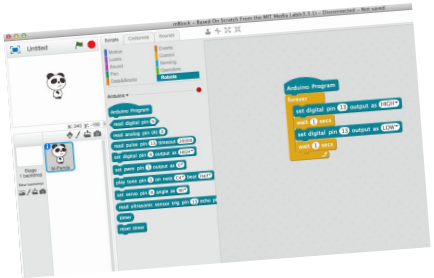
Zombiebots!

Learning electronics, robotics and programming to save us from the zombie apocalypse



v.1.1

Overview

Useful Zombie fact: while slow moving, zombies never seem to tire. Run & frequently make right angle turns to escape.

Chapter One

Unboxing

1.1 What are Zombiebots?

Zombiebots crawled out of the crypt because we believe kiwi kids could do with a hand to become better tinkerers, inventors and innovators. We've devised a fun way for people to learn electronics, robotics and programming while they save us from the zombie menace. The Zombiebot Nano is made out of colour-coded, snap-together parts and programmed with a drag-and-drop interface.

Because if there's a zombie apocalypse, we're going to need robots, right?

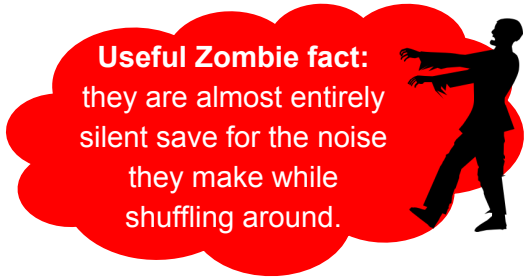
1.2 Jargon Buster

Arduino: a type of microcontroller (tiny computer) that can be programmed to do things. We're using an Arduino Nano

Breadboard: a plastic base that allows you to create circuits without solder. If you'd like to see how they work, click here.

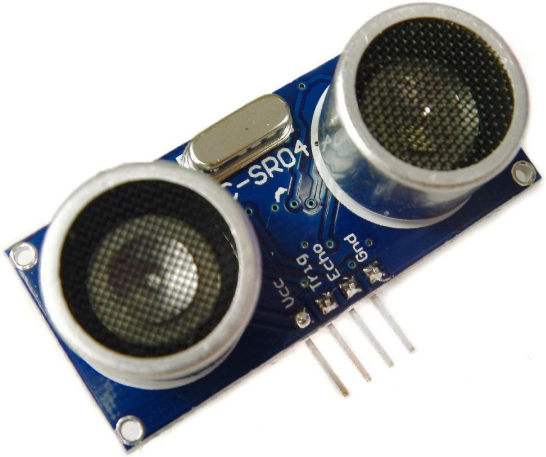
Jumper: a little wire that can be used to make connections between the holes of a breadboard

Microcontroller: a mini computer that you can program to do simple tasks. An Arduino is a microcontroller.

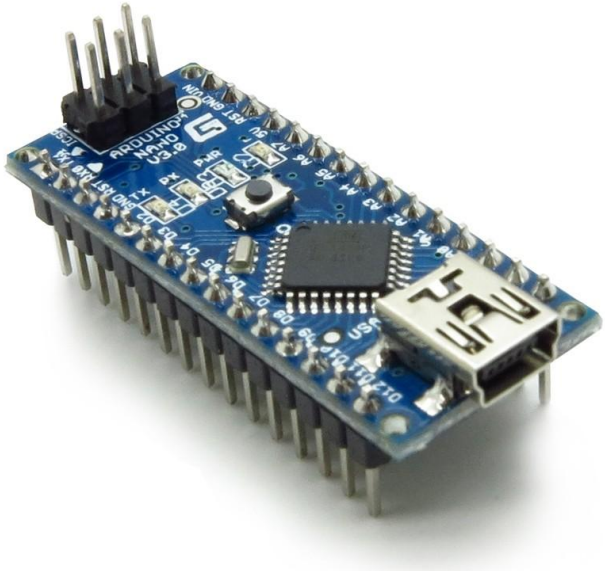


1.3 What's in the box? (Nano)

Ultrasonic sensor. Sends a pulse out and counts how long before the pulse comes back to it. Can be used to sense distance.



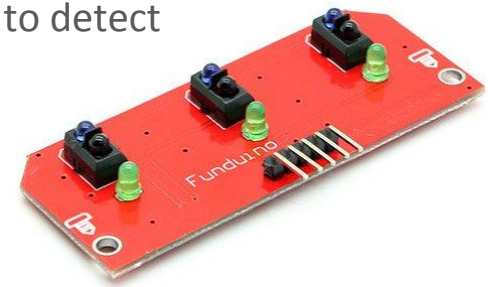
Arduino Nano. An Arduino is a microcontroller (like a mini computer). We're going to programme it to make lights blink, play a tune, and eventually control our robot.



1.4 What's in the box? (Nano plus)

Bluetooth module. This allows us to communicate with our Arduino via Bluetooth. The easiest way to do this is via an Android smartphone that is Bluetooth capable.

Infrared sensor array. This is a set of three infrared sensors that can detect things like colour. We use it to detect the difference between black and white so your robot can follow a line.



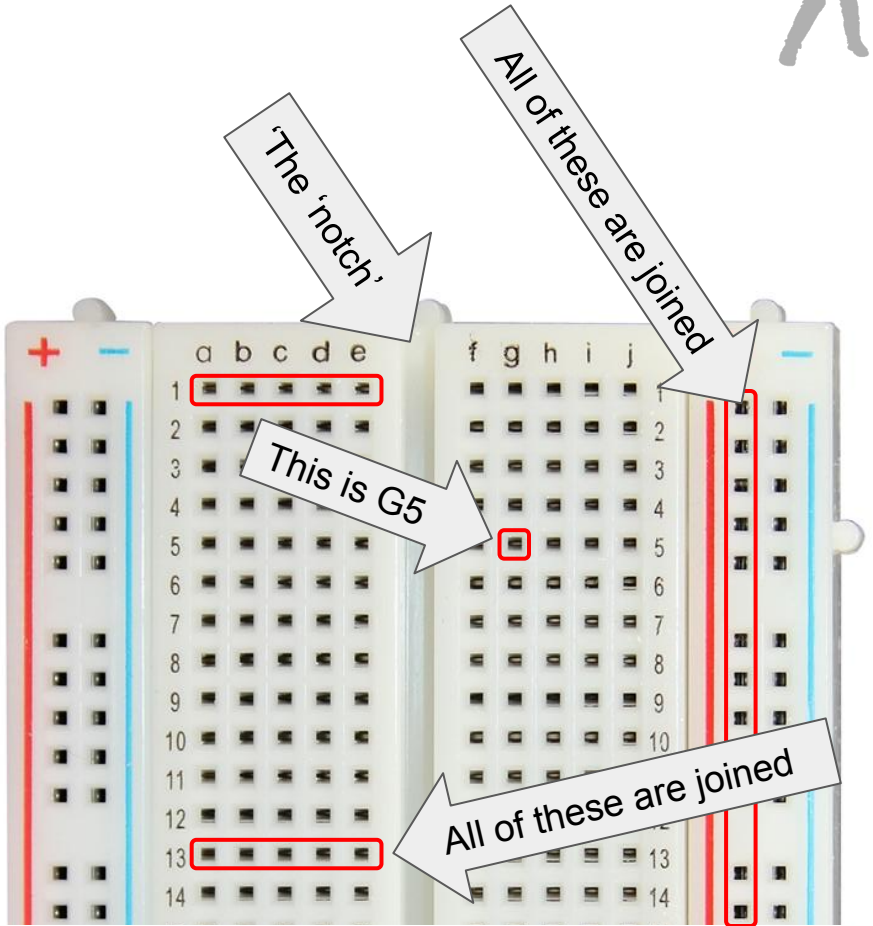
1.5 Breadboard

A breadboard is sometimes called a prototyping board and is an easy way to make circuits without soldering. If you put a wire into 1A, and another into 1B (or 1C, 1D or 1E) they are joined by clips inside the breadboard.

The notch separates the two sides of the board, so A-E and F-J are separate.

The 25 pins in each (+) and (-) strip down the sides are also linked together. These are called power rails.

Got the idea? Take our '[How to use a breadboard]' video quiz



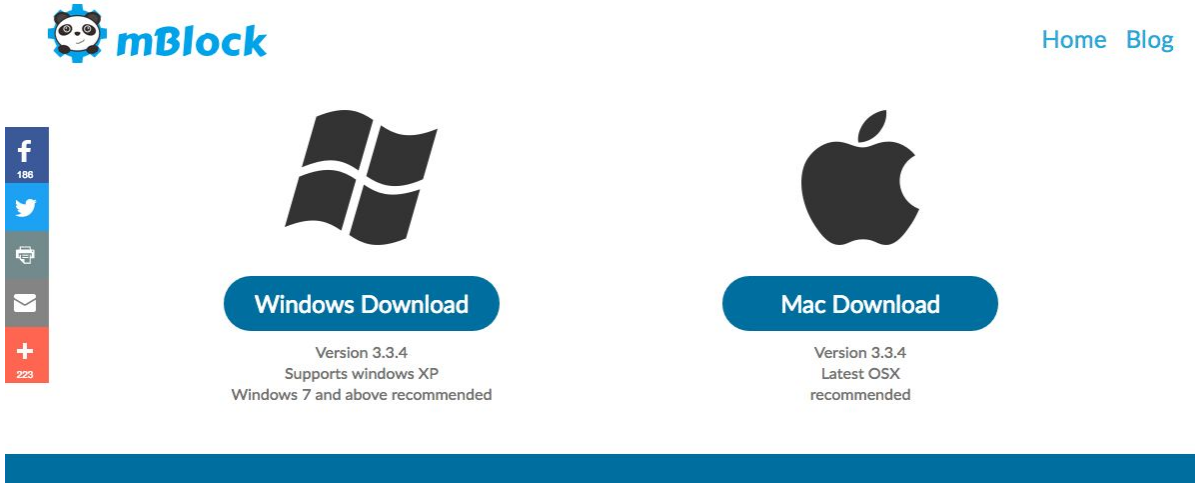
Chapter Two

Getting Set Up

2.1 Get the software

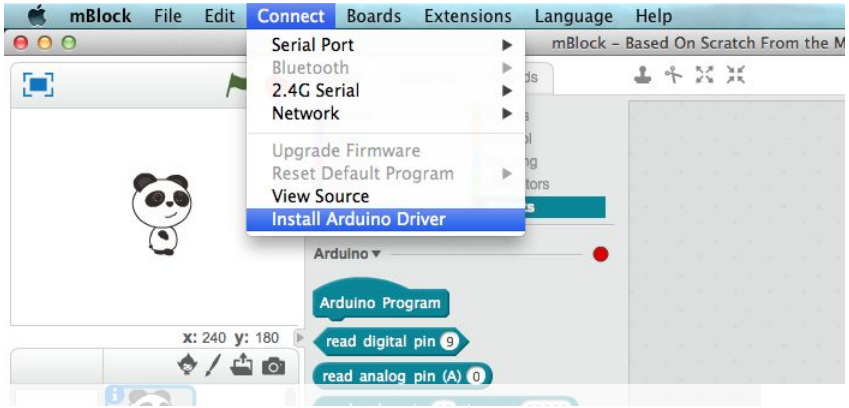
To program our robot, we are going to use software called mBlock.

You can download and install it from here: www.mblock.cc/download

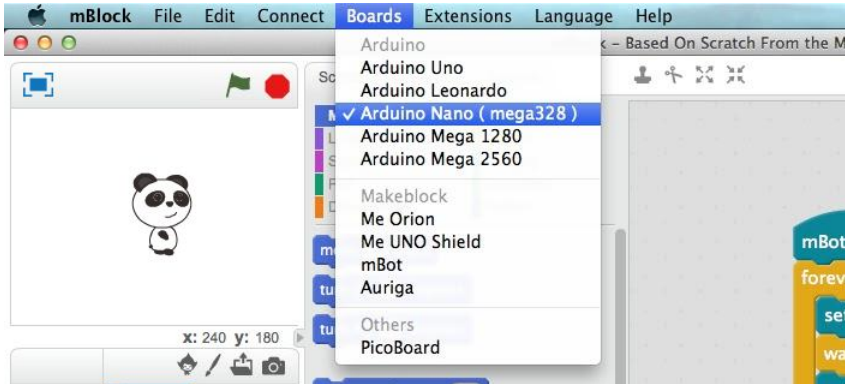


Useful Zombie hint: always keep a pair of running shoes at the front *and* the back doors.

2.2 Set up the software

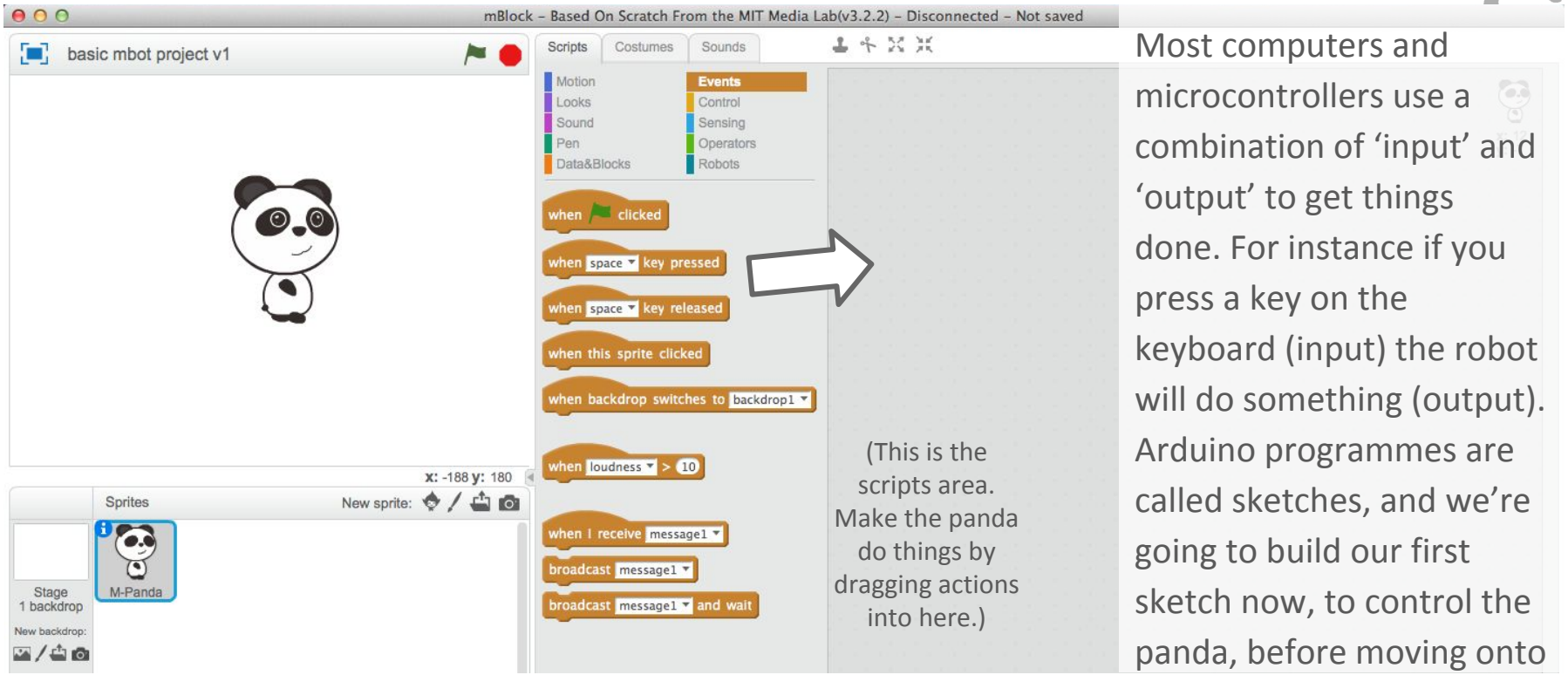


1. From the ‘Connect’ menu, choose ‘Install Arduino Driver’. This ensures your computer will be able to talk to your Arduino/robot.



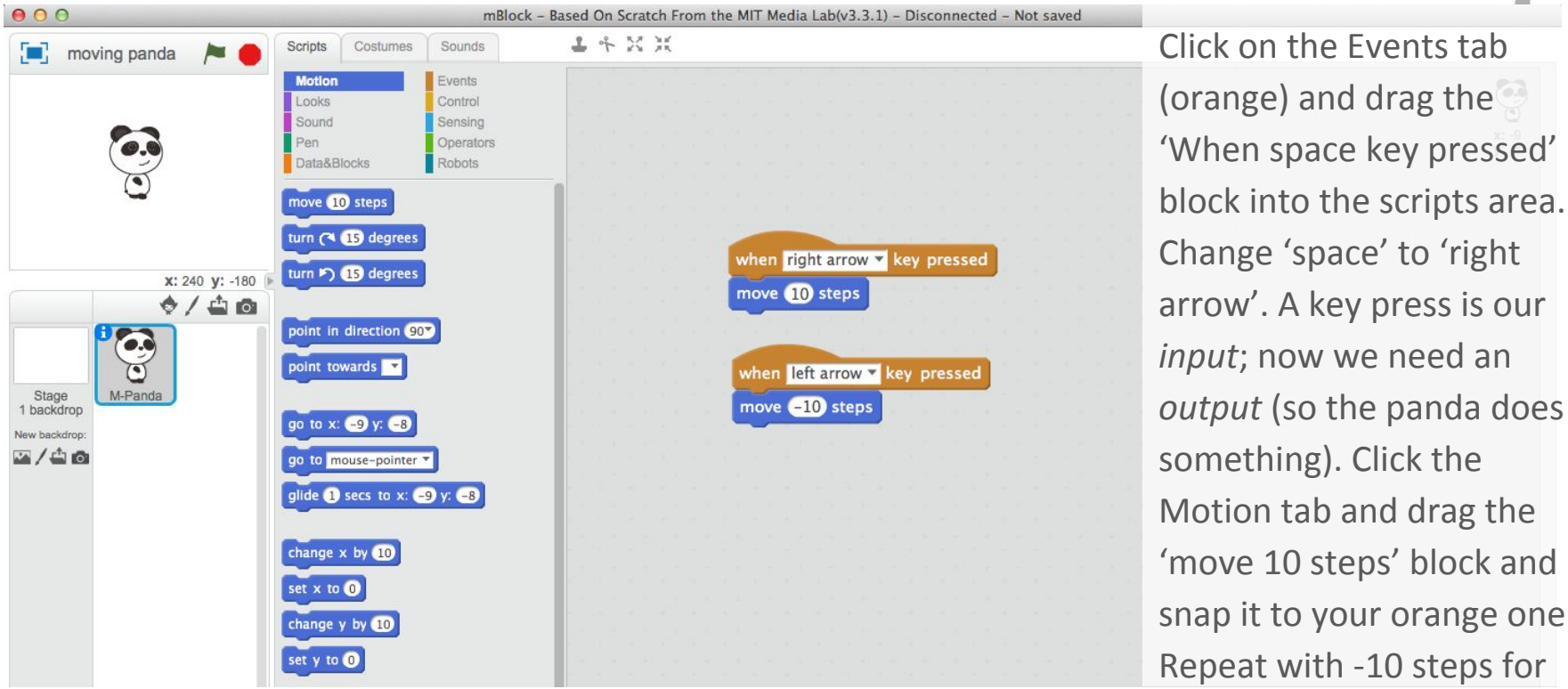
2. From the ‘Boards’ menu, choose Arduino Nano (mega328). This tells your computer what kind of Arduino you’re using.

