Bits & Bots

Anna Gerber

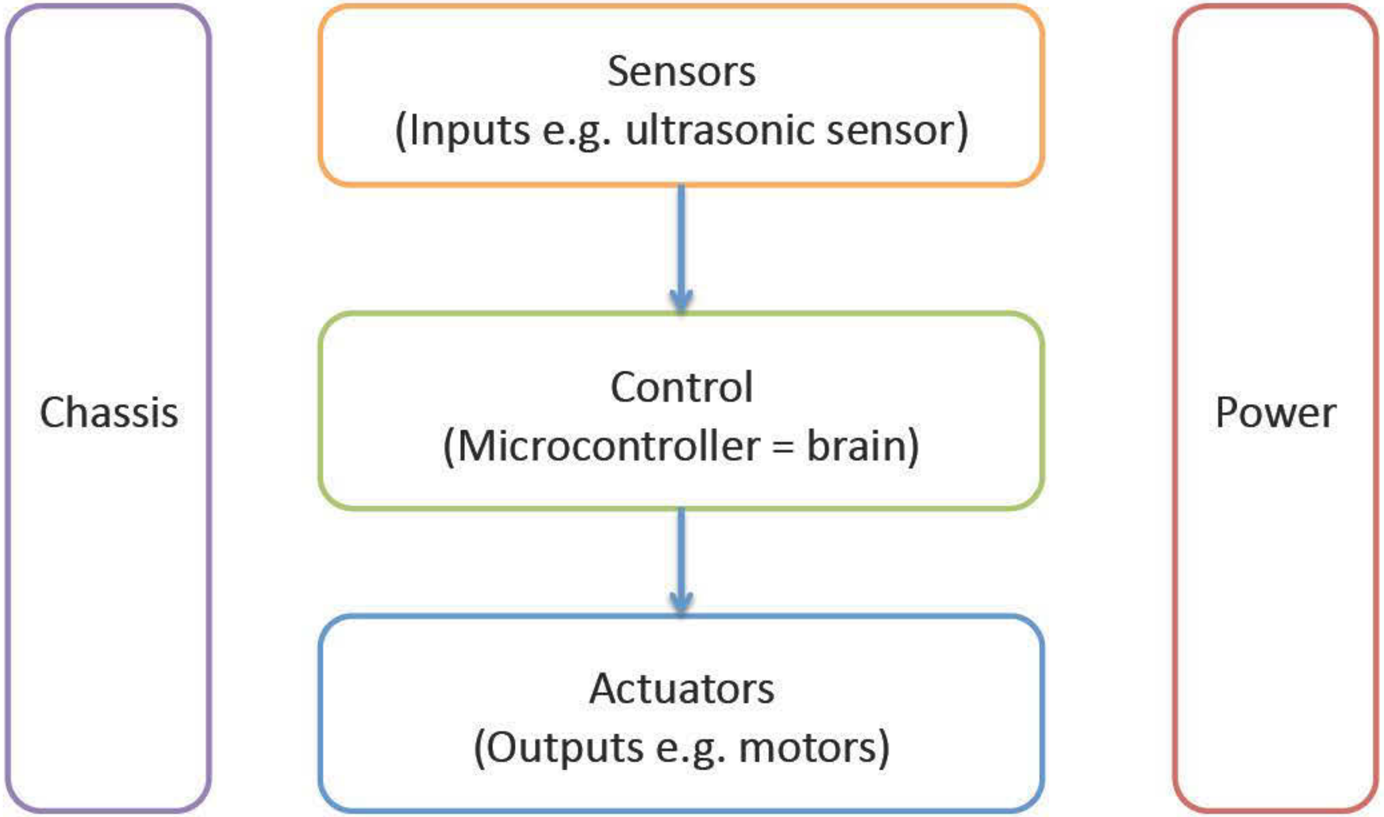
Bits & Bots Sessions

Session	Topic
Tuesday 20th May, 6 – 8pm	**Intro to 3D Design**: Design custom robot parts to print on the 3D printers
Tuesday 27th May, 6 – 8pm	**Intro to Electronics**: Learn how the electronic parts in the kit work, design our robot circuits
Tuesday 3rd Jun,e 6 – 8 pm	**Intro to Arduino**: Write NodeJS programs to read from sensors and control actuators
7th June, 1 – 5 pm	**Intermediate 3D Design**: Design more complex robot parts: gears, claws etc
14th June, 1 – 5 pm	**Intermediate Arduino**: Develop our robots' locomotion, sensing and responding behaviours
21st June, 1 – 6 pm	**Advanced Bits & Bots**: Finalise robot design and assembly, develop advanced robot control programs

