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

SENSING

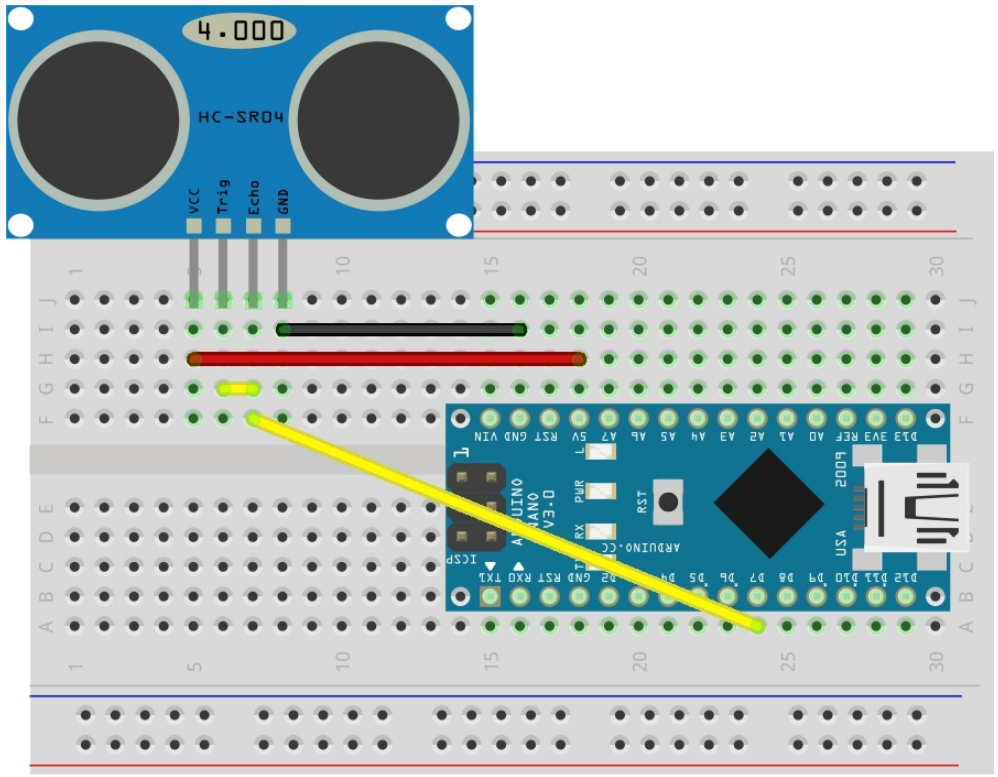
Update Firmata

- To support the Ultrasonic (ping) sensor we will need to update the version of Firmata that we are using on our board
- Download the simplebot code from https://github.com/nodebotsau/simplebot/archive/master.zip
- Open SimpleBotFirmata in Arduino IDE and upload to the Arduino

Loading Firmata onto the Arduino

- Connect the microcontroller board via USB
- 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

Connecting the Ultrasonic sensor



- Connect Vcc to 5V
- Connect GND to ground
- Connect both Trig and Echo to pin 7