2.3 Simple scripts



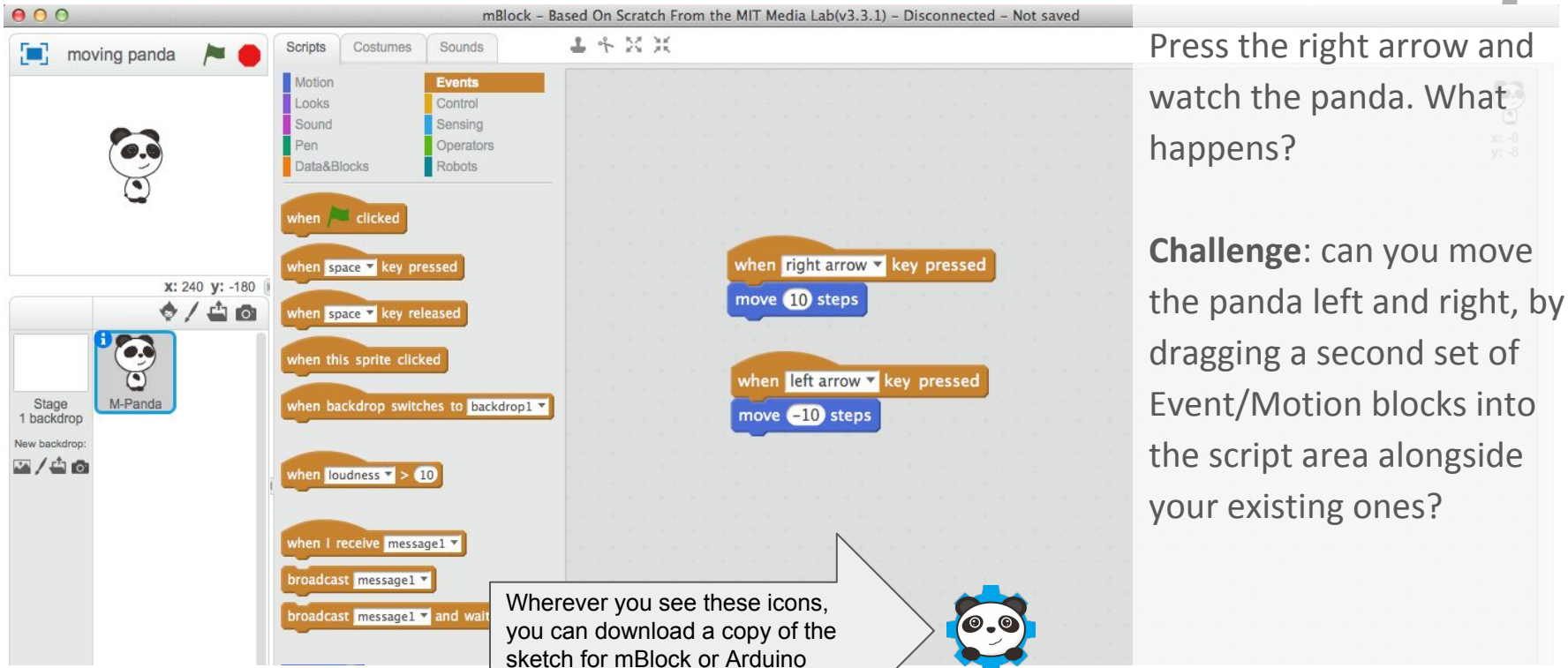
Most computers and microcontrollers use a combination of 'input' and 'output' to get things done. For instance if you press a key on the keyboard (input) the robot will do something (output). Arduino programmes are called sketches, and we're going to build our first sketch now, to control the panda, before moving onto the robot.

2.4 Make the Panda move



Click on the Events tab (orange) and drag the 'When space key pressed' block into the scripts area. Change 'space' to 'right arrow'. A key press is our *input*; now we need an *output* (so the panda does something). Click the Motion tab and drag the 'move 10 steps' block and snap it to your orange one. Repeat with -10 steps for left arrow.

2.4 Make the Panda move

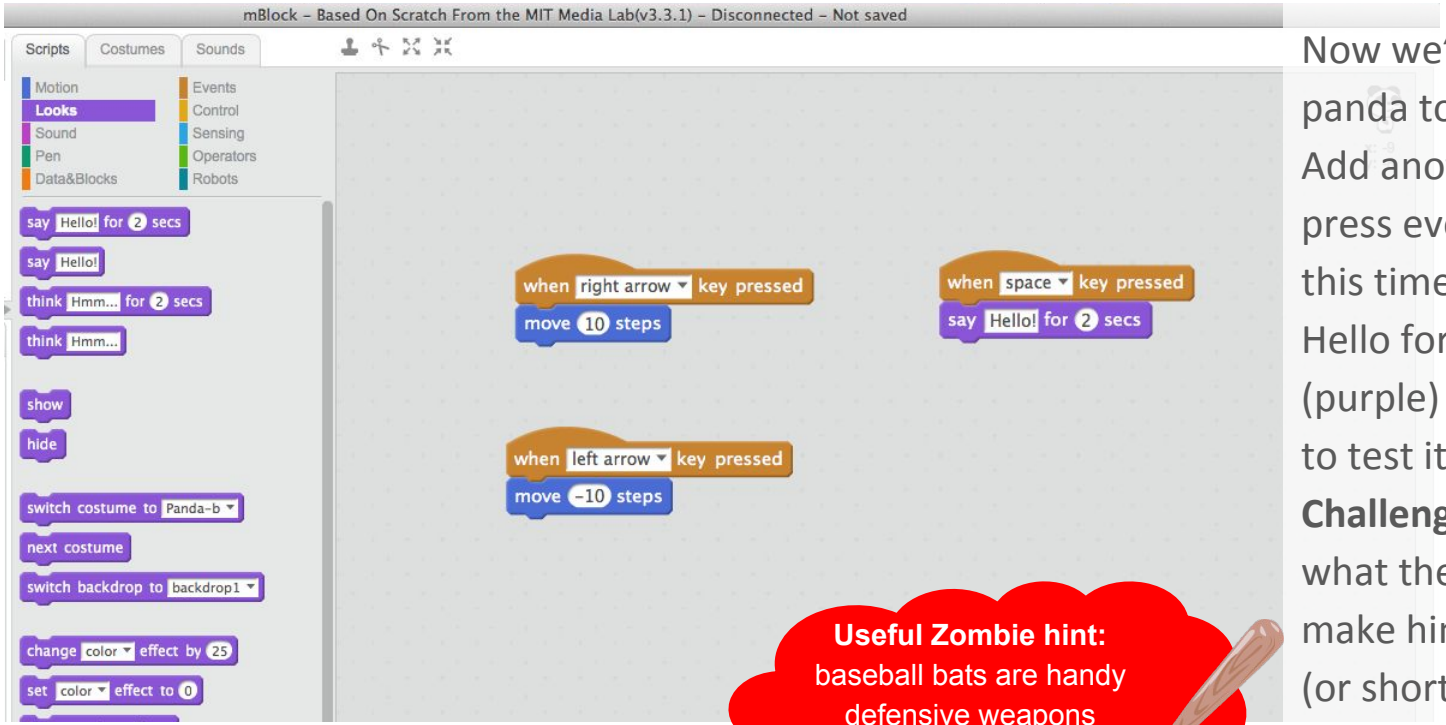


Wherever you see these icons, you can download a copy of the sketch for mBlock or Arduino

Press the right arrow and watch the panda. What happens?

Challenge: can you move the panda left and right, by dragging a second set of Event/Motion blocks into the script area alongside your existing ones?

2.5 Make the Panda Say hello!

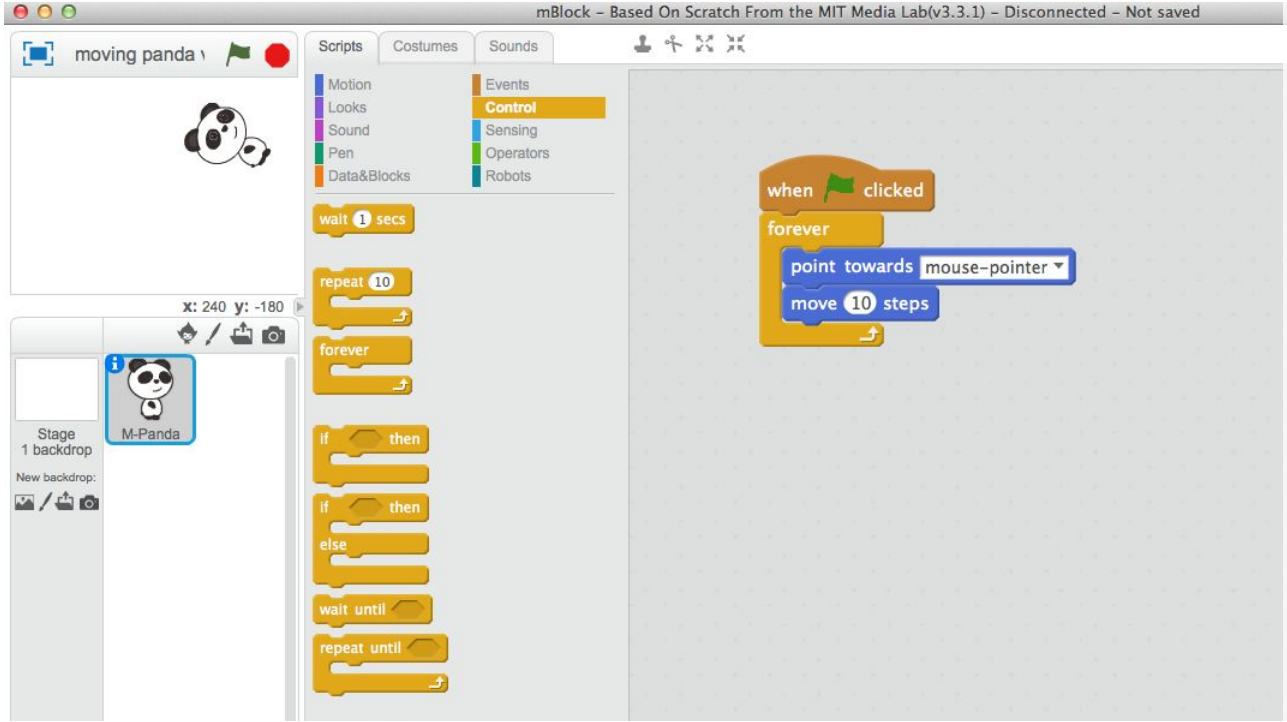


Now we're going to get our panda to do a bit more. Add another space key press event (orange) and this time snap the the 'Say Hello for 2 secs' block (purple) to it. Hit space key to test it out.

Challenge: can you change what the panda says, and make him say it for longer (or shorter)? (Play around with the options.)

Useful Zombie hint: baseball bats are handy defensive weapons because they don't require ammunition.

2.6 Make the panda follow your mouse

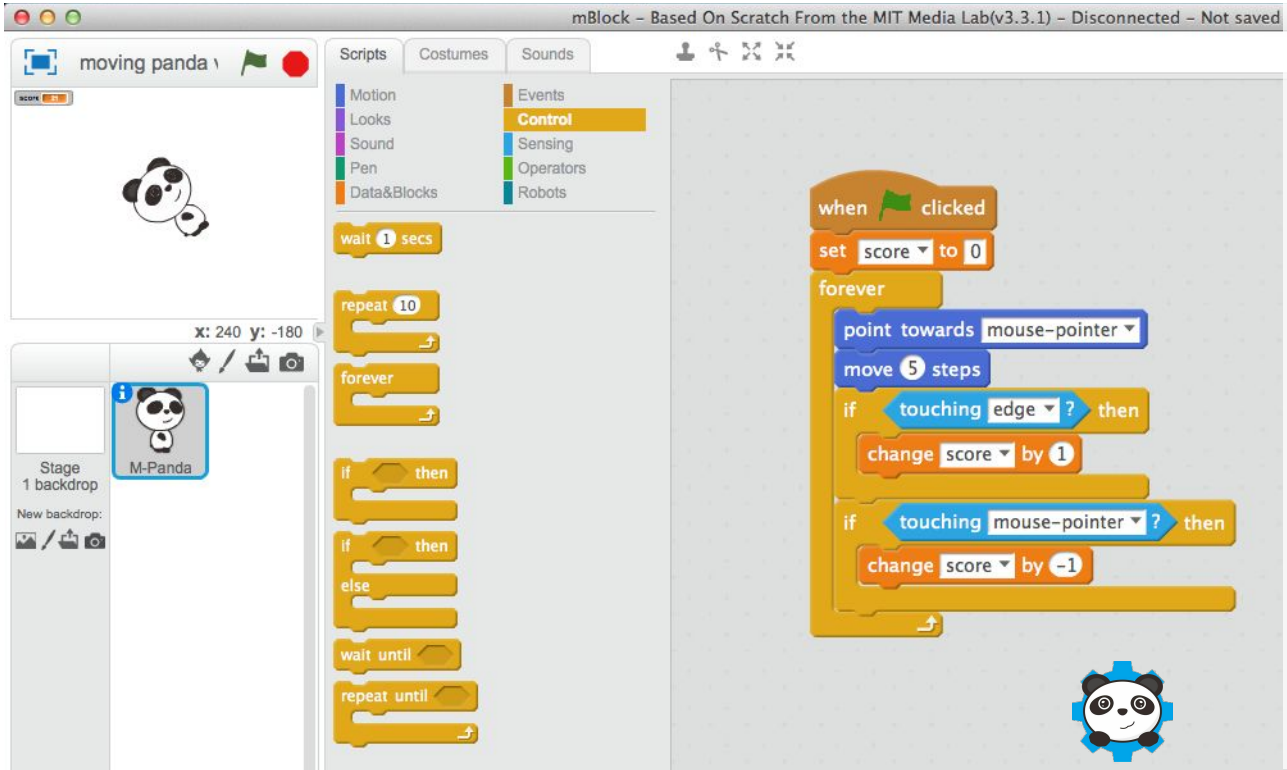


Let's get a bit tricky now. Instead of using a key press for input, let's use the mouse.

Find the blocks on the left and assemble them in the correct order. (Choose 'mouse pointer' from the drop down menu in the 'point towards' block).

Play with the number of steps to make the panda faster or slower.

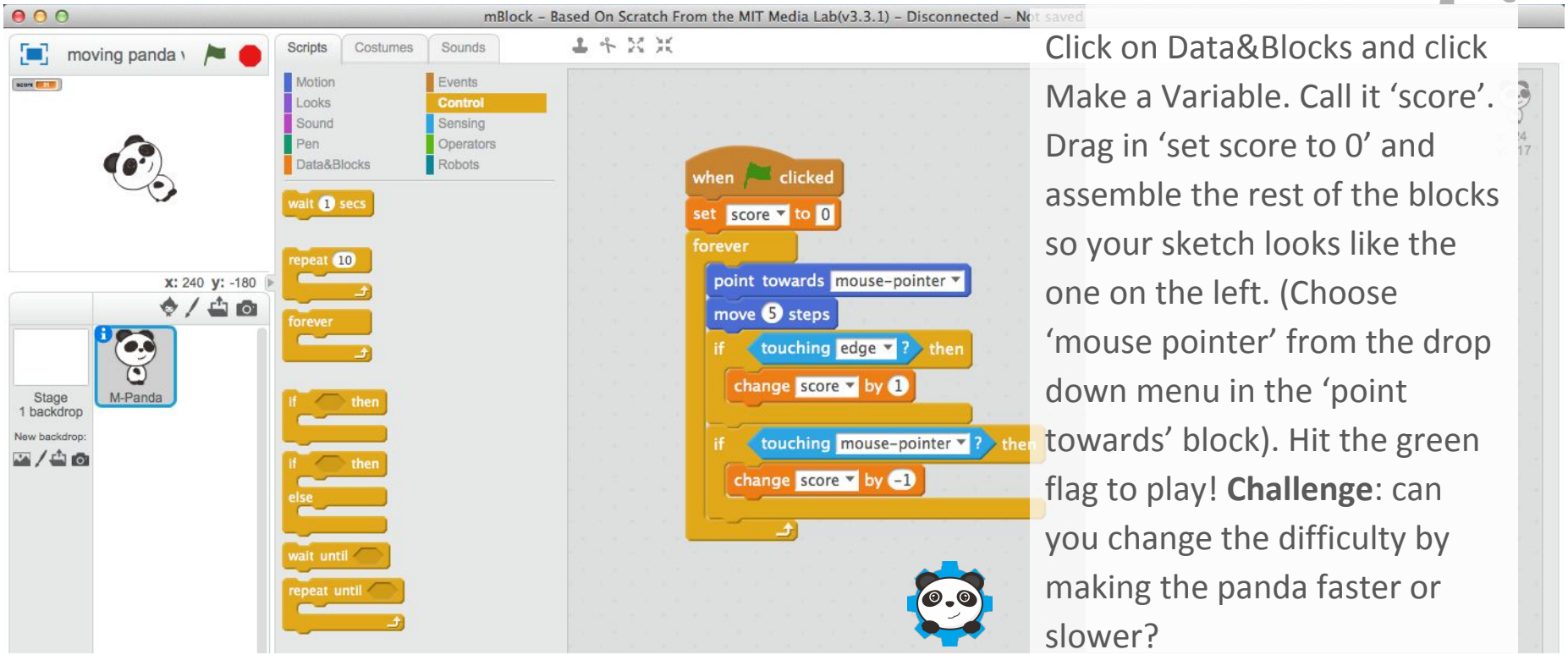
27 Keeping score with variables



Variables are containers that can hold store numbers that change (like a score or a countdown).

We're going to make a game that uses a variable to keep score. Using our 'panda chasing the mouse' sketch, we're going to set it up so that if the panda touches the edge of the box we gain a point, and if the panda touches the mouse pointer we lose a point.