Bits & Bots Slides etc

Slides and other materials for the course will be published after each session here:

https://github.com/AnnaGerber/bits-n-bots



Chassis

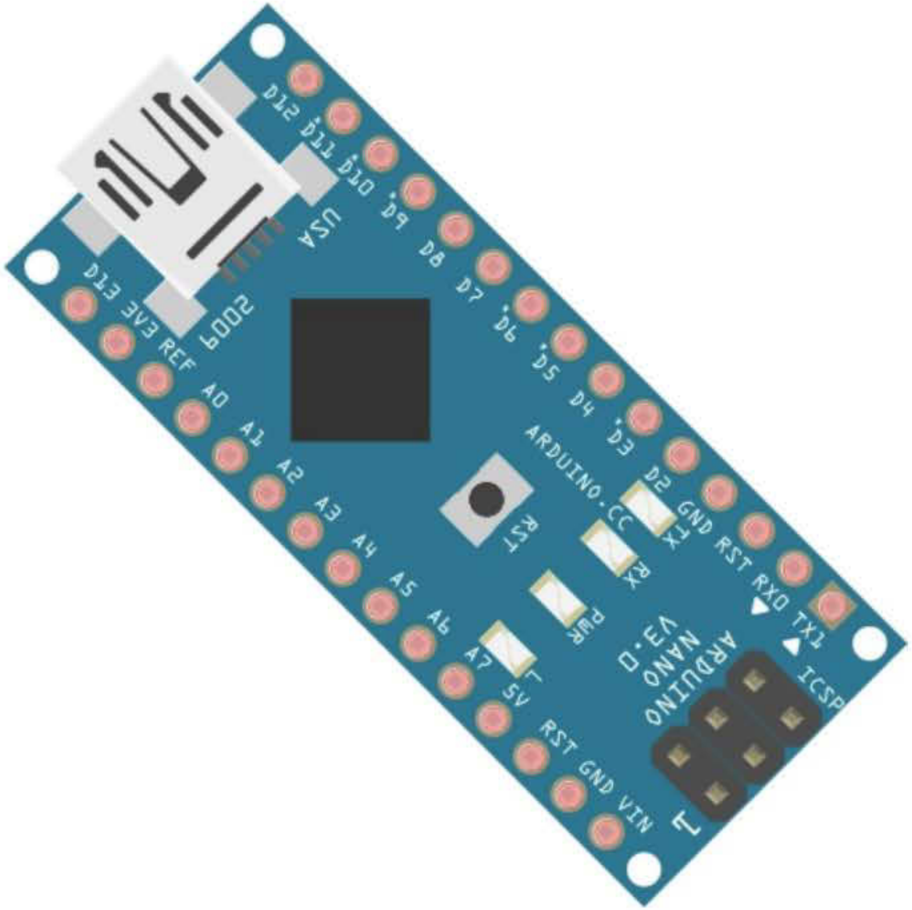
Sensors
(Inputs e.g. ultrasonic sensor)

Control
(Microcontroller = brain)

Actuators
(Outputs e.g. motors)

Power

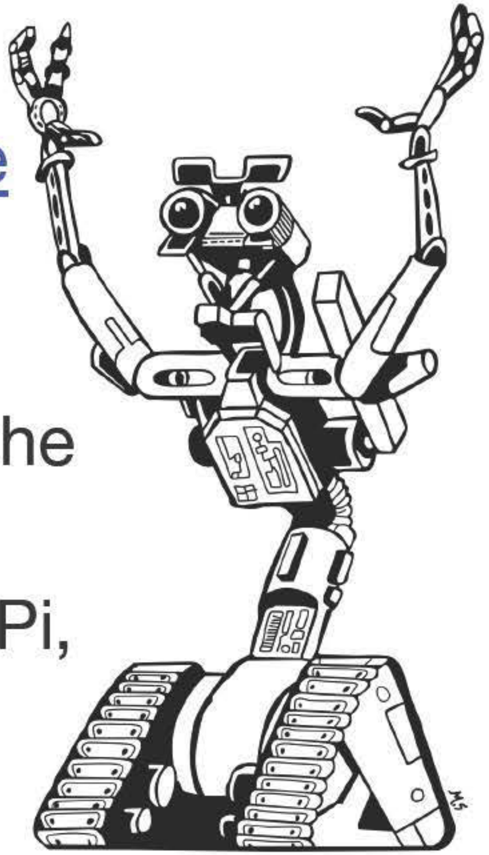
Control



- Microcontroller co-ordinates robot inputs (sensors) and outputs (actuators)
- We are using an Arduino Nano clone
- See http://arduino.cc/

Johnny-Five

- Open Source JavaScript Framework for programming Arduino
- https://github.com/rwaldron/johnny-five
- Works with nodejs, a platform that runs programs using Chrome's JS runtime
- Communicates with the Arduino using the Firmata protocol
- Supports other devices e.g. Raspberry Pi, BeagleBone Black, via I/O Plugins