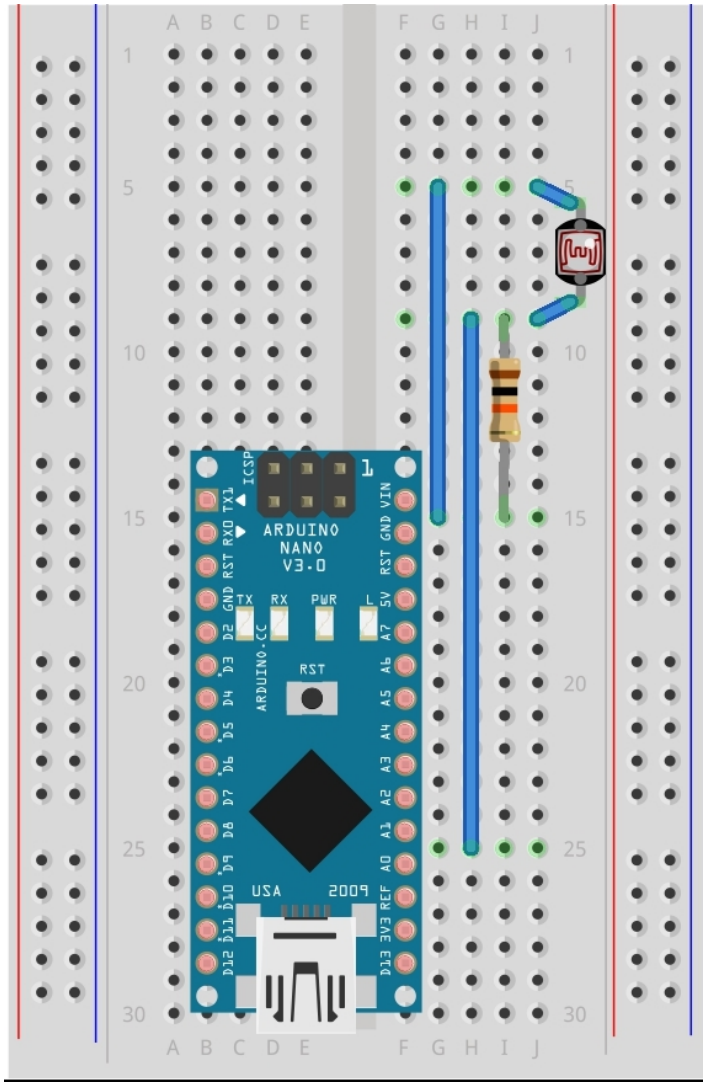
Reading from the Ultrasonic sensor

```
var five = require("johnny-five"),
        board = new five.Board();
  board.on("ready", function() {
    var ping = new five.Ping(7);
    ping.on("change", function() {
      console.log('Detected object at '
        + this.cm + ' cm away');
  });
});
```

Ping (ultrasonic) sensor properties

- this.microseconds: Roundtrip distance in microseconds
- this.inches: Calculated distance to object in inches
- this.cm : Calculated distance to object in centimeters

Connecting 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

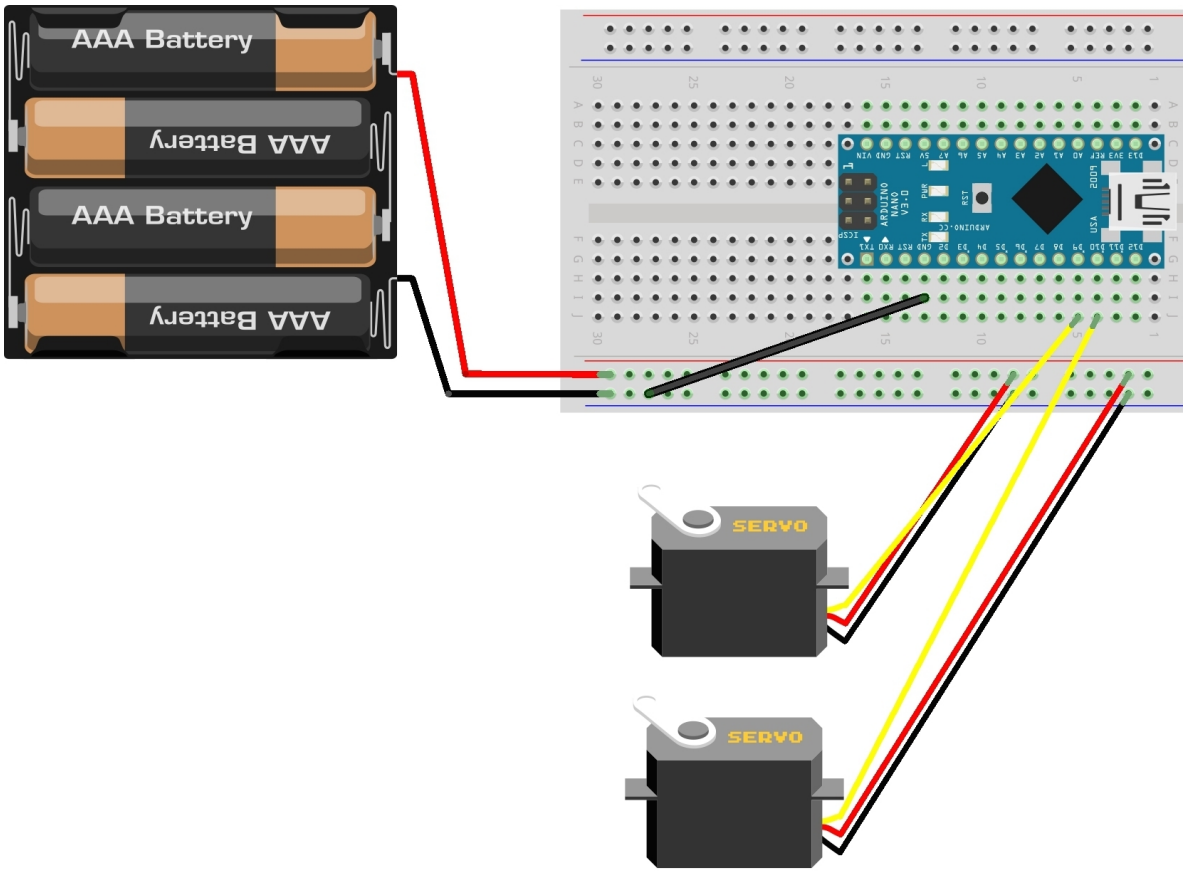
```
photoresistor = new five.Sensor({
   pin: "A0",
   freq: 250
});

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

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

LOCOMOTION

Connecting the CR servos



- Connect the continuous rotation servos that will drive the wheels:
  - Connect the signal (orange) wires to pins 9 and 10
  - Connect the brown wires to ground
  - Connect the red wires to 5V
  - Optionally use batteries to supply power

Creating CR Servo objects