2.7 Keeping score with variables

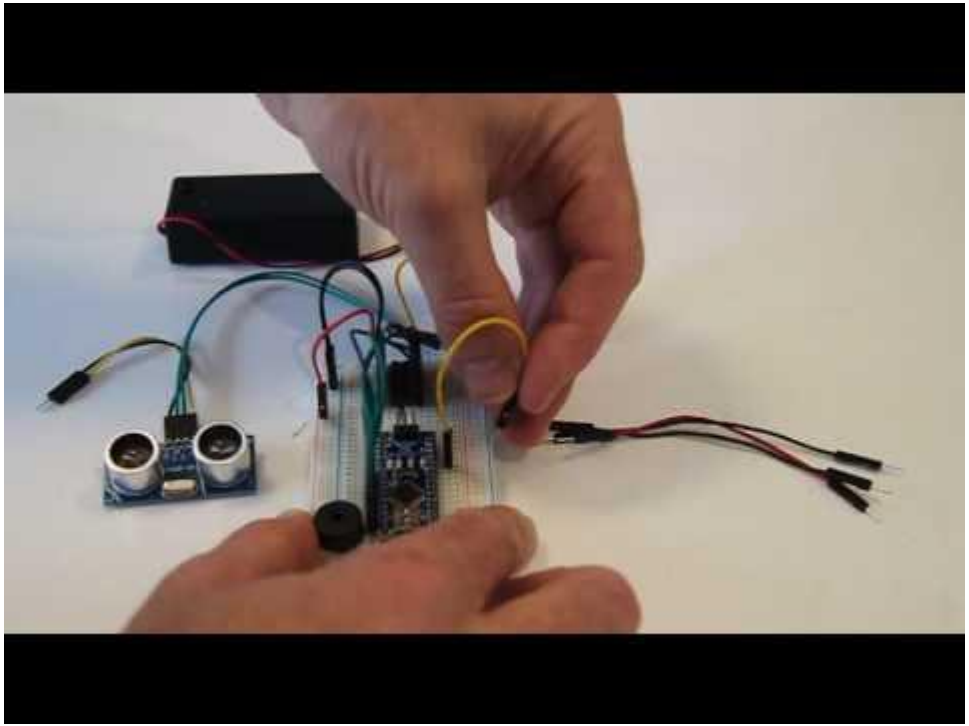


Click on Data&Blocks and click Make a Variable. Call it 'score'. Drag in 'set score to 0' and assemble the rest of the blocks so your sketch looks like the one on the left. (Choose 'mouse pointer' from the drop down menu in the 'point towards' block). Hit the green flag to play! **Challenge**: can you change the difficulty by making the panda faster or slower?

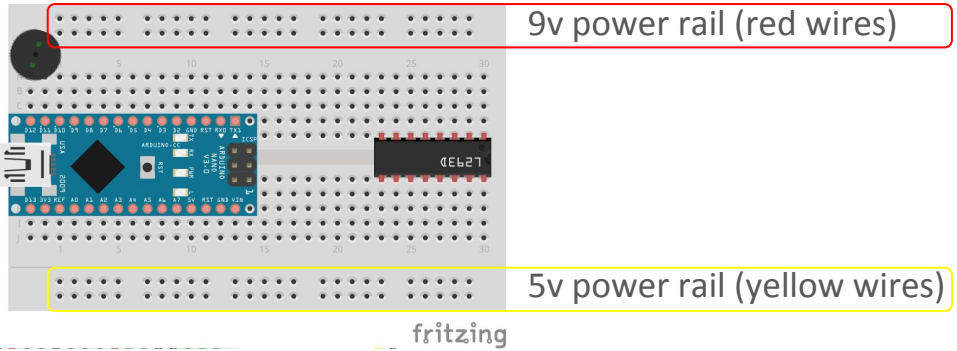
Chapter Three

Assemble your robot

Assembly video: Step one



3.1 Start with the breadboard

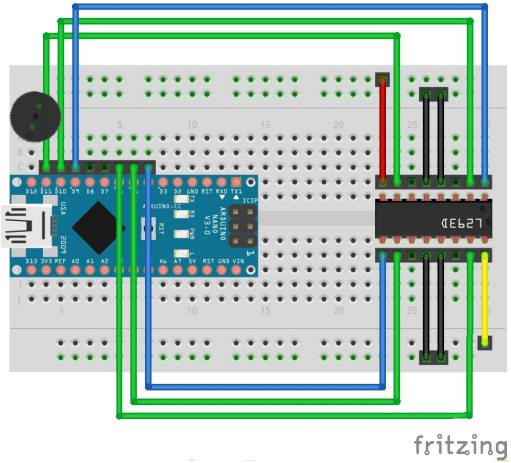


Start with your breadboard and Arduino. You'll notice two other items on the breadboard:

1. the round piezo buzzer (we'll use this to make a bit of noise), and
2. the chip that controls the motors (we'll send signals to the chip to get it to move the motors forward or back.

Useful Zombie hint: have an emergency kit with enough food, water & supplies to last at least two days until you reach a safe zone.

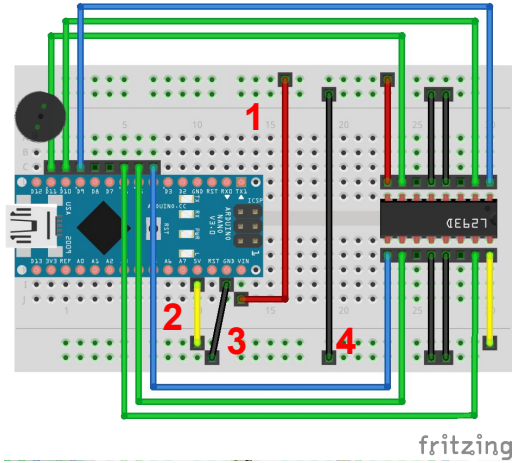
3.2 Add wire assembly



Add the wiring assembly, matching up the colours correctly.
Follow the diagram to the left to ensure the three blocks go in holes:

- C22-C29 (blue and green, with the blue wires in C22 and C27)
- D1-D8 (red wire in D8)
- G1-G8 (yellow wire in G1)

3.3 Add jumper wires



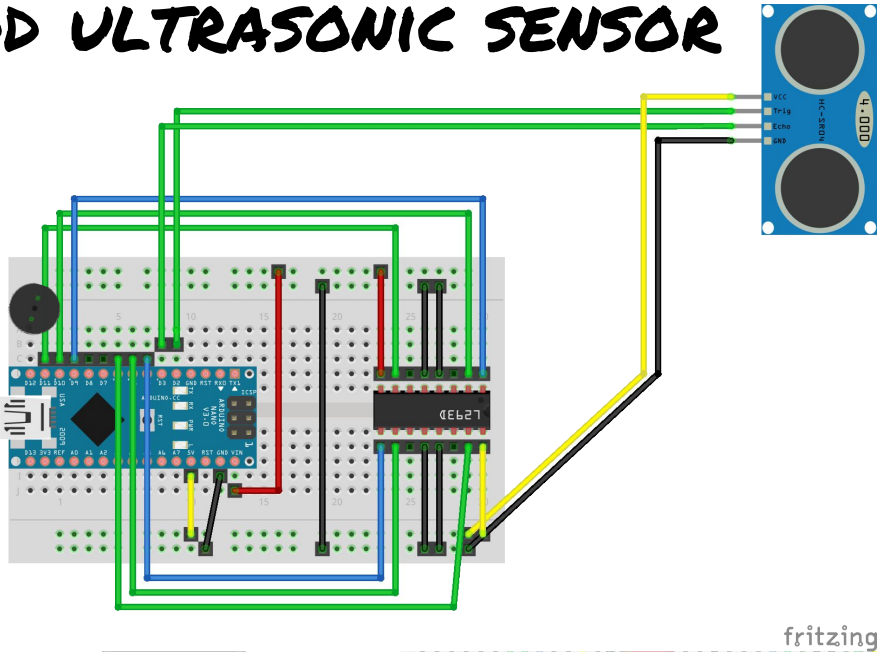
9v power rail (red wires)

5v power rail (yellow wires)

Now add these jumper wires:

1. Red from the red (+) side of the 9v power rail to the VIN pin (J16)
2. Yellow from the 5v pin (J19) to the red (+) side of the 5v power rail.
3. Black from J17 to the blue (-) side of the 5v power rail.
4. Black from the blue (-) side of the 9v power rail to the blue (-) side of the 5v power rail.

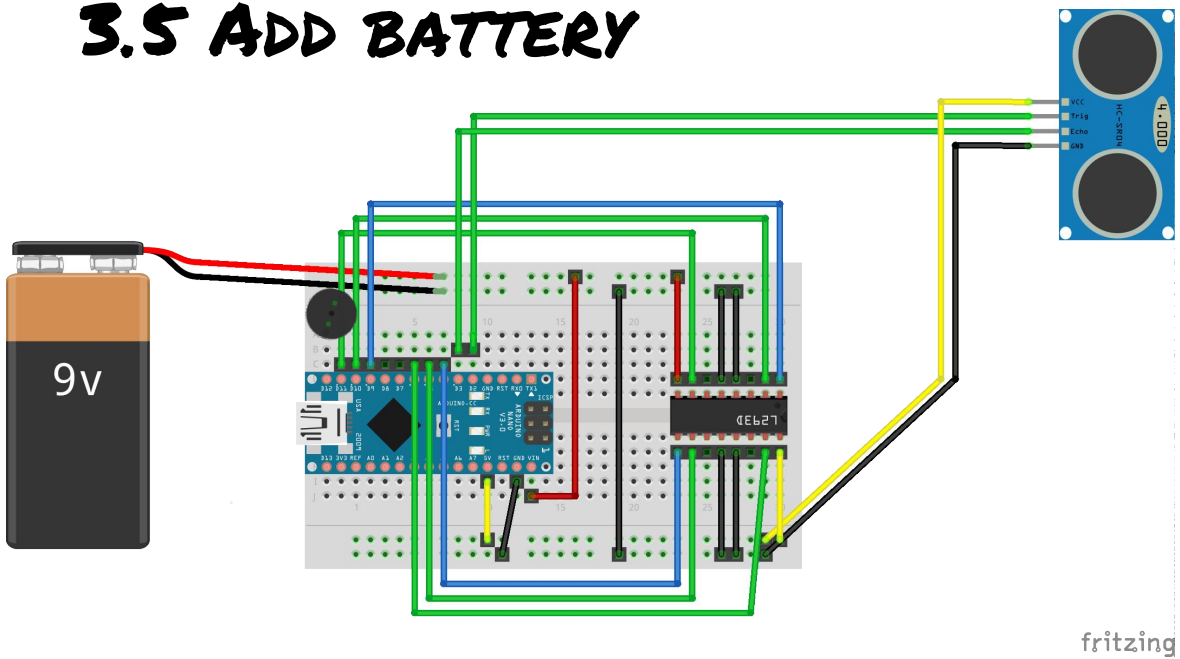
3.4 Add ultrasonic sensor



Connect the ultrasonic sensor:

1. Plug the yellow (VCC) & black (GND) wires from the Ultrasonic sensor into the 5v power rail, matching the colours (yellow + , black -).
2. Connect the green TRIG jumper to Arduino pin D2 (B20)
3. Connect the green ECHO jumper to Arduino pin D3 (B21)

3.5 Add Battery

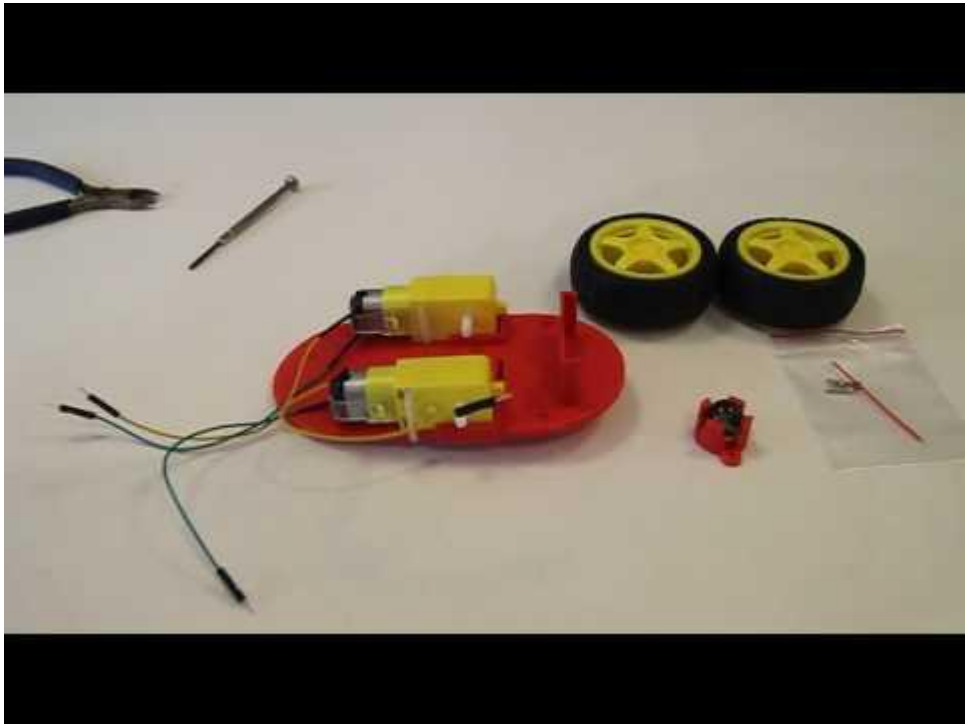


Add your battery by pushing the green screw terminal into the 9v power rail, taking care to match the colours: red and black. The battery will slot under the breadboard between the motors.

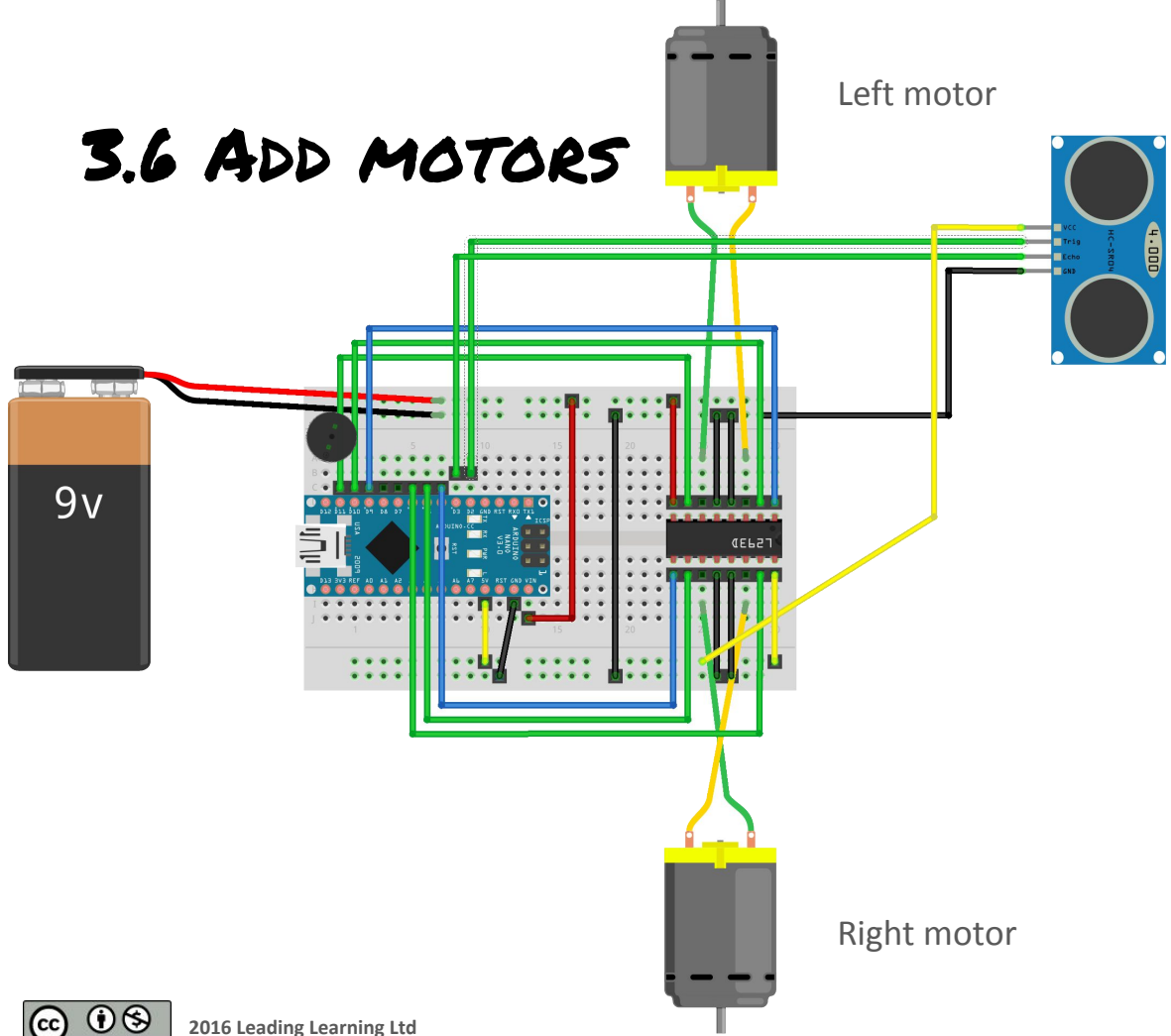
Your battery can be recharged by taking it out of the black holder and plugging it into a micro USB phone charger.

Important: don't recharge it in a traditional 9v battery charger.

Assembly video: Step Two



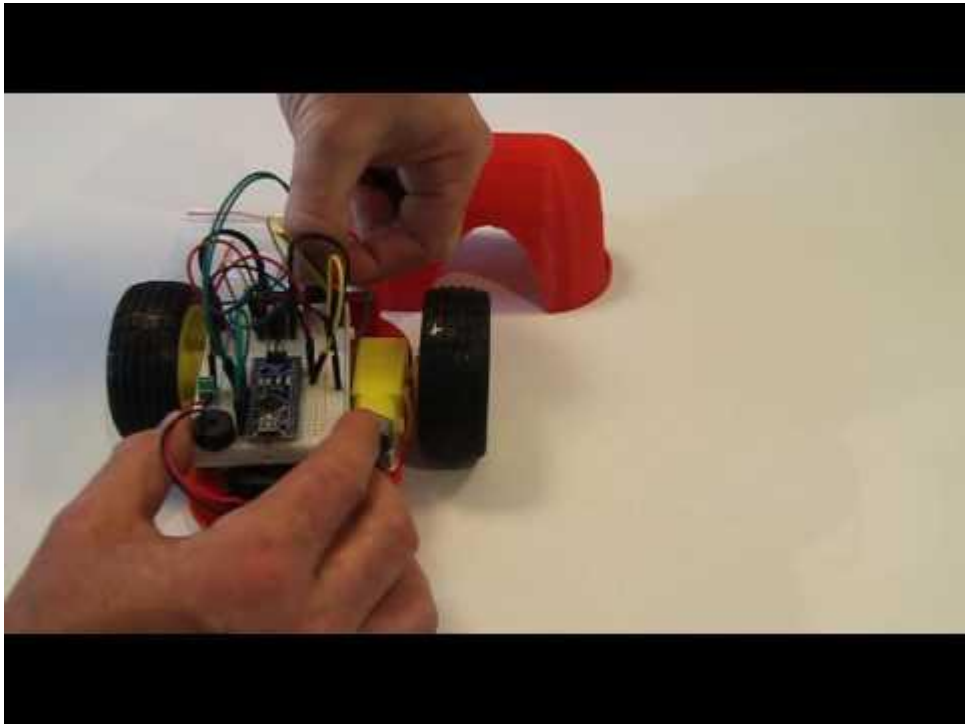
3.6 Add motors



Time to add the motors. Bring the wires forward underneath the breadboard so they are at the front of the base.