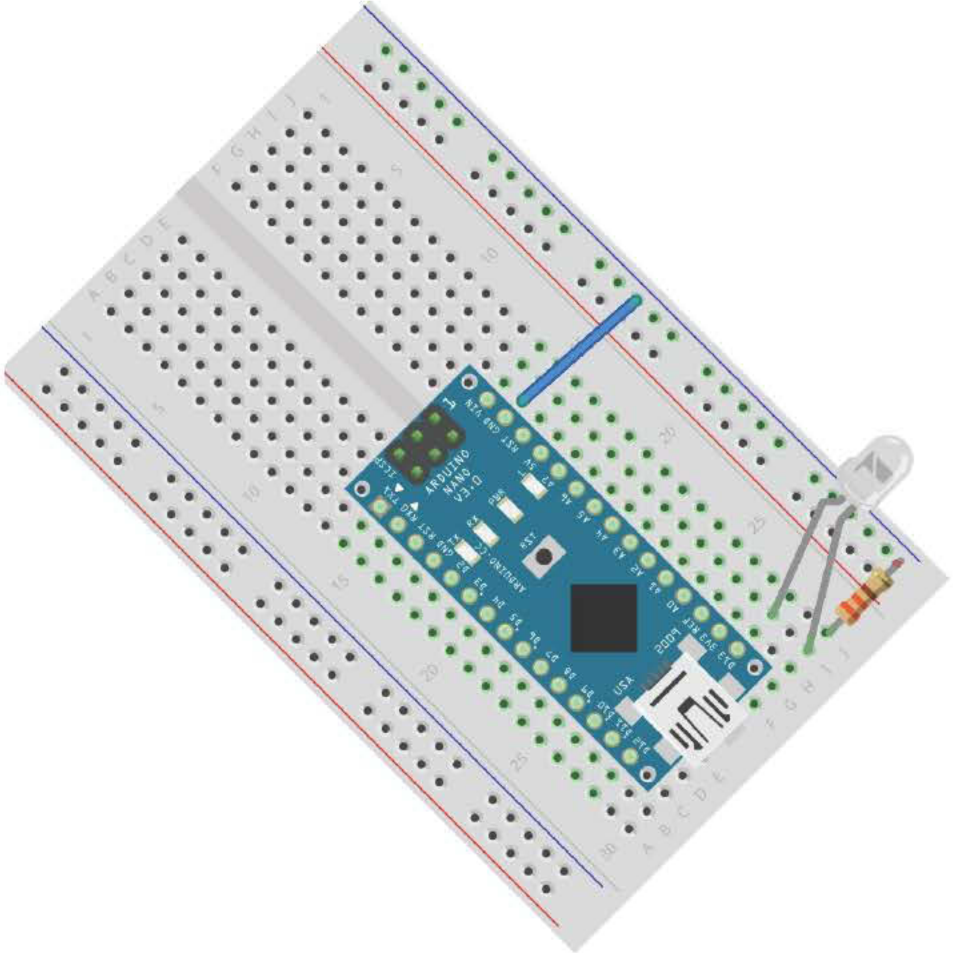
Loading Firmata onto the Arduino

- Once-off setup to prepare our Arduino for use with Johnny-Five:
  - Connect the microcontroller board via USB
  - Launch Arduino IDE and open the Firmata sketch via the menu: `File > Examples > Firmata > StandardFirmata`
  - Select your board type (e.g. Arduino Nano w/ ATmega328) via `Tools > Board`
  - Select the port for your board via `Tools > Serial Port >` (the port of your Arduino) e.g. /dev/tty.usbserial-A9GF3L9D
  - Upload the program by clicking on `Upload`
  - Close the IDE

WORKING WITH ACTUATORS

BLINKING AN LED

Connecting an LED to the Arduino



- Unplug the Arduino!
- Attach long lead of LED to pin 13 of Arduino
- Connect resistor to cathode of resistor and ground rail of breadboard
- Connect GND pin of Arduino to ground rail of breadboard using a jumper wire

Creating the Johnny-Five program

1. Create a JavaScript file (e.g. blink.js)
2. Edit it using a text editor e.g. Atom
3. At the start of your program load the johnny-five library into a variable:

```
var j5 = require("johnny-five");
```

A variable is a named "container" for storing data, including values and functions (reusable blocks of code)

Creating a Board object

JavaScript objects are groupings of properties (state) and functions (behaviour), and in our programs they correspond to sensors, actuators and to the Arduino.

- We can create a Board object which corresponds to our Arduino and store it in a variable.
- The new keyword indicates that we are creating a new object via a constructor function.

Let Johnny-Five autodetect the board:

```
var myBoard = new j5.Board();
```

OR Tell it exactly which board to use:

```
var myBoard = new j5.Board({
  port: "/dev/tty.usbserial-A9GF3L9D"
});
```

Ready event

- When the board is ready for our code to start interacting with it and the attached sensors and actuators, it will trigger a ready event. We can write an event handler (anonymous function) that is run when the event occurs:

```
myBoard.on("ready", function() {
  // code for sensors, actuators goes here
});
```

Controlling the LED

- Then we can start to read from sensors or control actuators attached to the Arduino within our function.

```
// attach LED on pin 13
var myLed = new j5.Led(13);

// call strobe function to blink once per second
myLed.strobe(1000);
```

- We can change the parameter to the strobe function to change the speed: This input value is provided in milliseconds

REPL

- Read, Eval, Print Loop
- A console for real-time interaction with the code
- Expose our variables to the REPL to enable interactive control:

```
// make myLED available as "led" in the REPL
this.repl.inject({
   led: myLed
});
```

- The `this` operator refers to the current execution context, in this case our board

The complete blink program

```
var j5 = require("johnny-five");
var myBoard, myLed;
myBoard = new j5.Board({port: "/dev/tty.usbserial-A9GF3L9D" });
myBoard.on("ready", function() {

  myLed = new j5.Led(13);

  // strobe every second
  myLed.strobe( 1000 );

  // make myLED available as "led" in REPL
  this.repl.inject({
      led: myLed
  });
});
```