```
var five = require("johnny-five"),
board, leftServo, rightServo;
board = new five.Board();
board.on("ready", function() {
  leftServo = new five.Servo({pin:9,
    type: 'continuous' });
  rightServo = new five.Servo({pin:10,
    type: 'continuous' });
  board.repl.inject({
    l: leftServo,
    r: rightServo
  });
});
```

Controlling a CR servo

- Try the following commands via the REPL:
  - `l.to(90) // stop left wheel`
  - `l.cw() // move clockwise`
  - `l.cw(0.5) // move clockwise, half speed`
  - `l.ccw() // move counterclockwise`

To move bot forwards: l.ccw(); r.cw()

Backwards: l.cw(); r.ccw()

Turn left: l.cw(); r.cw()

Turn right: l.ccw(); r.ccw()

This may be the opposite depending on how you have attached your servos!

Co-ordinating the wheel servos

- An easier way to control the wheel servos is to invert one servo in the code and to group them into one object:

```
wheels = {};
wheels.left = new five.Servo({
  pin: 9,
  type: "continuous"
});

wheels.right = new five.Servo({
  pin: 10,
  type: "continuous",
  isInverted: true
});
```

Driving

```
wheels.both = new five.Servos().stop();

// Drive forwards
wheels.both.cw();

// Stop driving after 3 seconds
this.wait(3000, function() {
  wheels.both.stop();
});
```

RESPONDING

Conditional behaviour

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

- Use comparison operators like == != < <= > >= and logical operators and (&&) or (||) and not (!)

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
});
```

Delayed behaviour

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

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

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);
```

Move away from obstacles

- Read from ping sensor
- If an object is getting close, move backwards a bit and turn, else move forwards

```
ping.on("change", function() {
   var distance = this.cm;
   if (distance < 5 && !turning) {
     turning = true;
     wheels.both.ccw(); // drive backwards
     board.wait(2000, function(){ // at 2 seconds, turn
       wheels.left.cw();
       wheels.right.ccw();
     })
     board.wait(3000, function() { // at 3 seconds drive forward
       wheels.both.cw();
       turning = false;
     });
   }
```