1. Connect the two wires from the left motor into breadboard holes C3 (yellow) and C6 (green).
2. Repeat for the right motor using holes H3 (yellow) and H6 (green)

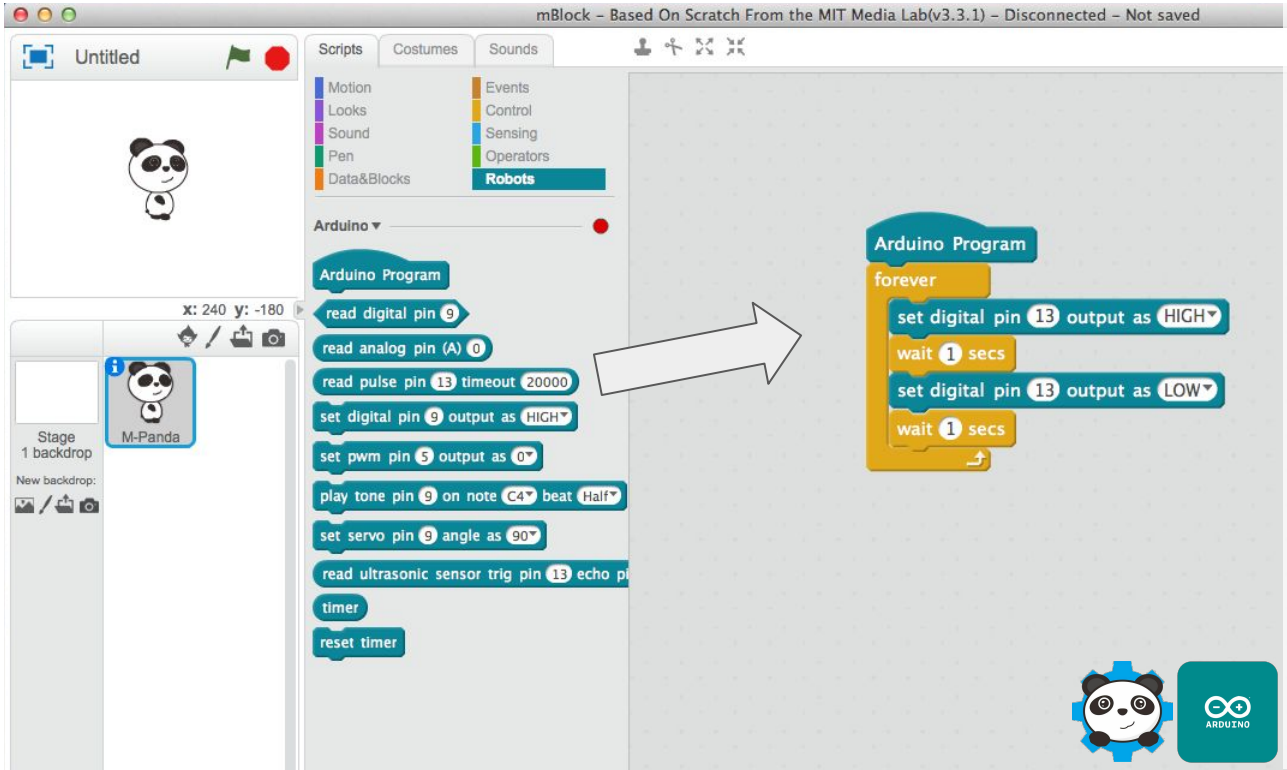
Assembly video: Step Three



Chapter Four

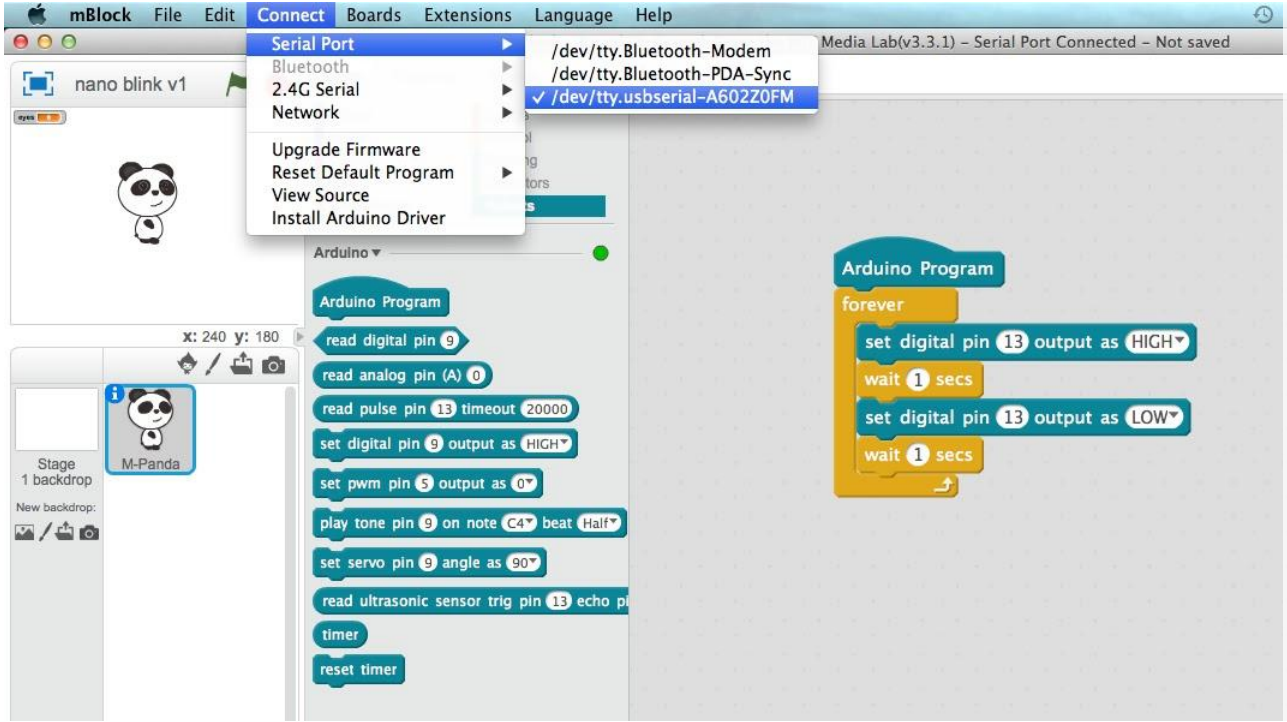
Practice your skills

4.1 Make a light blink



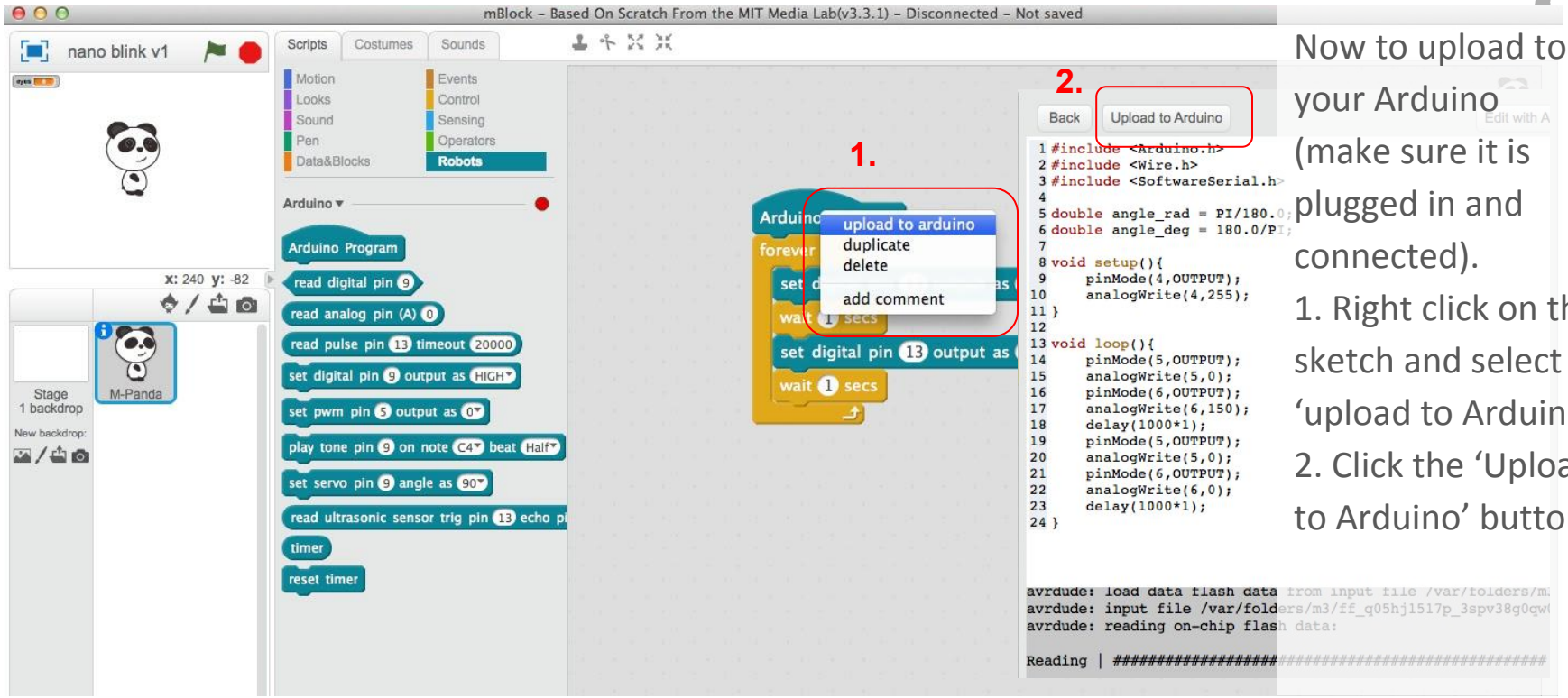
There is an LED light built into your Arduino. This sketch will make it flash on and off a second at a time. From the 'Robots' and 'Control' tabs, drag these actions across and assemble them in this order. You'll need to change the digital pin from 9 to 13 because that's where our LED is.

4.2 Connect your nano



1. From the Connect' Menu, choose Serial Port then dev/tty.usbserial-XXXXXXX (COM on Windows)
2. Click the 'Robots' tab to make sure the red light has turned to green. This indicates you are connected.

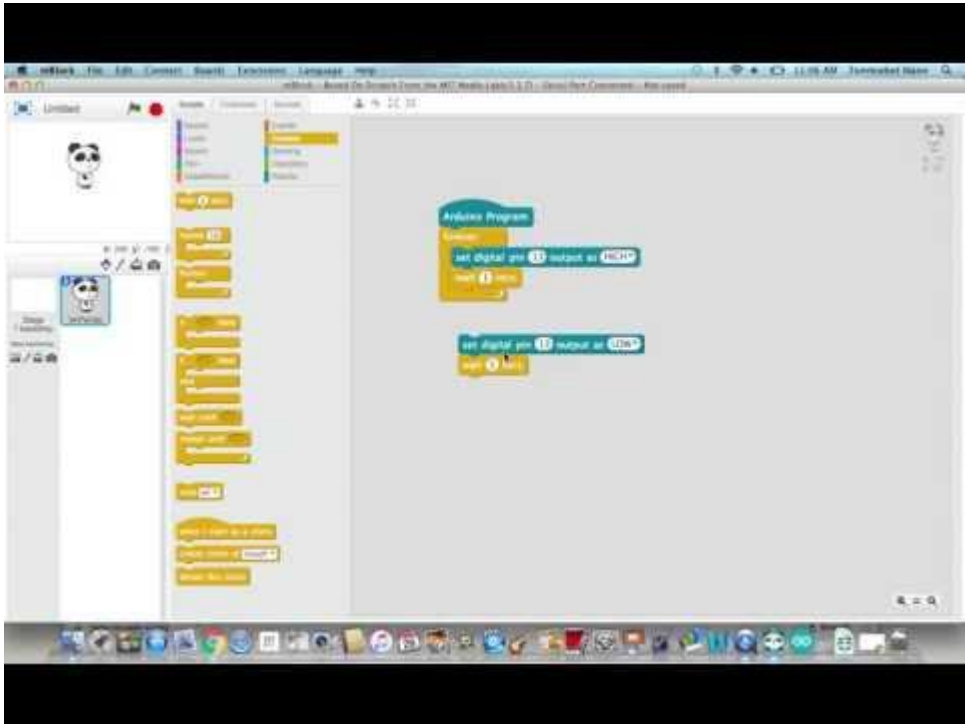
4.3 Upload your sketch



Now to upload to your Arduino (make sure it is plugged in and connected).

1. Right click on the sketch and select ‘upload to Arduino’
2. Click the ‘Upload to Arduino’ button.

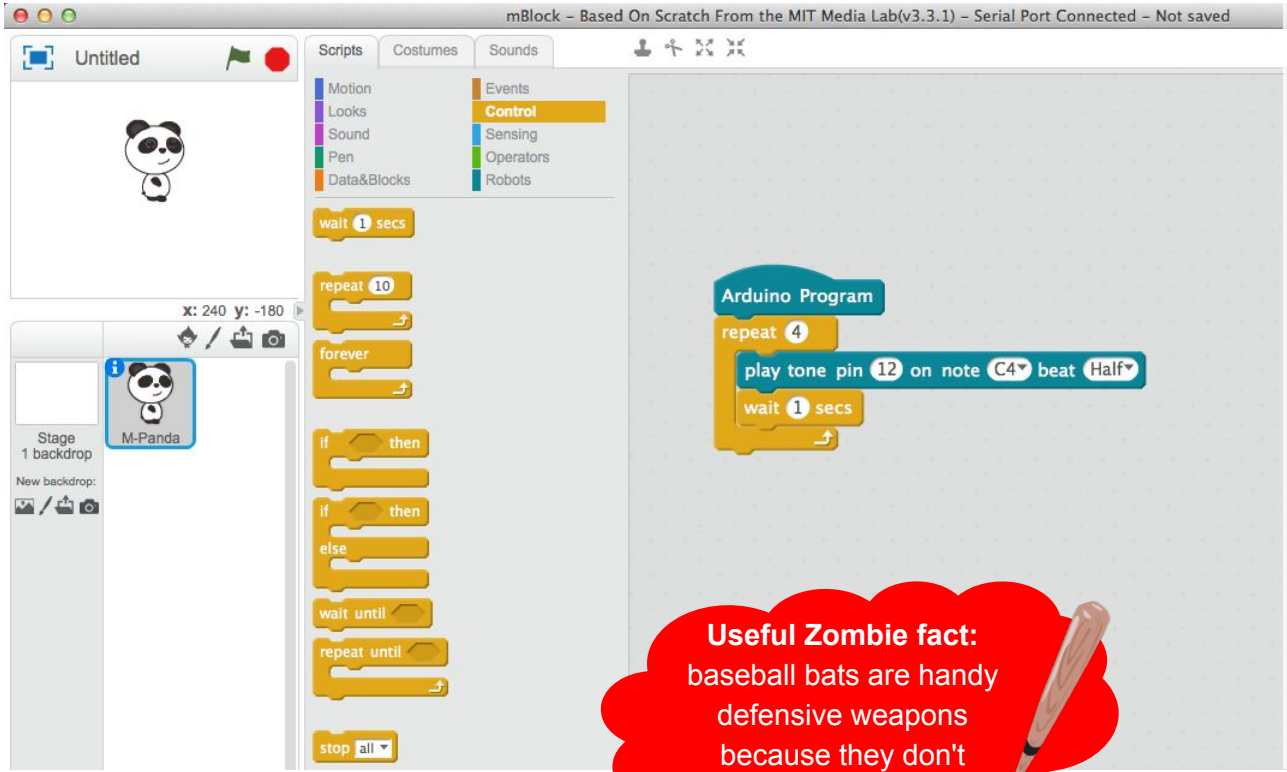
4.3 Upload your sketch



Video tutorial: build a simple sketch to make the LED blink and upload it to your Arduino

Challenge: can you make the LED blink faster or slower? What would you need to change? Re-upload to see if you are right.
Can you make the LED blink quickly three times then wait for two seconds then blink quickly three time again?

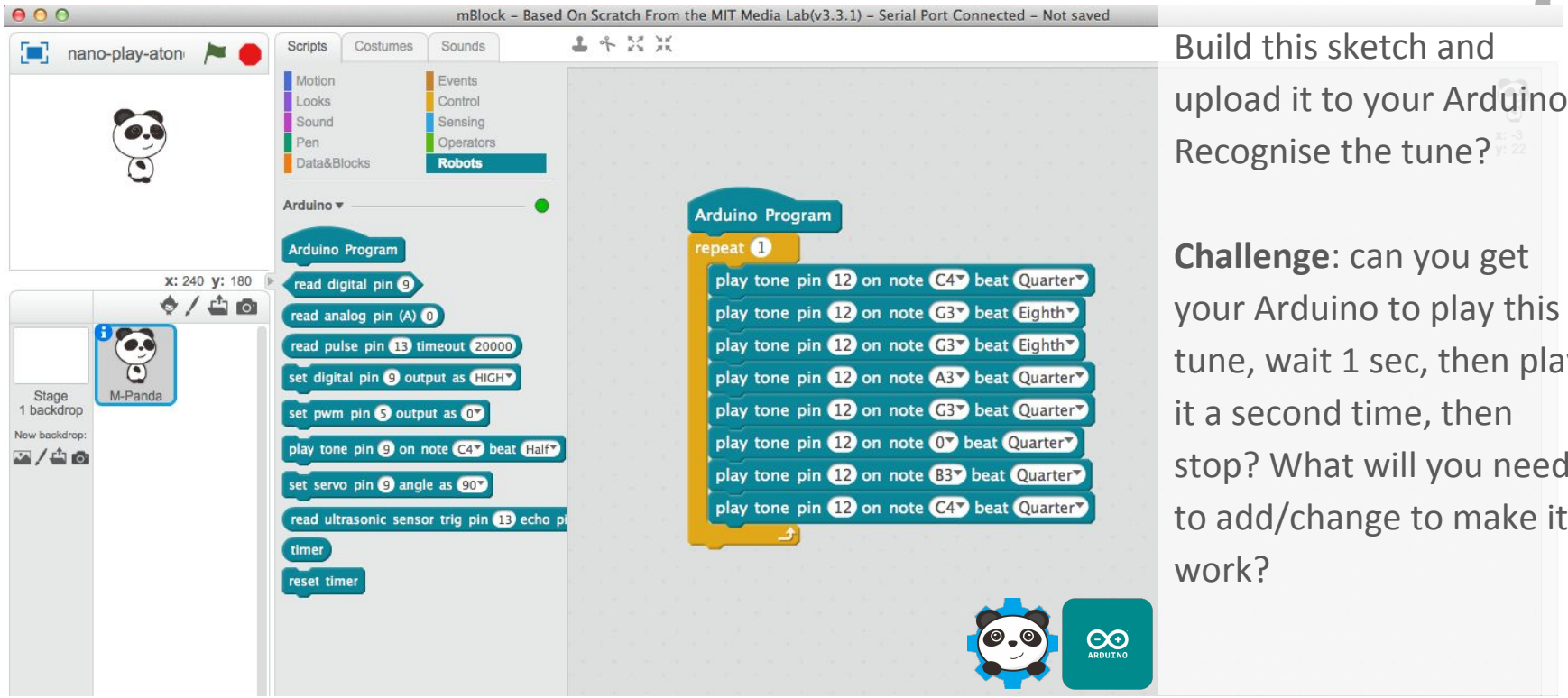
4.4 Make a sound



Useful Zombie fact: baseball bats are handy defensive weapons because they don't require ammunition.