Running our program

- Open the Terminal app
- Change directory to the location where you have stored your code e.g.

  ```
  cd ~/Desktop/code/
  ```

- Run your program using node e.g.

  ```
  node blink.js
  ```

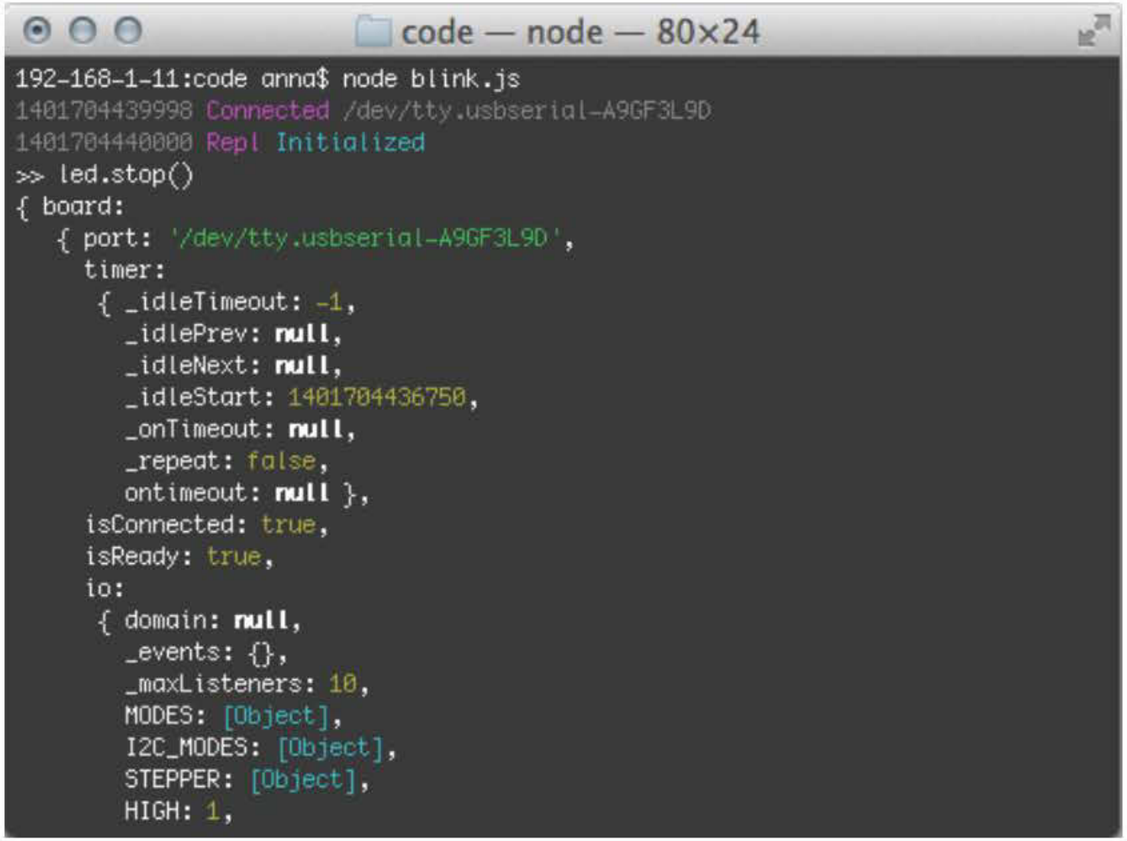
- Hit control-D to stop the program at any time

Controlling the LED via the REPL

- At the REPL prompt type commands followed by enter
- Try:
    - stop,
    - on,
    - off,
    - toggle,
    - strobe

  e.g:

