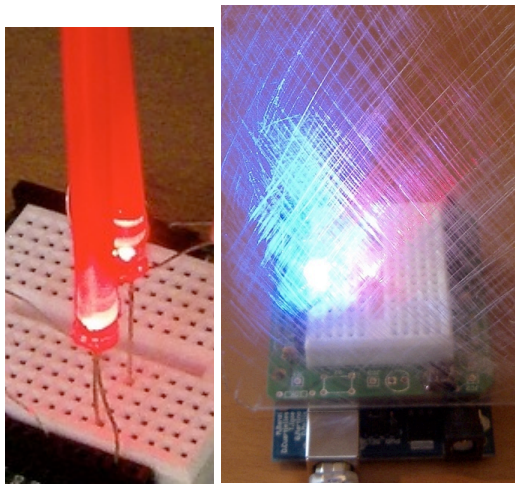
- I2C devices
  - Try out some other devices
  - Just string them on the same two wires used for the Nunchuck
- Cooperative Multitasking
  - Try making a theremin with nunchuck & piezo
  - See if previous examples can be made more responsive

Going Further

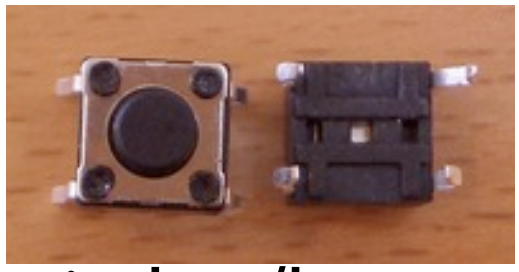
- Nunchuck
  - It's a freespace motion sensor. Control anything like you're waving a magic wand!
  - What about the joystick? We didn't even get a chance to play with that
  - Alternative input device to your computer: control Processing, etc.

Summary

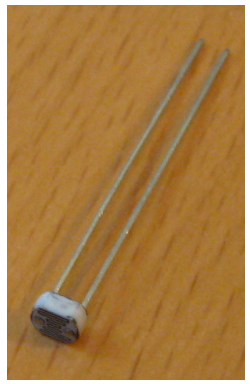
You've learned many different physical building blocks



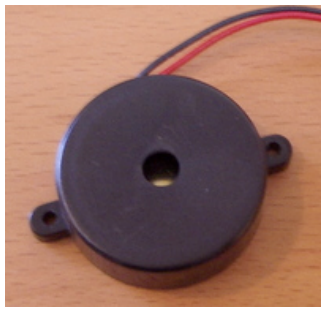
LEDs



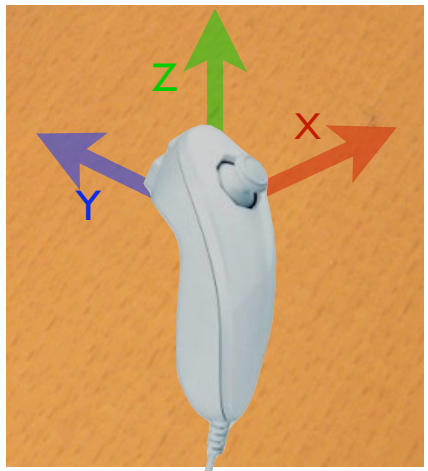
switches/buttons



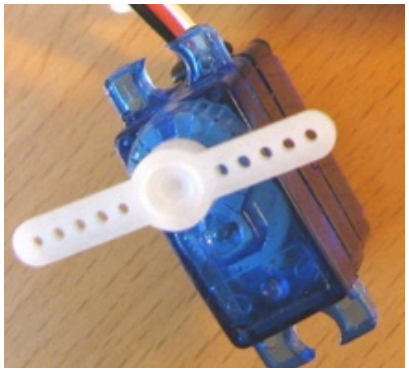
resistive sensors



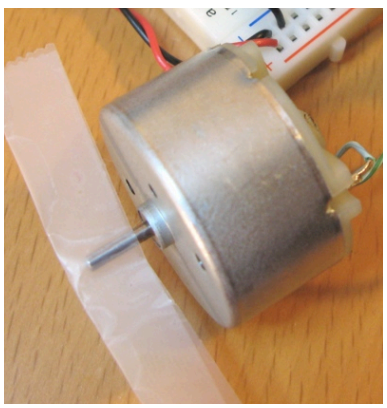
piezos



accelerometers



servos



motors

Summary

And you've learned many software building blocks

pulse width modulation

serial communication

I2C

analog I/O

data driven code

digital I/O

frequency modulation

multiple tasks

Summary

Hope you had fun and continue playing with Arduino

Feel free to contact me to chat about this stuff

END Class 4

http://todbot.com/blog/bionicarduino/

Tod E. Kurt

tod@todbot.com

Feel free to email me if you have any questions.