We're going to use the piezo buzzer on our breadboard to make sound now. The software has tones built into the 'Robots' tab. Make a sketch like this one (you'll need to change the pin to 12, because that's where our buzzer is). Connect via USB and upload the sketch. **Challenge**: can you play three different tones in a row?

4.5 Play a tune



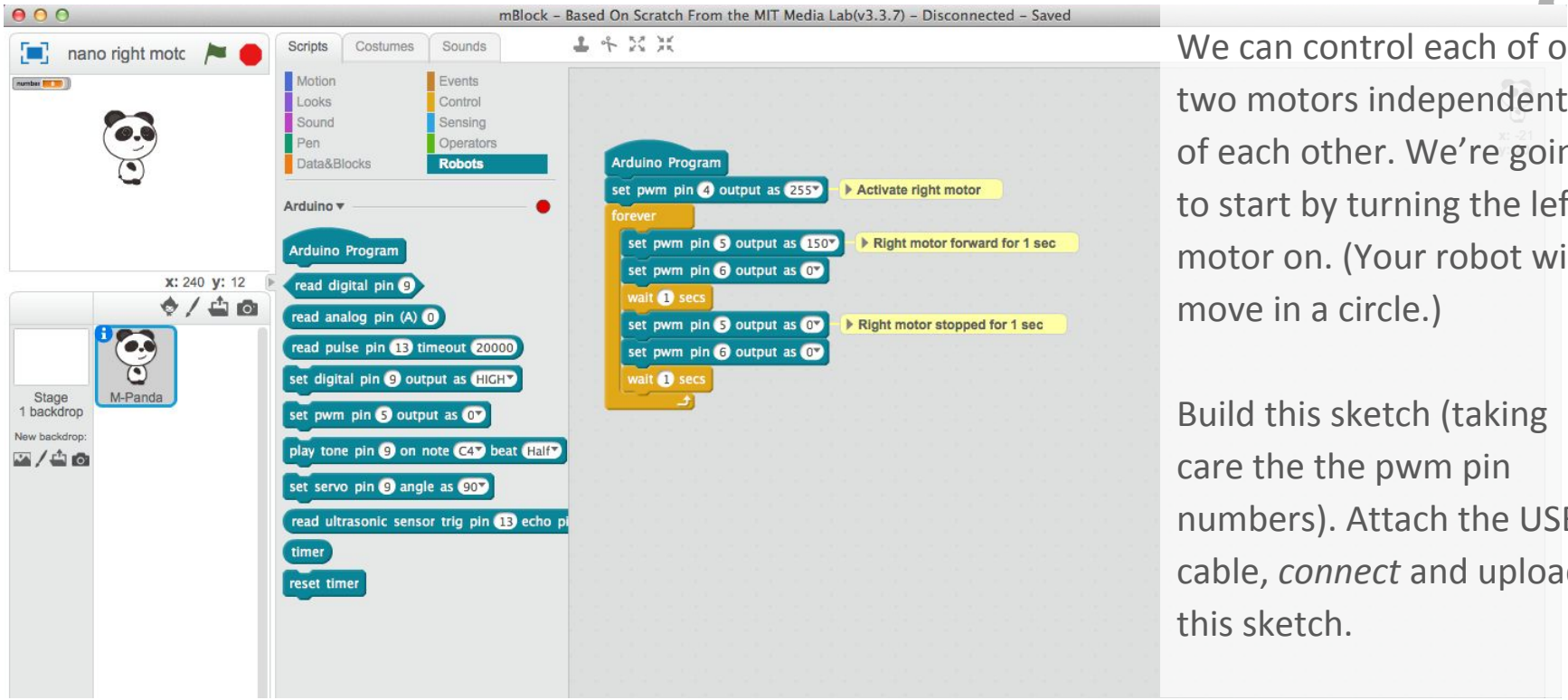
Build this sketch and upload it to your Arduino. Recognise the tune?

Challenge: can you get your Arduino to play this tune, wait 1 sec, then play it a second time, then stop? What will you need to add/change to make it work?

Chapter Five

Let's get moving!

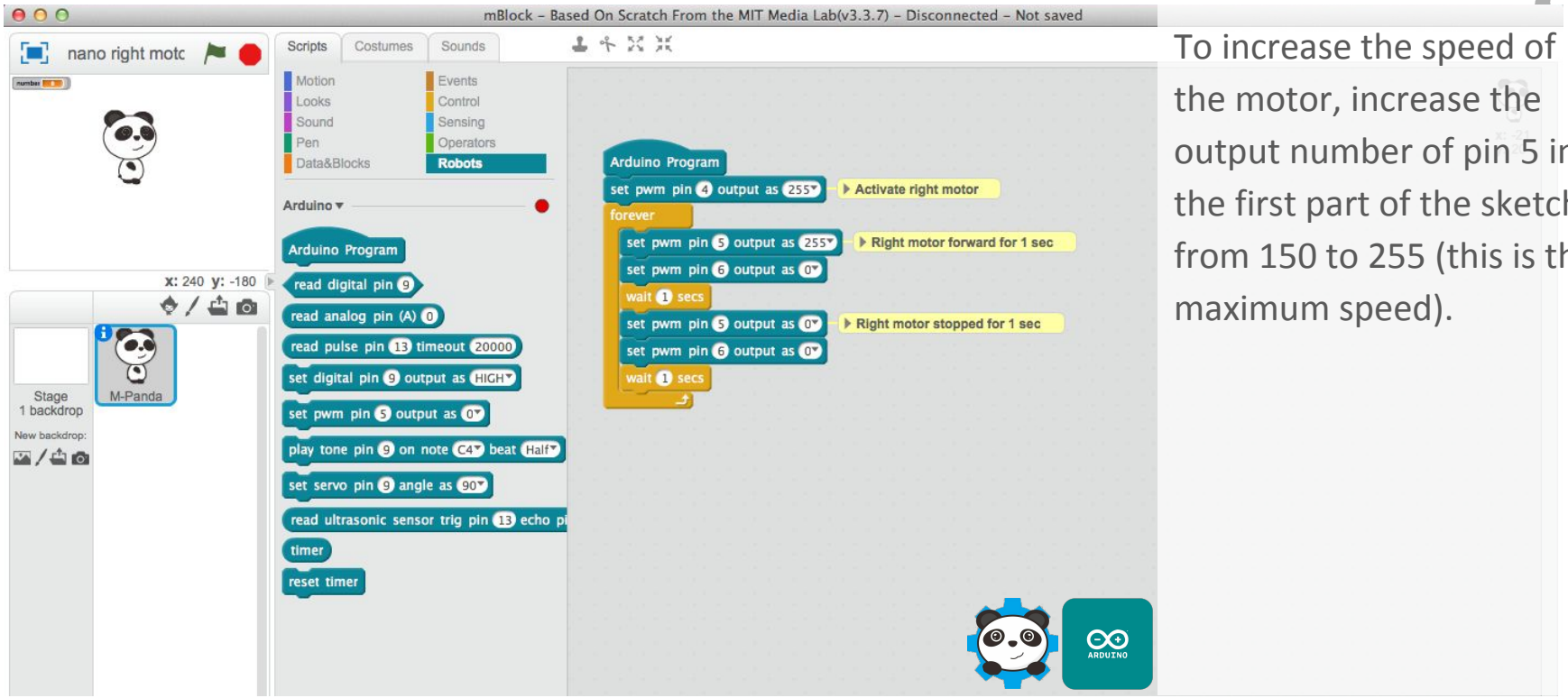
5.1 Make the right motor go forward slowly



We can control each of our two motors independently of each other. We're going to start by turning the left motor on. (Your robot will move in a circle.)

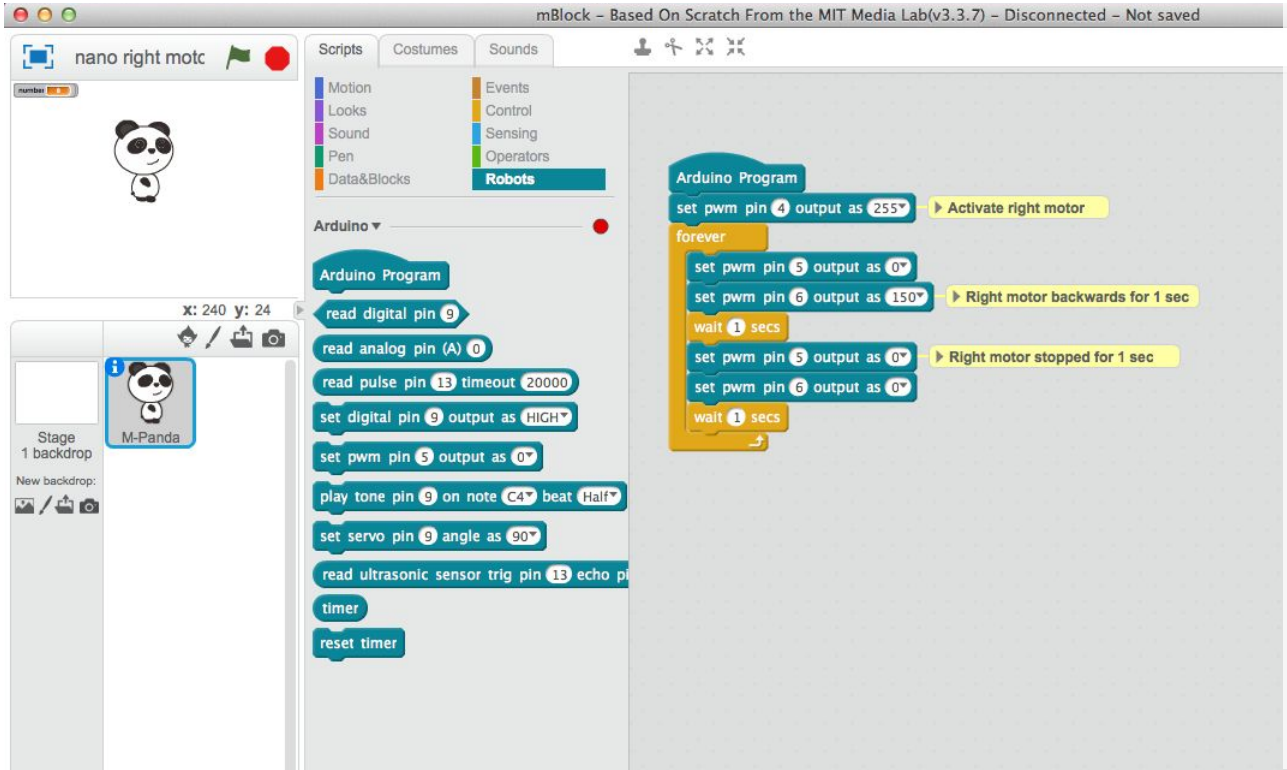
Build this sketch (taking care the the pwm pin numbers). Attach the USB cable, *connect* and upload this sketch.

5.2 Make the right motor go forward quickly



To increase the speed of the motor, increase the output number of pin 5 in the first part of the sketch from 150 to 255 (this is the maximum speed).

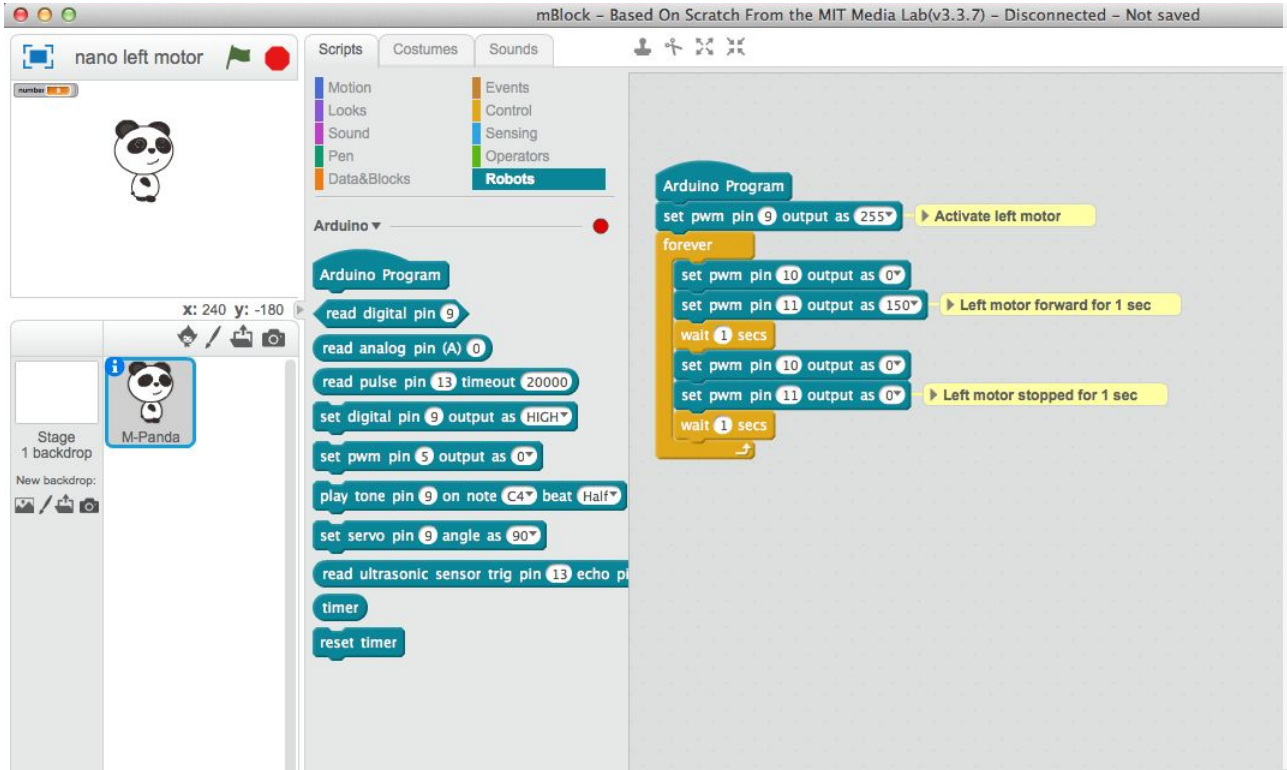
5.3 Make the right motor go backwards



For the right motor, a value of 0 on pin 5 and a value of more than 100 on pin 6 will make the motor go backwards.

Set your sketch up to make the right motor go backwards for a second and stop for a second.

5.4 Make the left motor go forward slowly



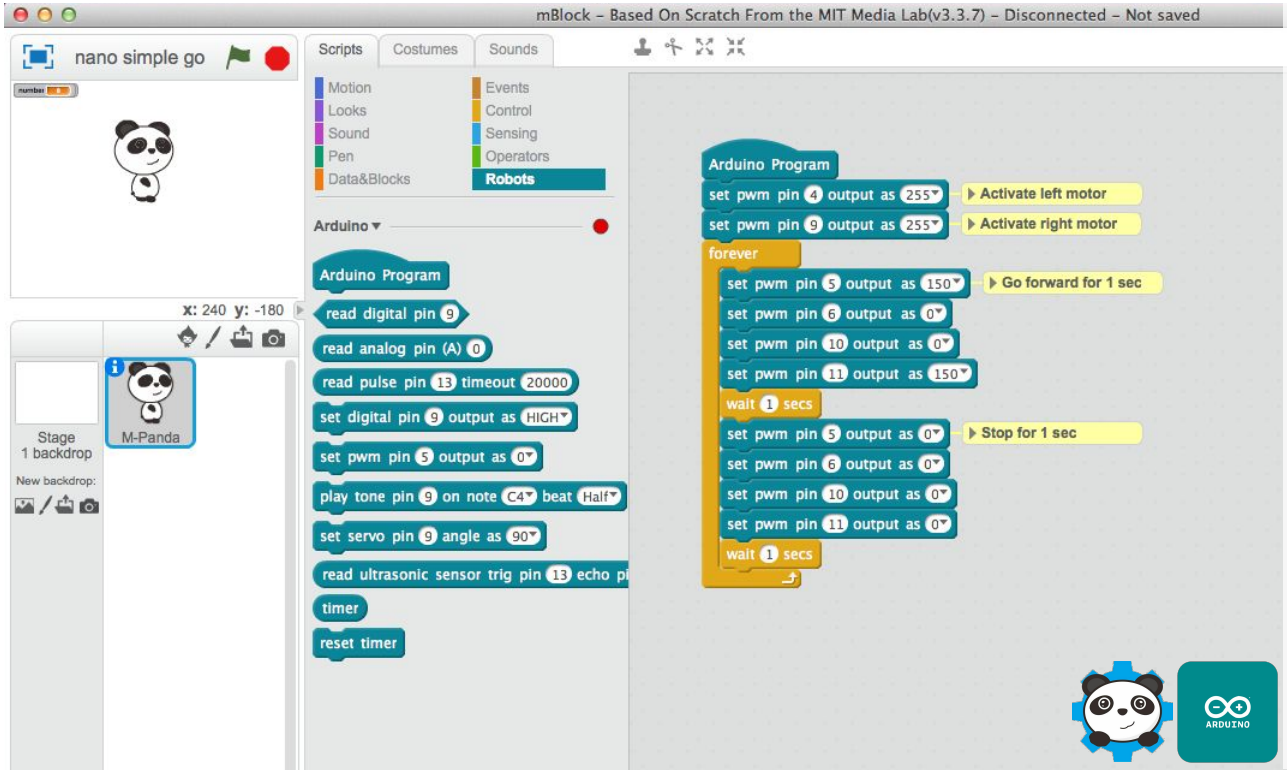
Now repeat the process with the left motor.

The left motor is controlled by the pins 9, 10, & 11.

Change the pwm pins so your sketch now controls the left motor.

Challenge: can you remember how to speed it up and slow it down? Can you remember how to make it go backwards?