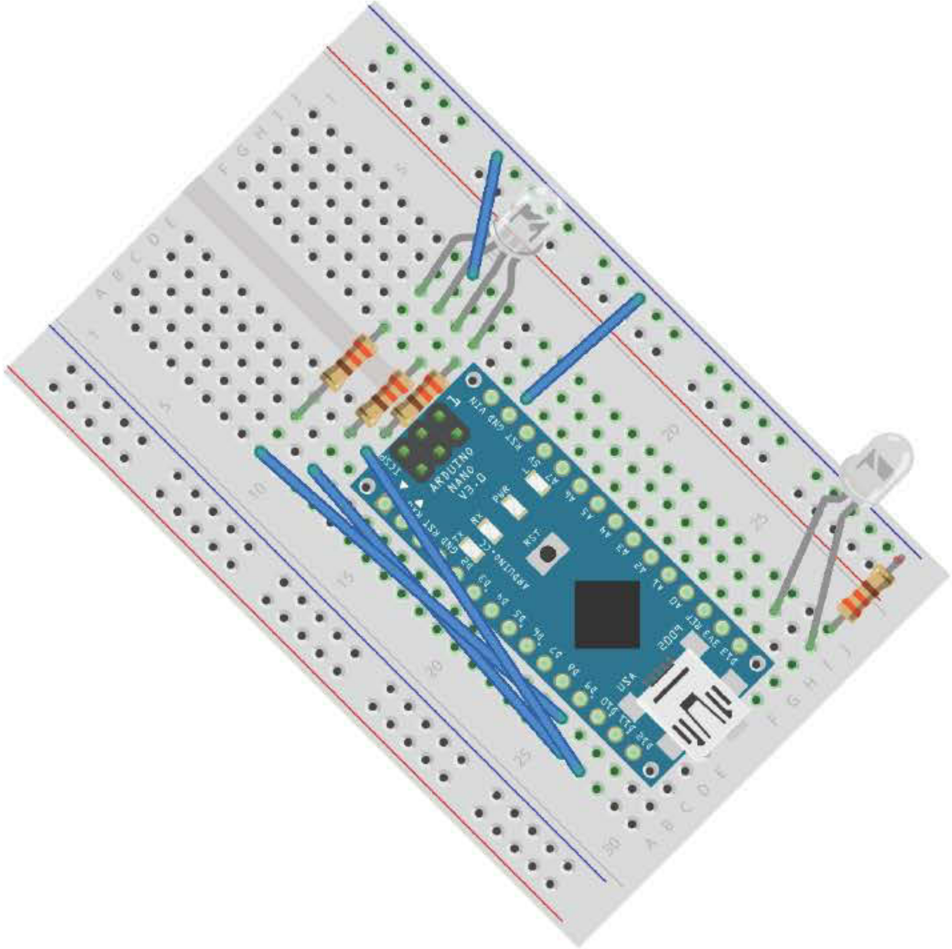
  `>> led.stop()`



```
192-168-1-11:code anna$ node blink.js
1401704439998 Connected /dev/tty.usbserial-A9GF3L9D
1401704440000 Repl Initialized
>> led.stop()
{ board:
   { port: '/dev/tty.usbserial-A9GF3L9D',
     timer:
      { _idleTimeout: -1,
        _idlePrev: null,
        _idleNext: null,
        _idleStart: 1401704436750,
        _onTimeout: null,
        _repeat: false,
        ontimeout: null },
     isConnected: true,
     isReady: true,
     io:
      { domain: null,
        _events: {},
        _maxListeners: 10,
        MODES: [Object],
        I2C_MODES: [Object],
        STEPPER: [Object],
        HIGH: 1,
```

ADDING SOME COLOUR

Add an RGB LED



- Connect the longest lead to the ground rail using a jumper wire
- Connect a resistor to all of the other leads (for Red, Green and Blue) and then use jumper wires to connect the resistors to pins 9, 10 and 11 on the Arduino

Controlling the colour of the LED

- Create an RGB object
- Provide an array of pins for R, G and B as a parameter to the RGB constructor
- Use the `color` function to set the colour (note the American spelling)

```
myBoard.on("ready", function() {
  var myLed = new j5.Led.RGB([ 9, 10, 11 ]);
  // make the LED red
  myLed.color("#ff0000");
});
```

Colours

- The colour codes are set using HEX values (like those used on the web)
- Johnny-Five takes care of the details of sending the right signals to each lead
- The red diode may be brighter than the others, so reduce the value for red, or use a higher value resistor on the red lead to compensate to balance the colours

Colour	Code
White	#FFFFFF
Silver	#C0C0C0
Gray	#808080
Black	#000000
Red	#FF0000
Maroon	#800000
Yellow	#FFFF00
Olive	#808000
Lime	#00FF00
Green	#008000
Aqua	#00FFFF
Teal	#008080
Blue	#0000FF
Navy	#000080
Fuchsia	#FF00FF
Purple	#800080

Delayed behaviour

- Use the wait function to schedule functions to occur a number of milliseconds in the future

```
this.wait( 1000, function() {
  // make the LED blue after 1 second
  myLed.color("#00ff00");
});
```

PWM

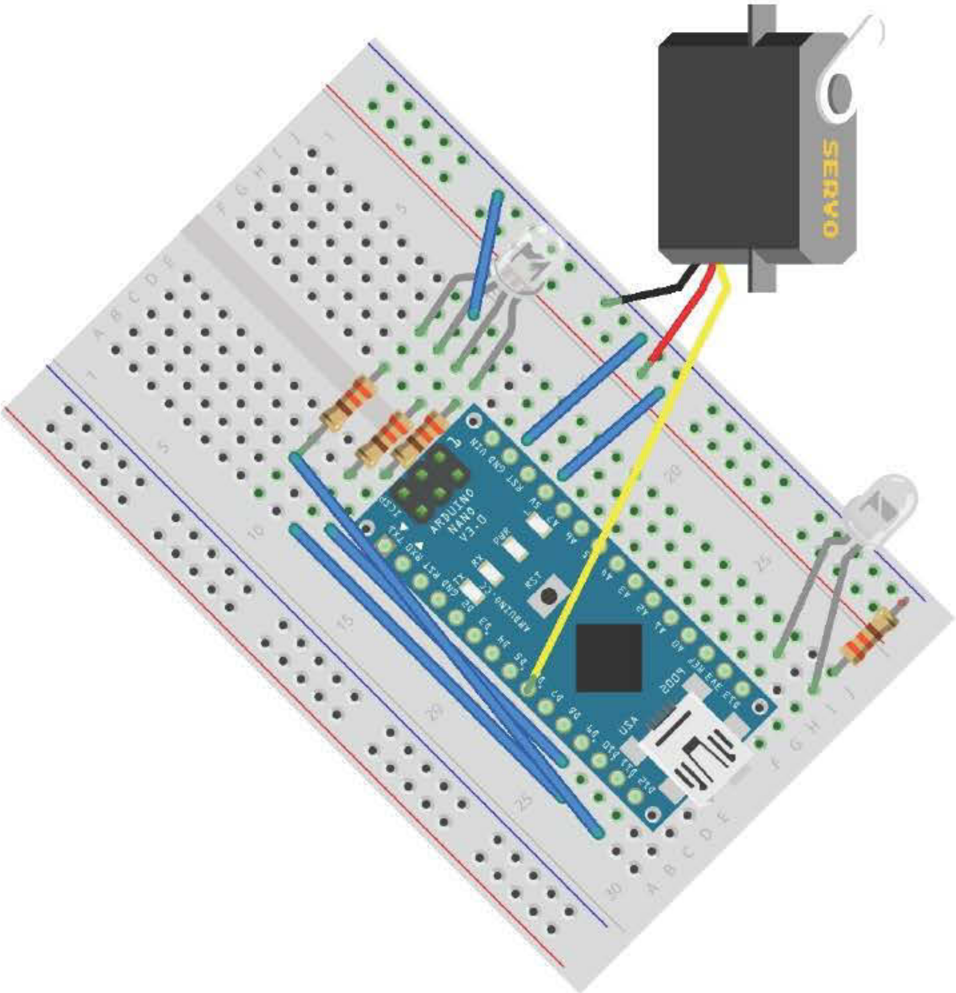
- Pulse Width Modulation
- Produce analog output via digital pins
- Instead of on or off, a square wave is sent to simulate voltages between 0V (off) and 5V (on)
- Used to control motors, fade LEDs etc
- Only enabled for some pins by default
  - 3, 5, 6, 9, 10, 11 on Arduino Nano

Pulsing the LED

- Because the R, G and B leads are connected to PWM pins 9, 10 and 11, we can control the brightness of the LEDs
- Try the following via the REPL or modify your program:
  - r.brightness(100) // set between 0 and 255
  - r.fadeIn(200) // fade in over 200 milliseconds
  - r.fadeOut(500) // fade out over 500 ms
  - r.pulse(1000) // pulse LED over one second

MOVEMENT

Adding a servo



- Add a servo to your circuit:
  - Connect the signal (orange) wire to pin 6
  - Connect the brown wire to ground
  - Connect the red wire to 5V

Creating a Servo object

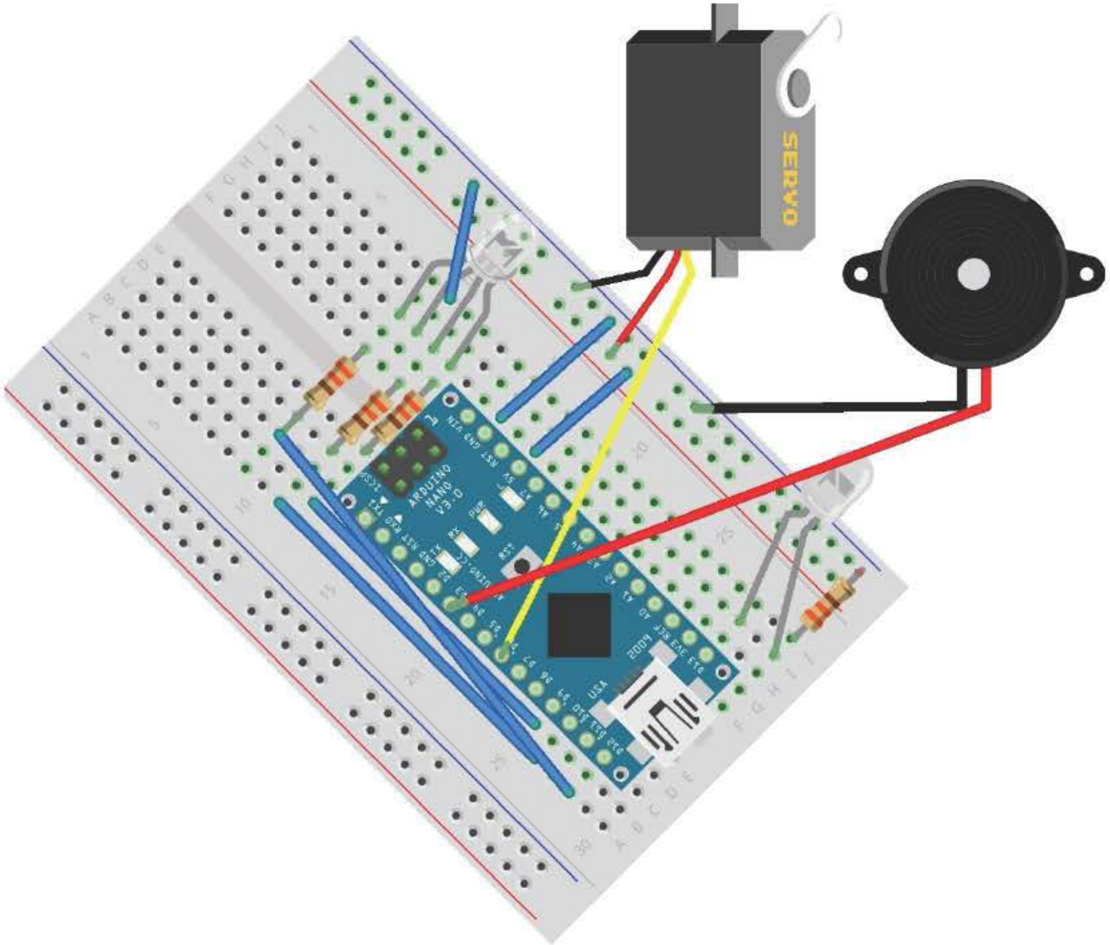
```
var five = require("johnny-five"),
board, myServo;
board = new five.Board();
board.on("ready", function() {
  myServo = new five.Servo(6);

  board.repl.inject({
    servo: myServo
  });

});
```