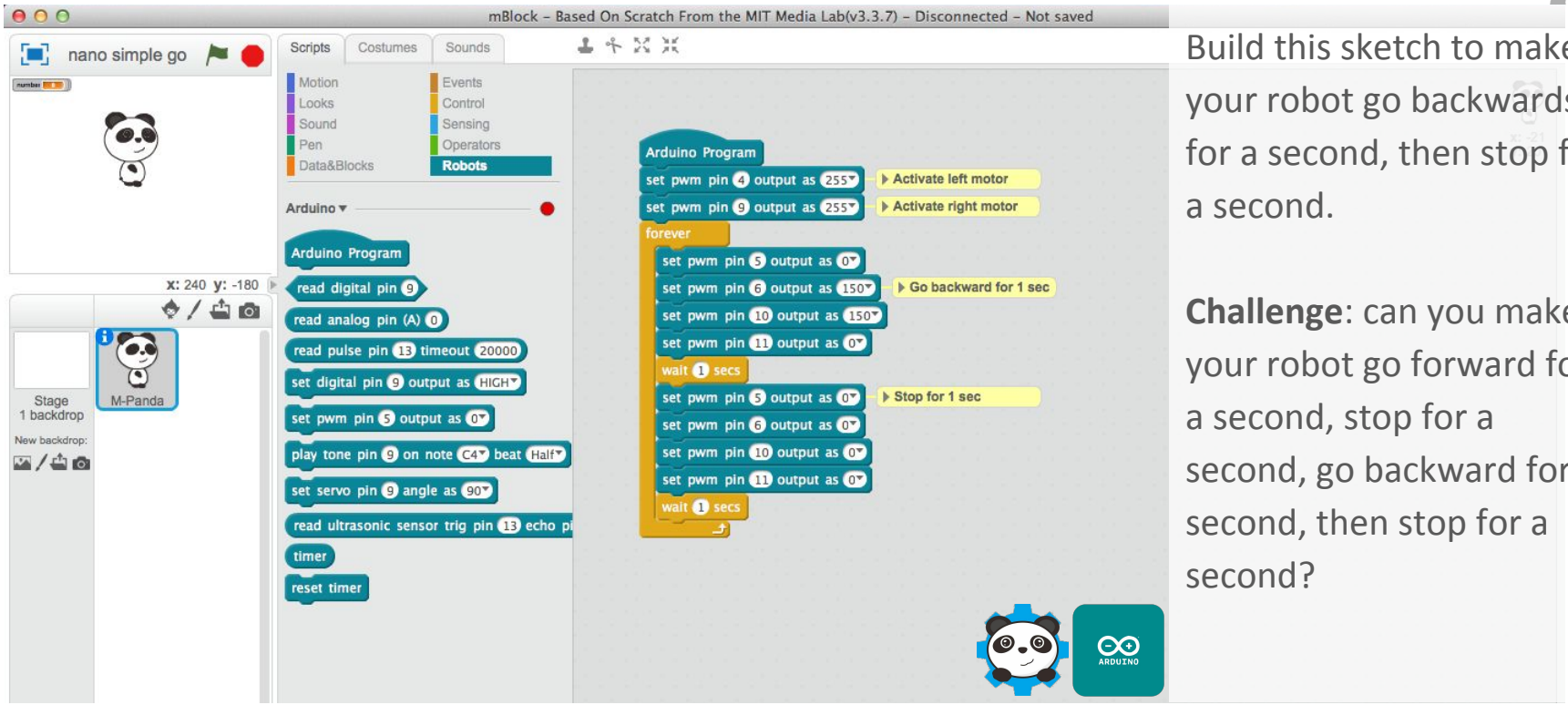
5.5 Putting it together: moving forward



Put everything you've learnt together to turn both

Challenge: can you make your robot move forward for 5 secs, turn around (using just the left or right motor) then go forward again for 5 secs?

5.6 Putting it together: moving backward



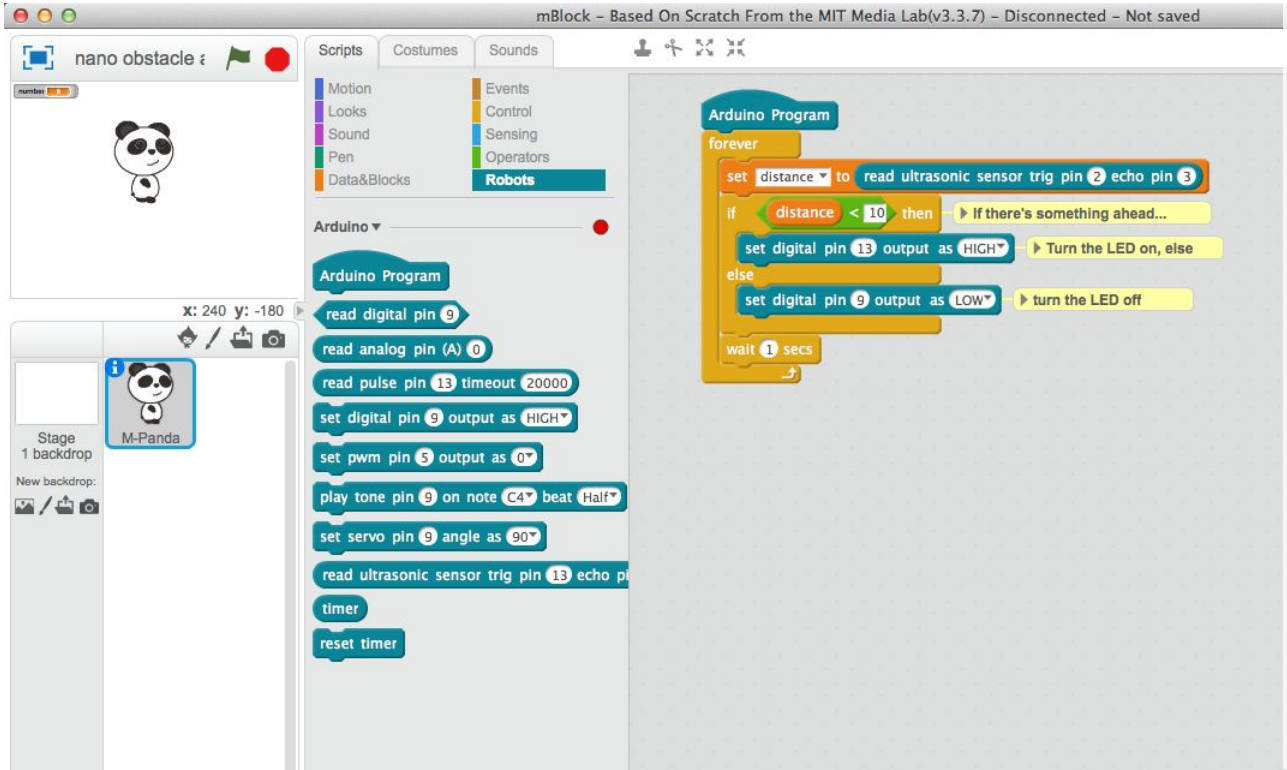
Build this sketch to make your robot go backwards for a second, then stop for a second.

Challenge: can you make your robot go forward for a second, stop for a second, go backward for a second, then stop for a second?

Chapter 6

Object-Avoiding robot

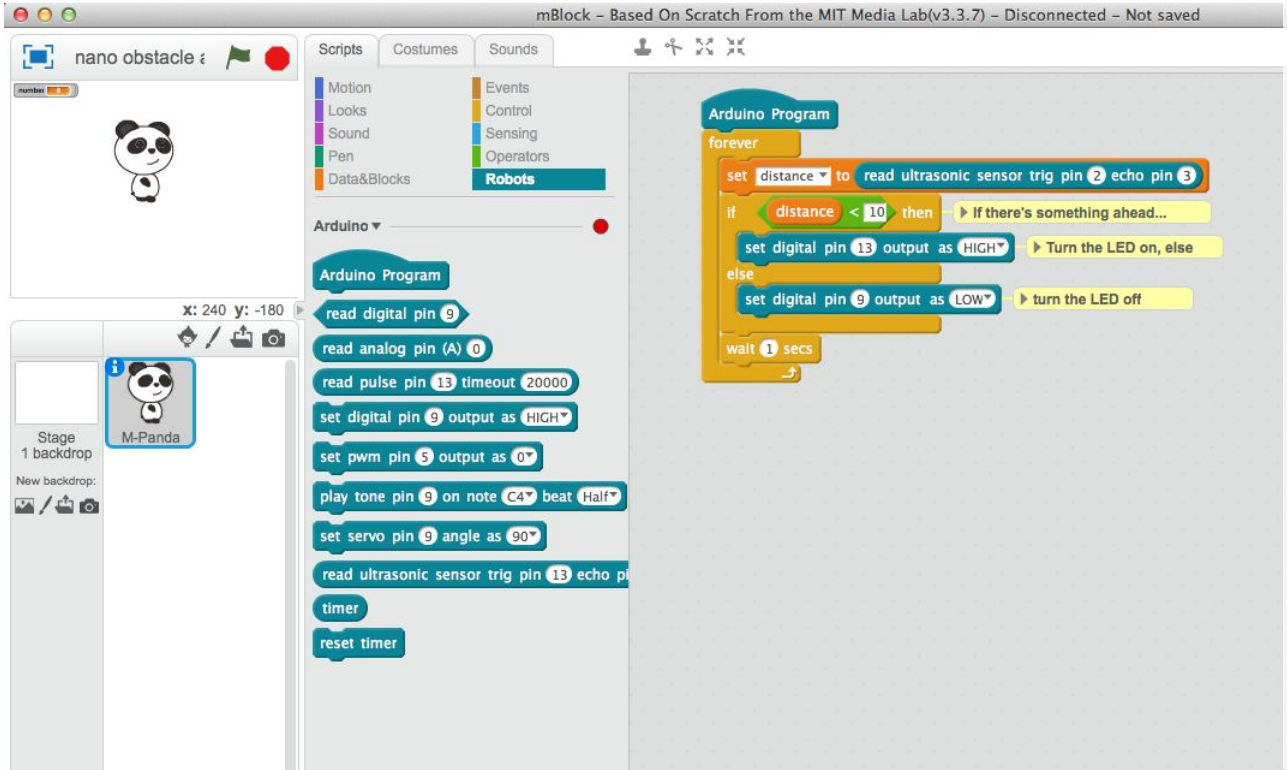
6.1 If-then statements



Remember variables from Chapter Two? We're going to use a variable to store the value returned by the ultrasonic sensor. The number will either be high (object is a long way away) or low (the object is close).

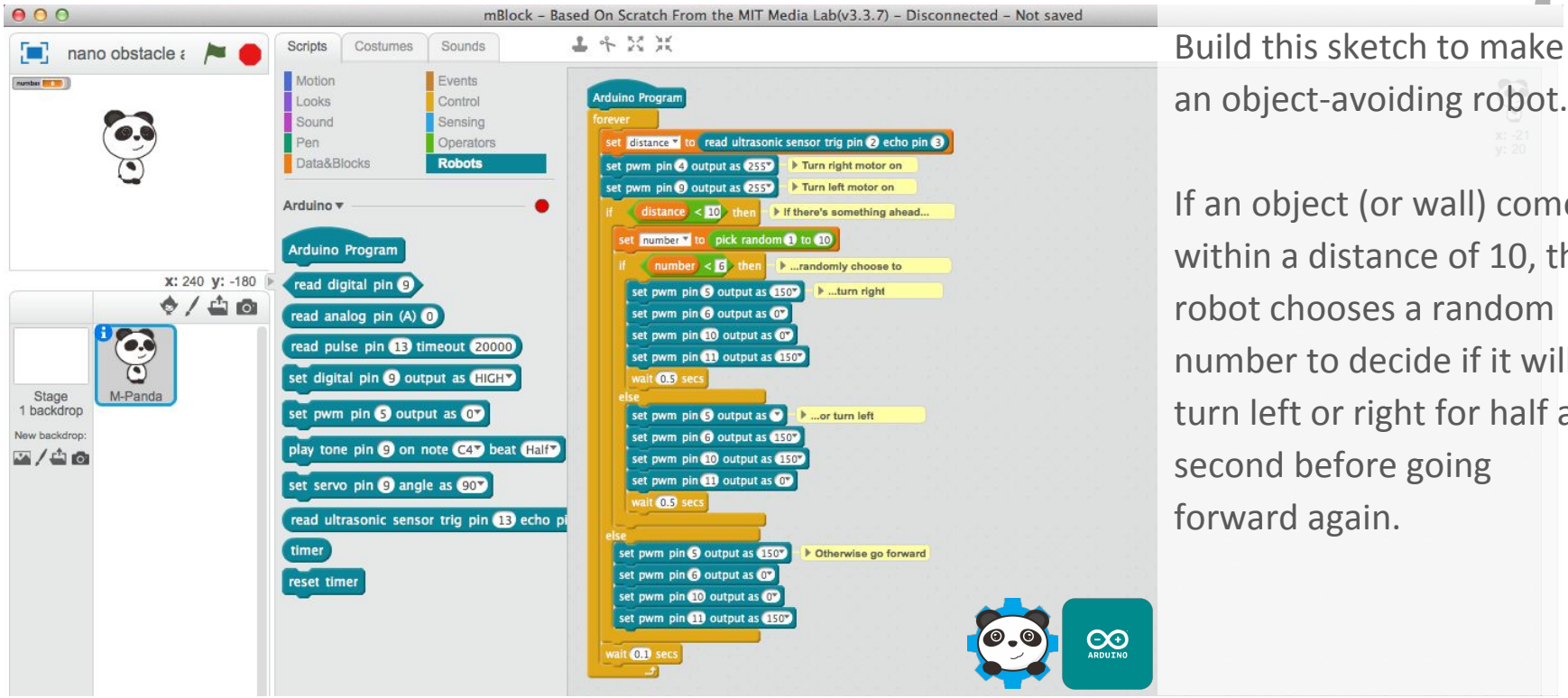
From the Data&Blocks tab, make a variable called distance and bring in the blocks to make this sketch.

6.1 If-then statements



If-then statements are often used in programming to make things happen:
IF a button is pushed THEN do something.
Set up a sketch like this to test the distance between your robot's ultrasonic sensor and any object. IF the object is close THEN the LED will turn on, ELSE the LED will turn off.

6.2 Object-avoiding robot



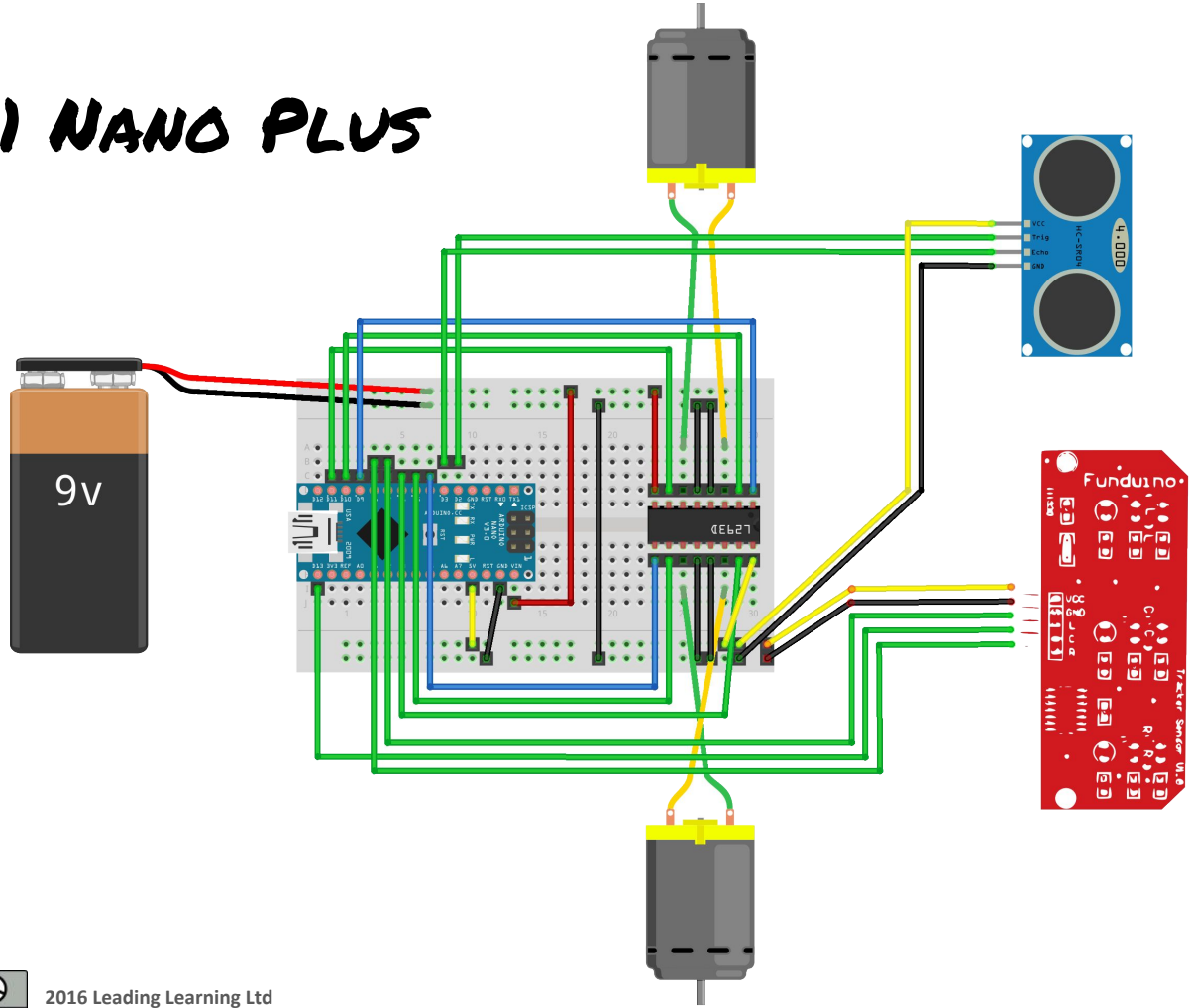
Build this sketch to make an object-avoiding robot.

If an object (or wall) comes within a distance of 10, the robot chooses a random number to decide if it will turn left or right for half a second before going forward again.

Chapter 7

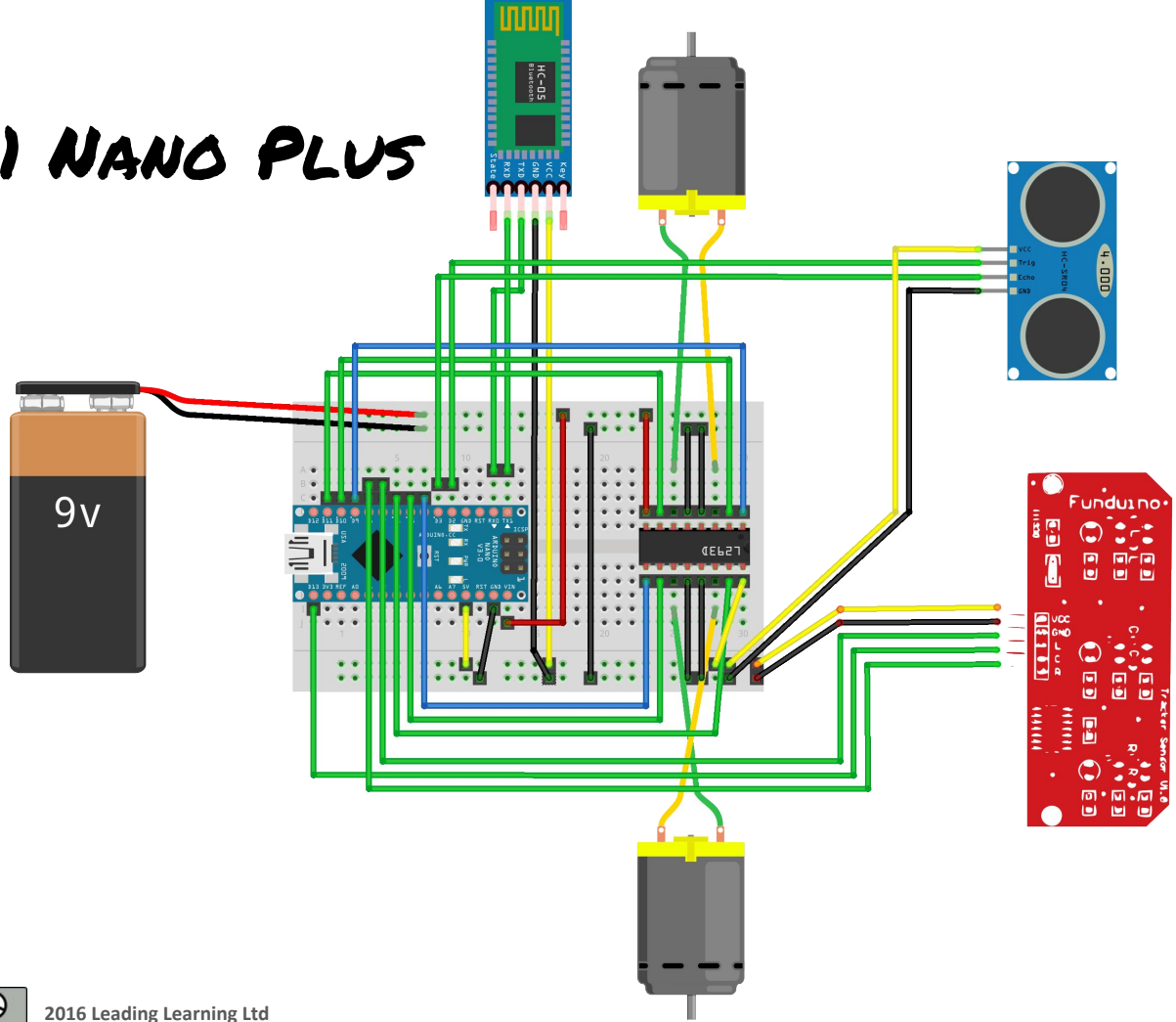
The Nano Plus

7.1 Nano Plus



The Nano Plus has an infrared sensor (for detecting colours) and a Bluetooth module (for receiving instructions from an Android Smartphone). Attach the infrared sensor to the bottom of the robot chassis using the screws provided, feed the jumpers through the hole, and connect the jumpers according to this diagram.