Controlling the servo

- Try the following commands:
  - `servo.sweep();`
  - `servo.stop();`
  - `servo.center();`
  - `servo.to(20);`
`// move to point in degrees`
  - `servo.min()`
  - `servo.max()`

SOUND

Adding a piezo element



- Add a piezo element
- Connect the ground lead to the ground rail on the breadboard
- Connect the + lead to pin 3 on the Arduino

Controlling the piezo

```
var piezo = new five.Piezo(3);

// notes and durations
// use spaces for rests
piezo.song("ccggaag", "2222224");
```

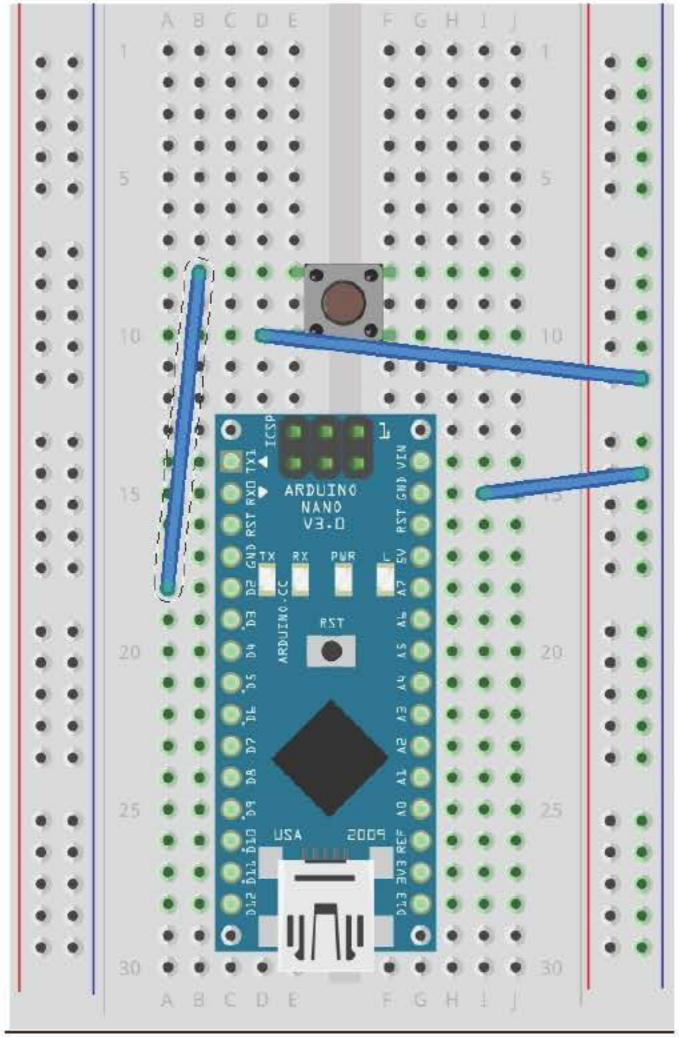
WORKING WITH SENSORS

Logging to the console

- Use the console.log( ) function to print information to the console, e.g. sensor readings
- Use the + operator to combine text-based messages (strings) with variable values e.g.

```
console.log("sensor 1 reading is " +
sensorVal);
```

Buttons


A B C D E
F G H I J
1
5
10
15
20
25
30
ICSP
ARDUINO
NANO
V3.0
TX RX PWR L
RST
ARDUINO.CC
USA
2009
TX1 RX0 RST GND D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
VIN GND RST 5V A7 A6 A5 A4 A3 A2 A1 A0 REF 3V3 D13

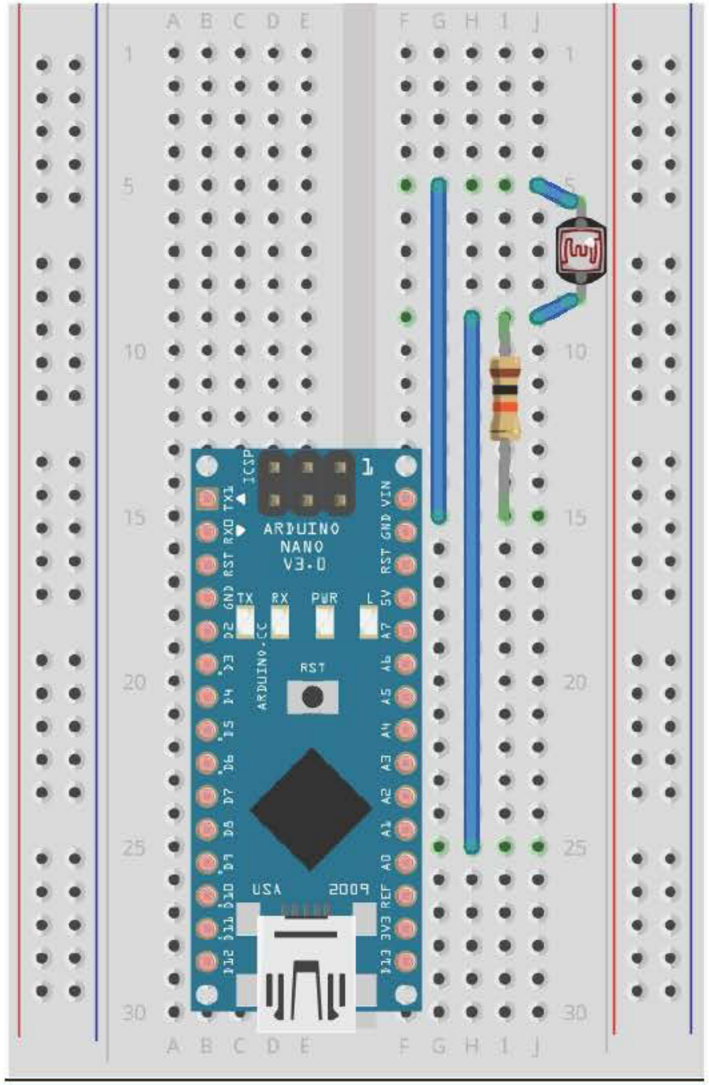
- Connect one button lead to ground and one to pin 2
- We will use a built in "pull-up" resistor. For info on how these work see:
  - http://arduino.cc/en/Tutorial/InputPullupSerial
  - https://learn.sparkfun.com/tutorials/pull-up-resistors
- Use the on-board LED or leave your LED from earlier connected to pin 13

Attaching handlers for button events

- Set the `isPullup` option to true to enable the pull-up resistor on the pin and to invert the input

```
var myButton = new five.Button({
  pin: 2,
  isPullup: true
});

var led = new five.Led(13);

myButton.on("down", function(value){
  console.log("button pressed!");
  led.toggle();
});
```

Adding a photo resistor



- Connect one lead to ground
- Connect the other lead to Analog pin 0
- Connect a 10K resistor from the same lead as A0 to 5V

Sensing: Light

```
photoresistor = new five.Sensor({
   pin: "A0",
   freq: 250
});

board.repl.inject({
   p: photoresistor
});

photoresistor.on("data", function(err, value){
   console.log("light reading is " + value);
});
```

Constrain and map

```
photoresistor.on("data", function(err,
  value) {
   var brightnessValue =
       five.Fn.constrain(
         five.Fn.map(value, 0, 900, 0, 255),
         0,
         255);
   myLed.brightness(brightnessValue);
});
```

Conditional Behaviour

```
if (x==0) {
 // do something
} else {
 // do something else
}
```

- Use comparison operators like == != < <= > >= and logical operators and (&&) or (||) and not (!)
- The conditional operator provides an inline shorthand e.g.

  ```
  var myString = "I have " + (x == 1 ? x + "thing" : x + "things");
  ```

Repeating behaviour (loops)

```
var myArray = [1,2,3];
for (var i = 0; i < myArray.length; i++) {
    // do something specified num of times
    console.log(myArray[i]);
}
while (x < 10) {
    // do something while condition is true
    console.log(x++);
}
board.loop(200, function(){
  // do something every 200 ms
});
```

Manually writing to pins

```
var five = require("johnny-five");
five.Board().on("ready", function() {
  var val = 0;
  var piezoPin = 3;
  // Set pin 9 to PWM mode
  this.pinMode( piezoPin, 3 );
  // beep continously
  this.loop(200, function(){
      if (val){
          this.analogWrite( piezoPin, 20 );
      } else {
          this.analogWrite(piezoPin, 0);
      }
      val = val ? 0 : 1;
  });
});
```

Where to find more code examples

- Johnny-Five docs and wiki
  - https://github.com/rwaldron/johnny-five/wiki

- Arduino Experimenters Guide for NodeJS
  - http://node-ardx.org

How to setup the software at home

- Install Arduino IDE
  - Optional, only required if you want to load Firmata again or experiment with programming the Arduino using C++
- Install NodeJS
  - Visit http://nodejs.org/ and click INSTALL
- Create a folder for your code
- Open up a terminal and install johnny-five from that folder e.g.

```
cd ~/Desktop/code
npm install johnny-five
```

- Install a code editor e.g. Atom (Mac only), SublimeText etc if you don't already have one