7.1 Nano Plus

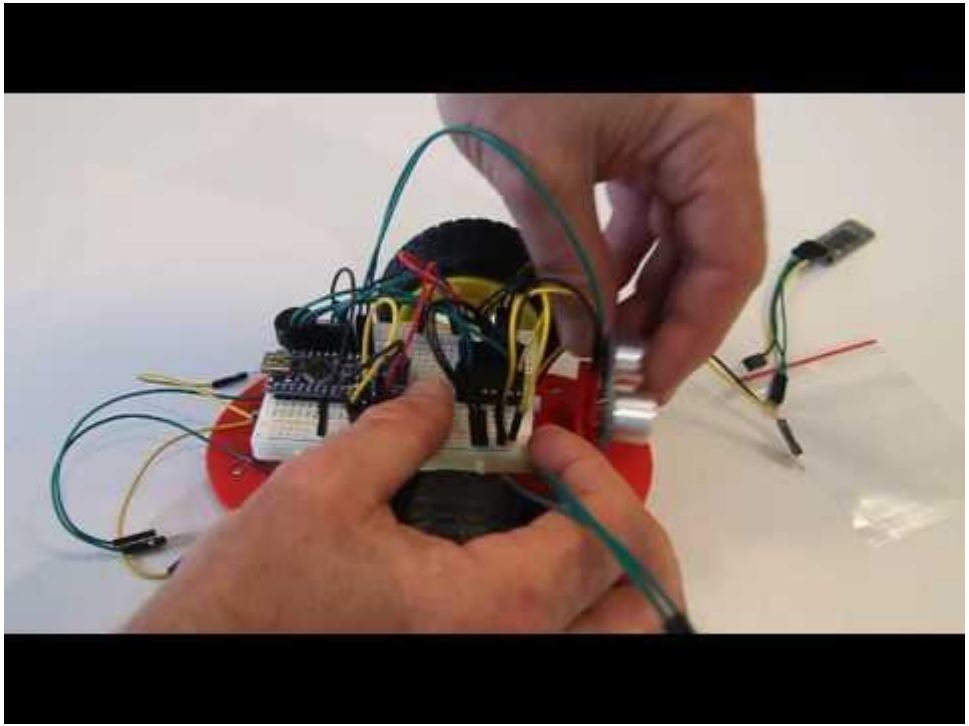


The Bluetooth module requires 5v power (yellow and black) and needs to be connected so the RXD pin on the Bluetooth module is plugged into B16 and the TXD pin is plugged into B17.

This enables a smartphone to talk to our Arduino.

fritzing

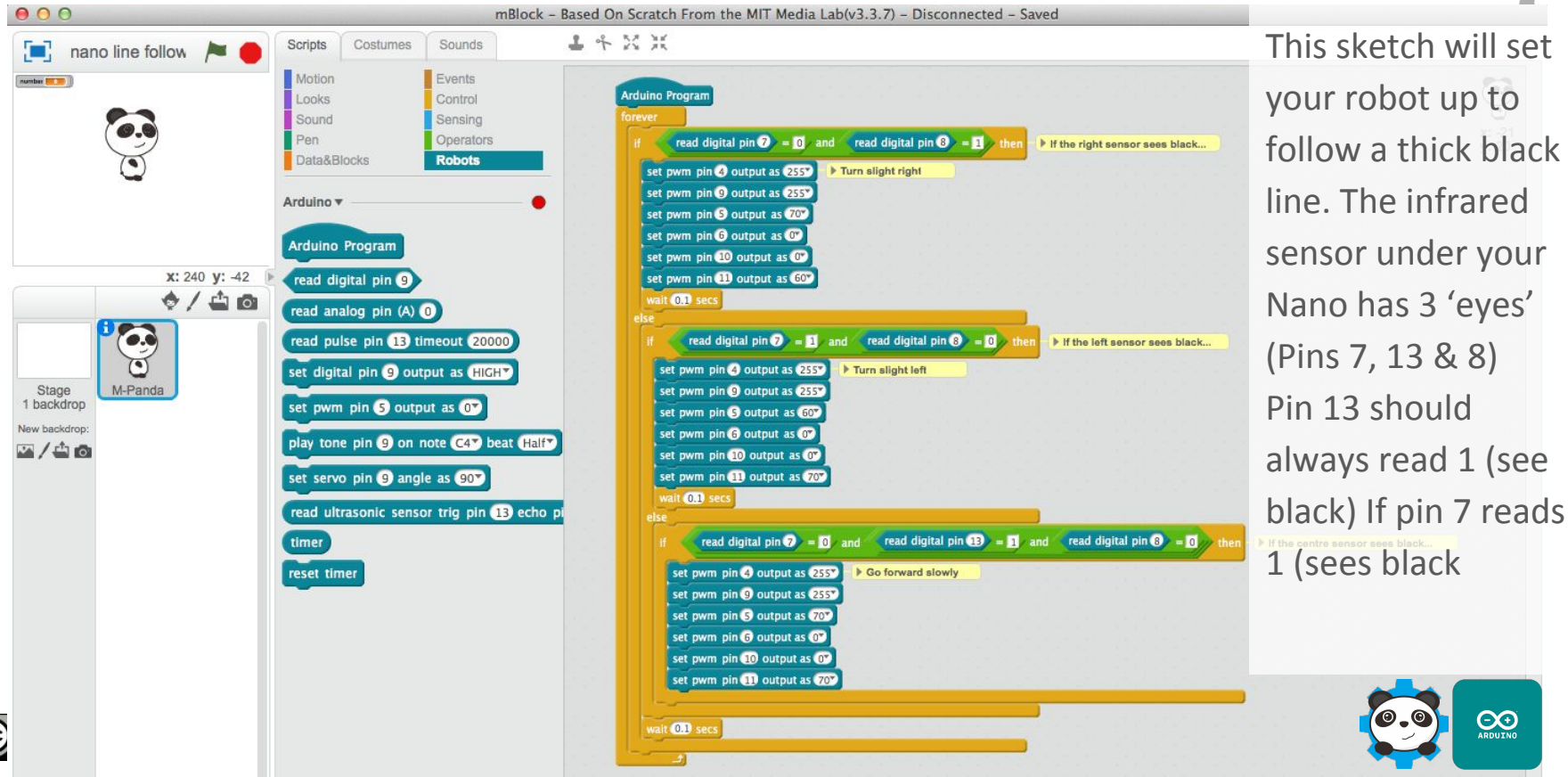
7.2 Nano Plus Assembly Video



This video might help you to assemble all of the parts of the Nano Plus.

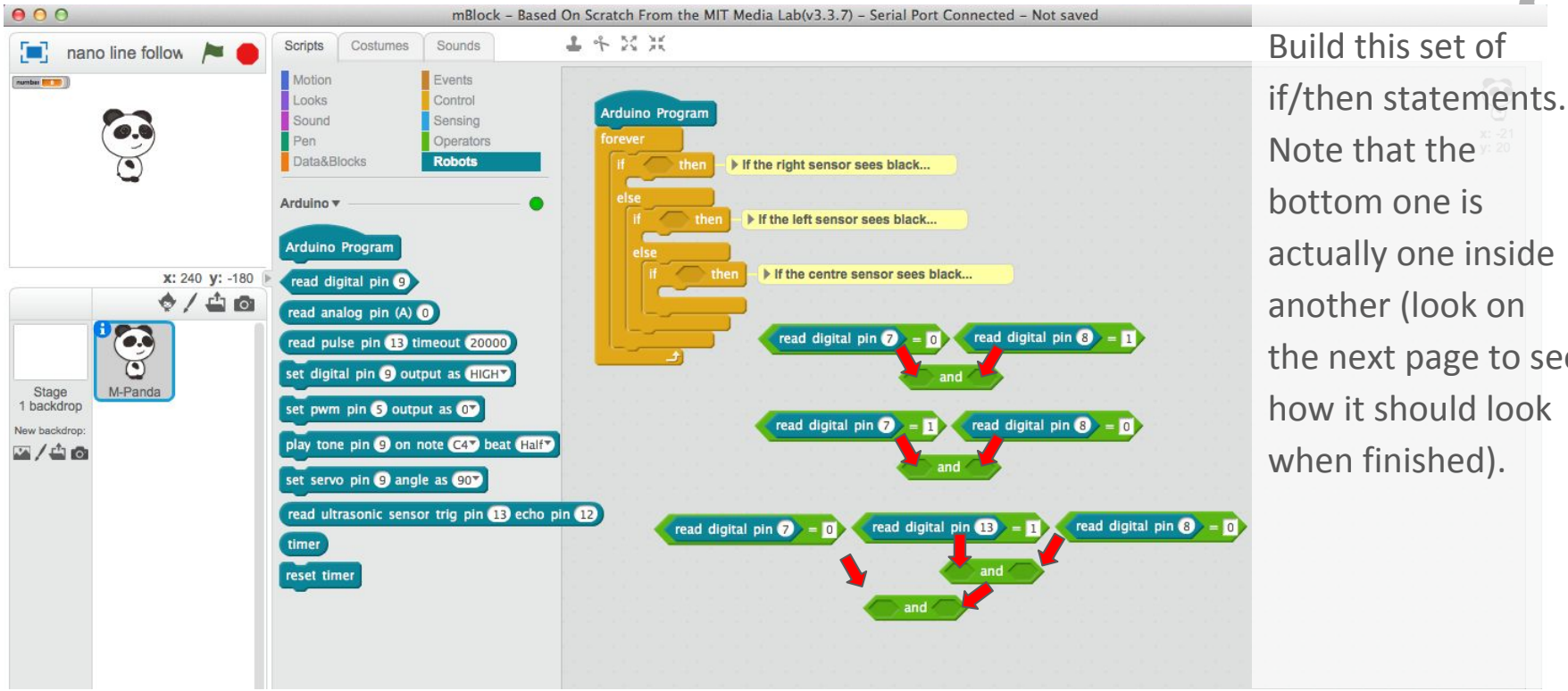
Useful Zombie fact: you can't reason with zombies. Don't even try to talk to them - just run.

7.3 Line-following robot



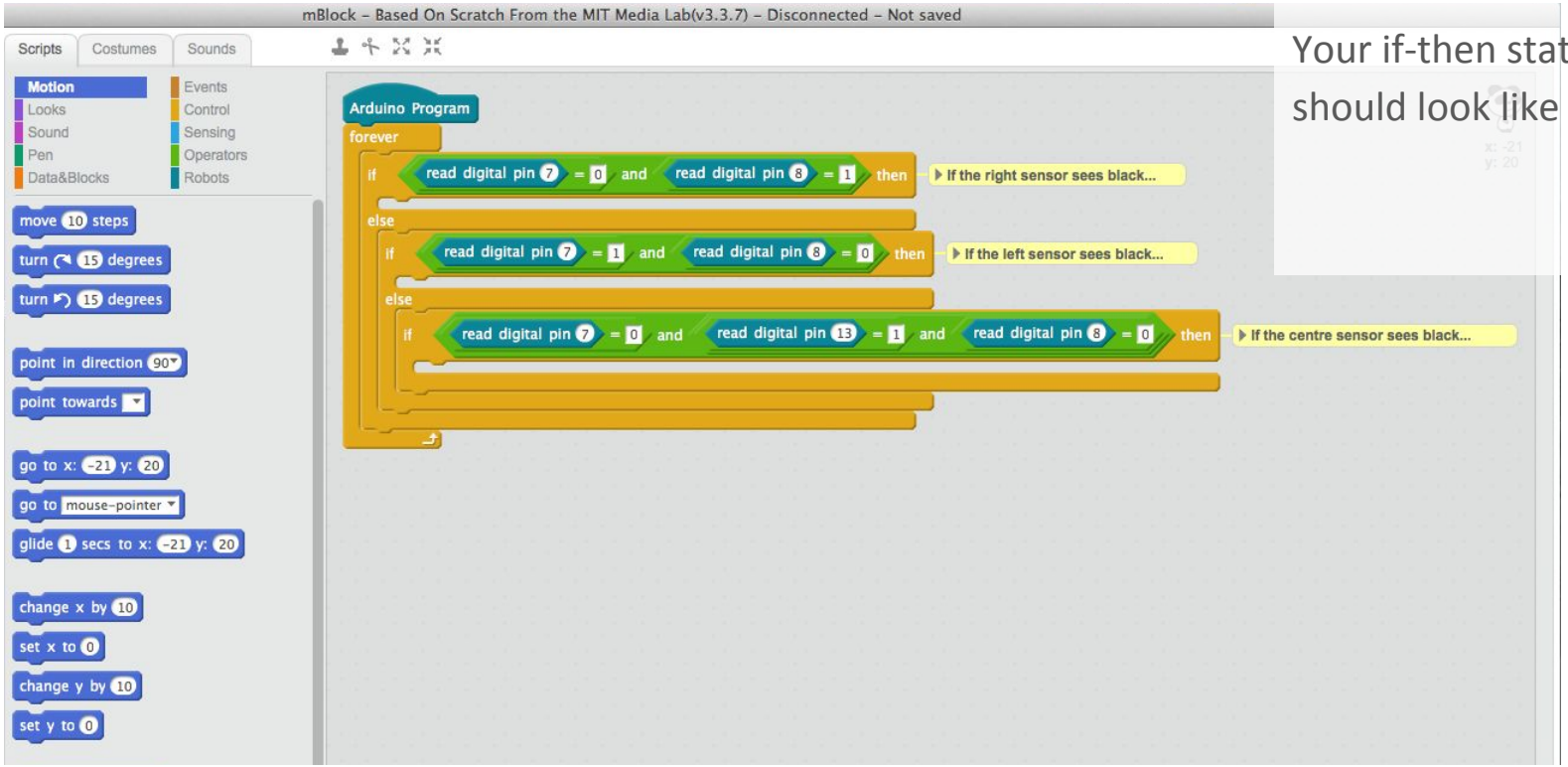
This sketch will set your robot up to follow a thick black line. The infrared sensor under your Nano has 3 ‘eyes’ (Pins 7, 13 & 8) Pin 13 should always read 1 (see black) If pin 7 reads 1 (sees black

7.3 Line-following robot



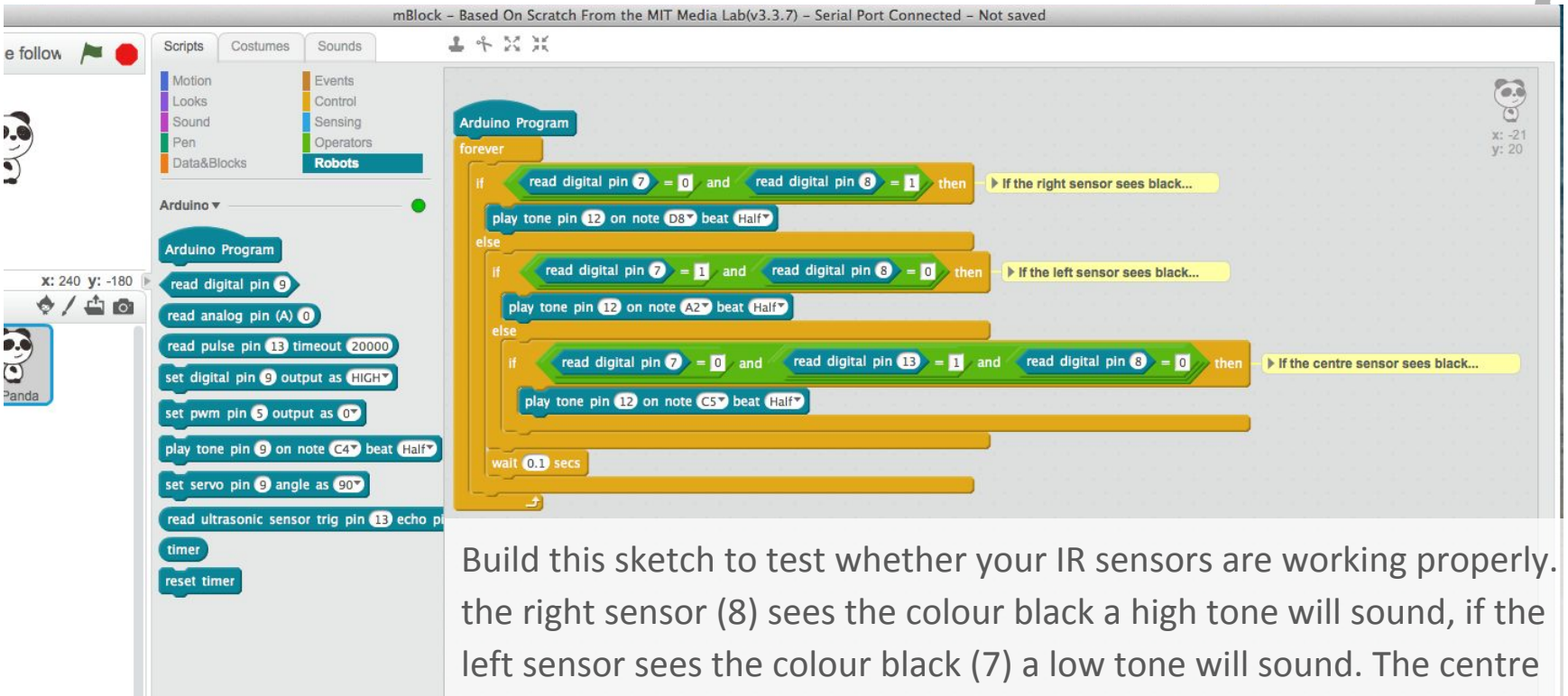
Build this set of if/then statements. Note that the bottom one is actually one inside another (look on the next page to see how it should look when finished).

7.3 Line-following robot



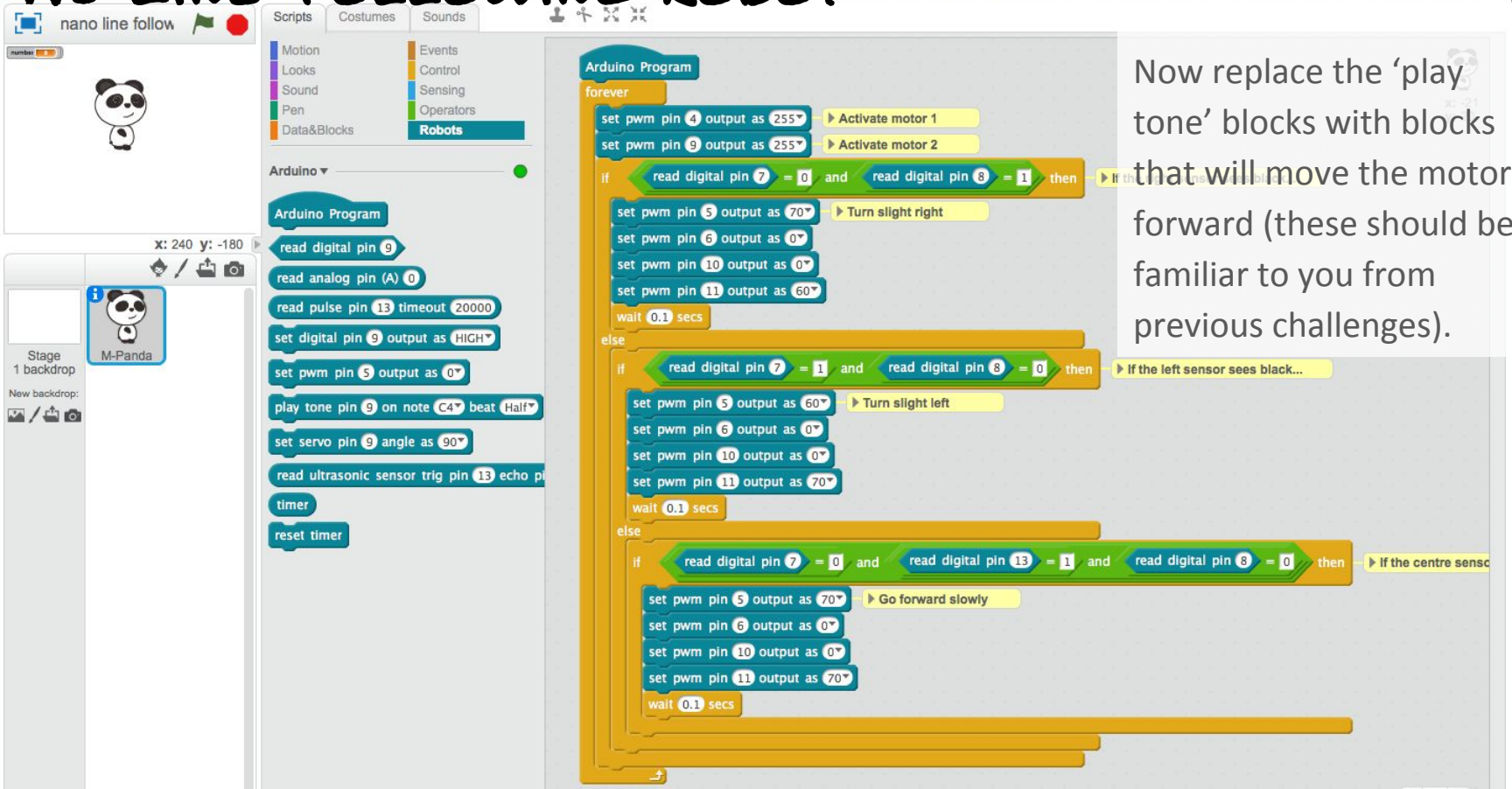
Your if-then statements should look like this.

7.4 Line-following robot



Build this sketch to test whether your IR sensors are working properly. If the right sensor (8) sees the colour black a high tone will sound, if the left sensor sees the colour black (7) a low tone will sound. The centre sensor (13) will play a mid-range tone.

7.5 Line-following robot

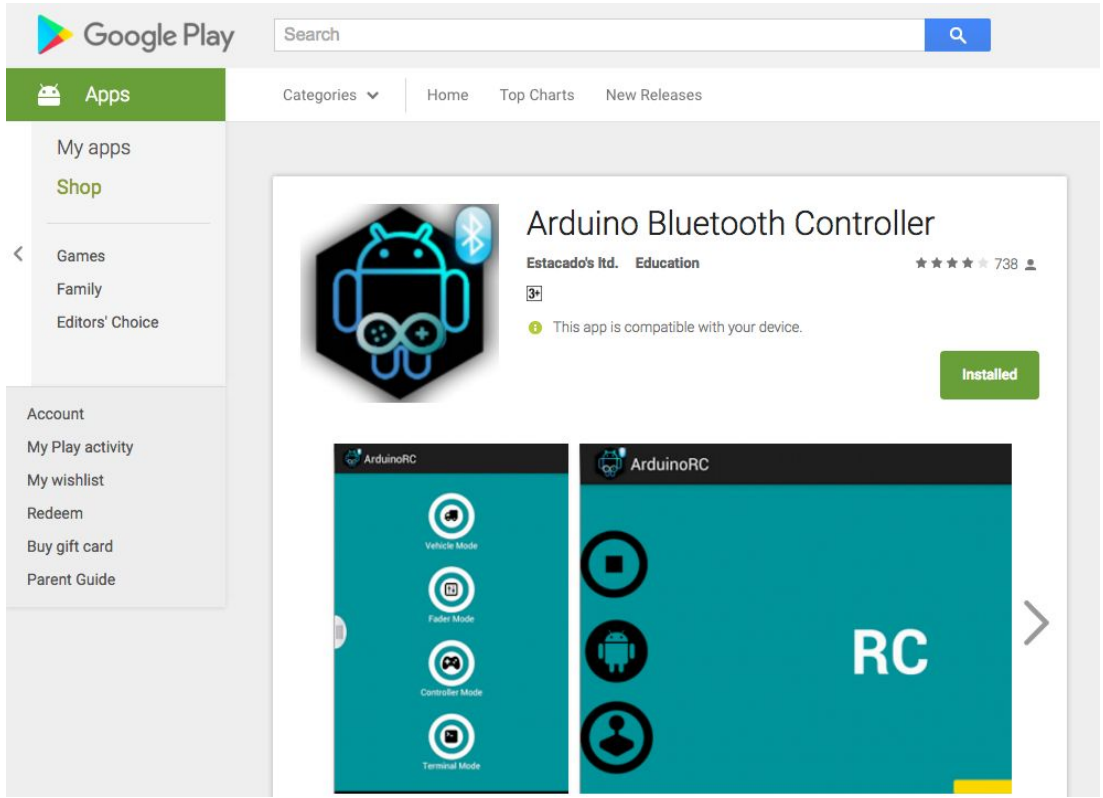


Now replace the 'play tone' blocks with blocks that will move the motors forward (these should be familiar to you from previous challenges).

Chapter 8

Smartphone-controlled robot

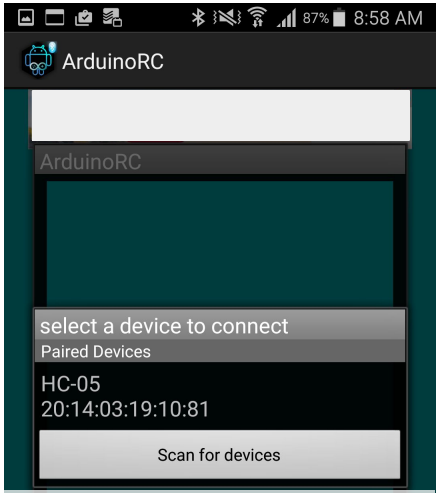
8.1 Install the Arduino Bluetooth App



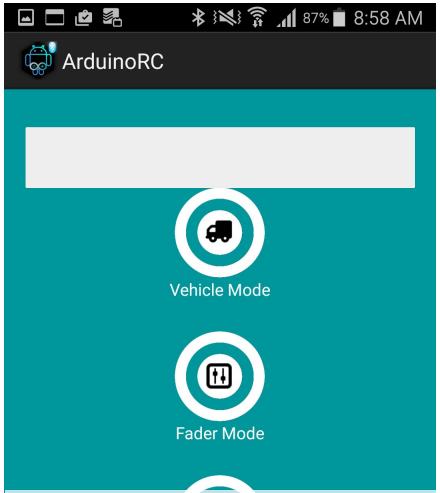
Install the *Arduino Bluetooth Controller* app from the Google Play store.
We're going to use this to control our robot.

Unfortunately there isn't an app for iOS that we can use.

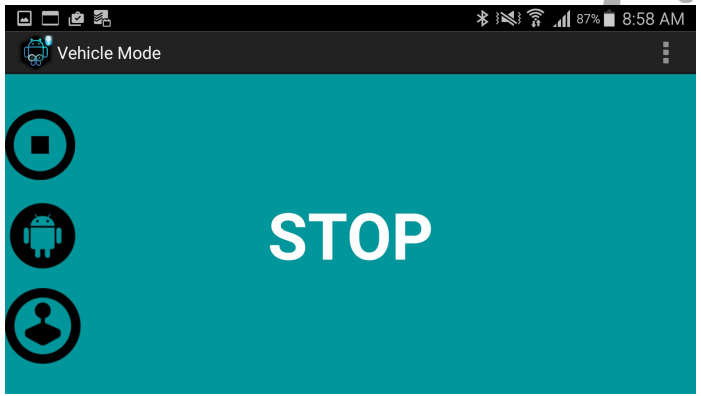
8.2 Set up the bluetooth app



Open the app, confirm your UUID and scan for devices. Look for one called HC-05. Use 1234 for a pairing code if it asks for one. Ensure your robot is powered on.

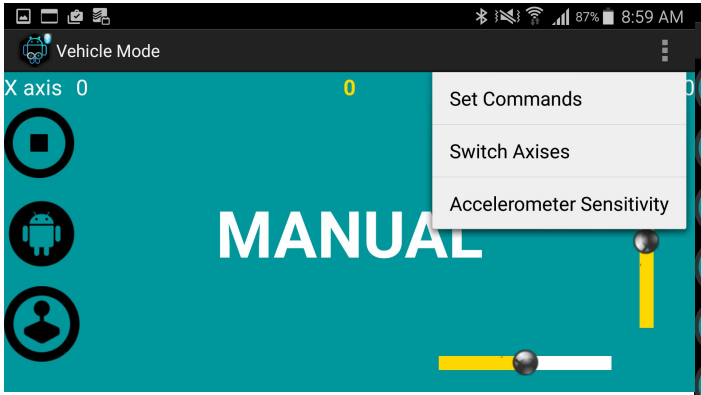


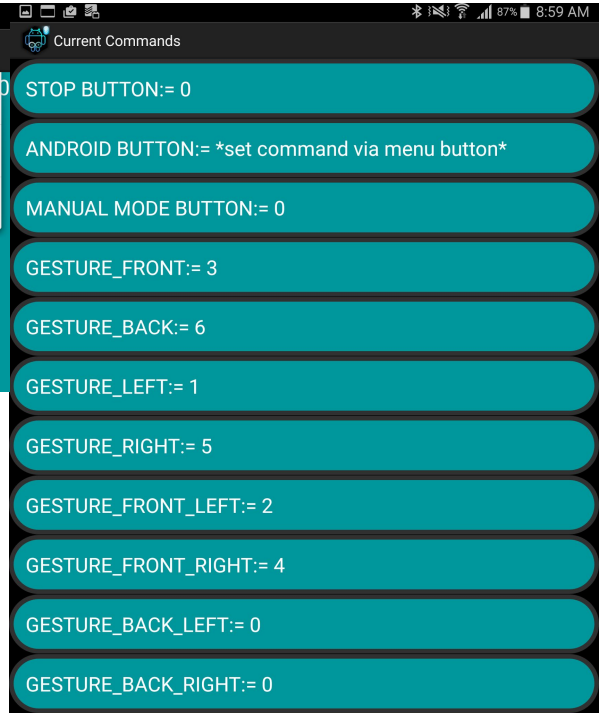
Once your phone is paired with and connected to your Bluetooth module, choose 'Vehicle Mode'.



When you see this screen, choose the joystick option (bottom).

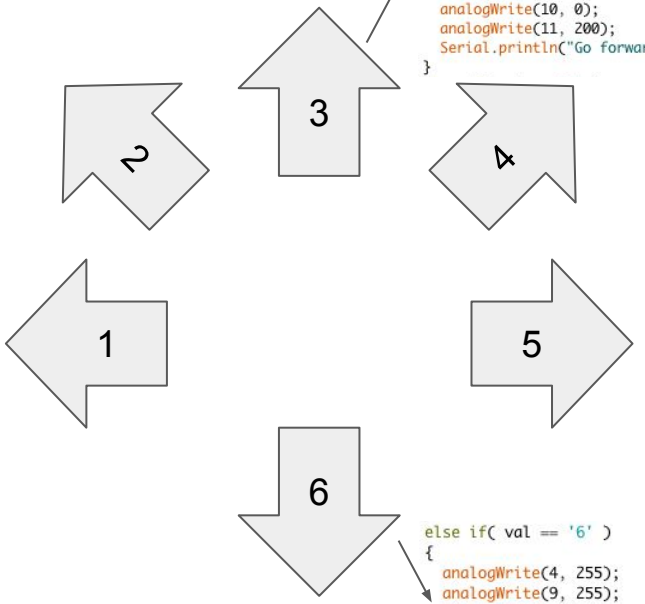
8.2 Set up the bluetooth app





From the menu on the top right, choose ‘Set Commands’. Go through the process of setting up each of the commands so they have these numbers attached to them (GESTURE_FRONT should be 3, and so on.)

8.2 Set up the bluetooth app



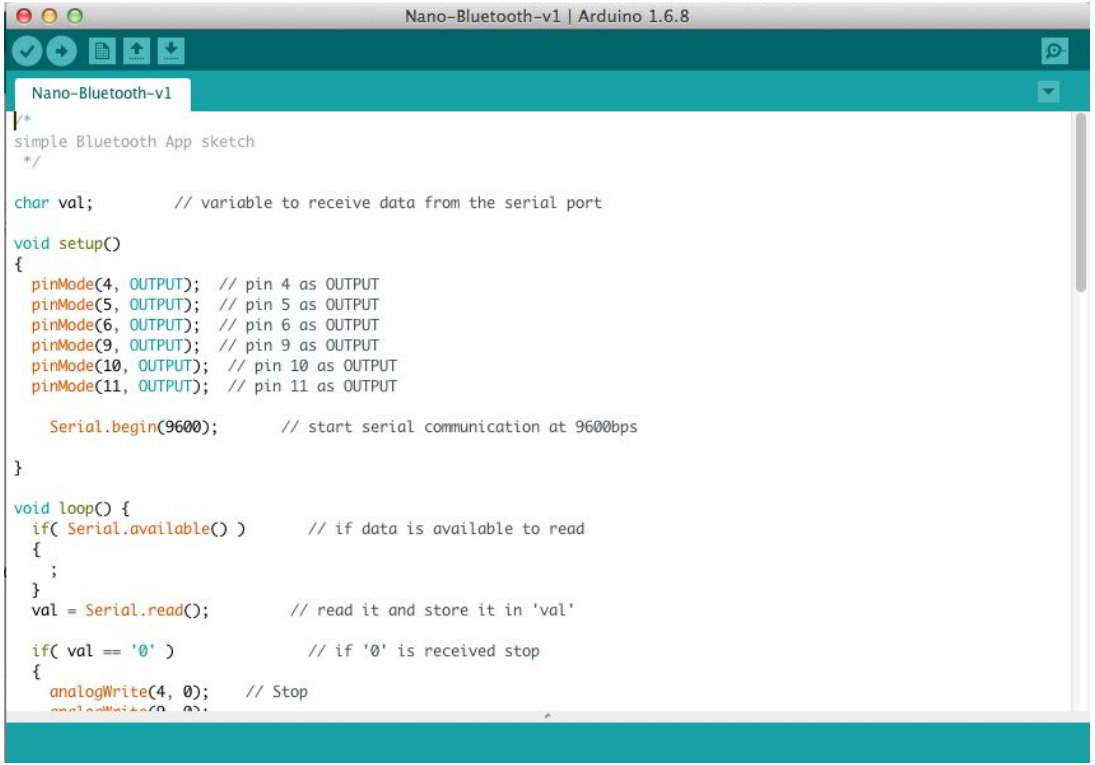

```
else if( val == '3' )              // if '3' is received go forward
{
  analogWrite(4, 255);    // go forward
  analogWrite(9, 255);
  analogWrite(5, 200);
  analogWrite(6, 0);
  analogWrite(10, 0);
  analogWrite(11, 200);
  Serial.println("Go forward");
}
```

```
else if( val == '6' )              // if '6' is received go backwards
{
  analogWrite(4, 255);    // go backwards
  analogWrite(9, 255);
  analogWrite(5, 0);
  analogWrite(6, 150);
  analogWrite(10, 150);
  analogWrite(11, 0);
  Serial.println("Go backwards");
}
```

We're going to send these numbers to our robot and make it move in a certain direction. When it gets a '3' from our phone, it'll go forward; when it gets a '6' it will go backwards, and so on.

You might recognise this code from previous challenges.

8.3 Upload the sketch



We're going to use the Arduino software (rather than mBlock) to programme our robot for Smartphone control. You should already have it installed on your computer, but if you don't, you can get it from https://www.arduino.cc/

Download this sketch, open it in Arduino and upload to your Nano.

8.4 Control your Zombiebot

If you've uploaded the sketch and set up your smartphone correctly, you should be able to do this with your smartphone.

Accelerometers are small devices in phones that detect changes in the way you are holding your phone. We're using them to tell your Zombiebot which direction to move.

Image credits:

https://commons.wikimedia.org/wiki/File:Emoji_u1f45f.svg
https://commons.wikimedia.org/wiki/File:Baseball_bat.svg
https://commons.wikimedia.org/wiki/File:Sports_icon.png
https://commons.wikimedia.org/wiki/File:Sport_balls.svg
https://upload.wikimedia.org/wikipedia/commons/e/ef/Hockey_Stick_and_Puck.png
https://upload.wikimedia.org/wikipedia/commons/4/4a/Nike_Free%2B_3_running_shoe.jpg
https://pixabay.com/en/tennis-shoes-running-shoes-shoes-297150/
https://pixabay.com/static/uploads/photo/2013/07/12/14/09/emergency-doctor-147857_640.png