C Lab 1

Arduino

This lab gives you a spec that describes a program for you to build, using the knowledge you've gained over the last few chapters.

This project is bigger than the ones you've seen so far. So read the whole thing before you get started, and give yourself a little time. And don't worry if you get stuck. There are no new C concepts in here, so you can move on in the book and come back to the lab later.

We've filled in a few design details for you, and we've made sure you've got all the pieces you need to write the code. You can even build the physical device.

It's up to you to finish the job, but we won't give you the code for the answer.

The spec: make your houseplant talk

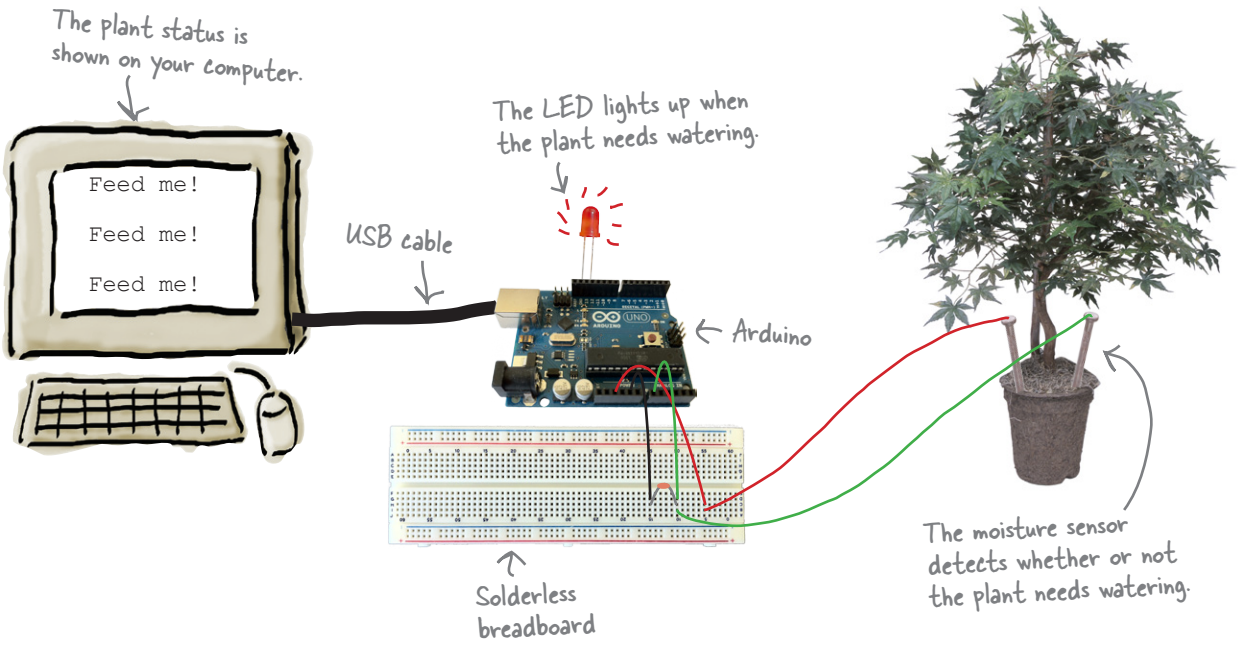
Ever wished your plants could tell you when they need watering? Well, with an Arduino they can! In this lab, you'll create an Arduino-powered plant monitor, all coded in C.

Here's what you're going to build.

The physical device

The plant monitor has a moisture sensor that measures how wet your plant's soil is. If the plant needs watering, an LED lights up until the plant's been watered, and the string "Feed me!" is repeatedly sent to your computer.

When the plant has been watered, the LED switches off and the string "Thank you, Seymour!" is sent once to your computer.



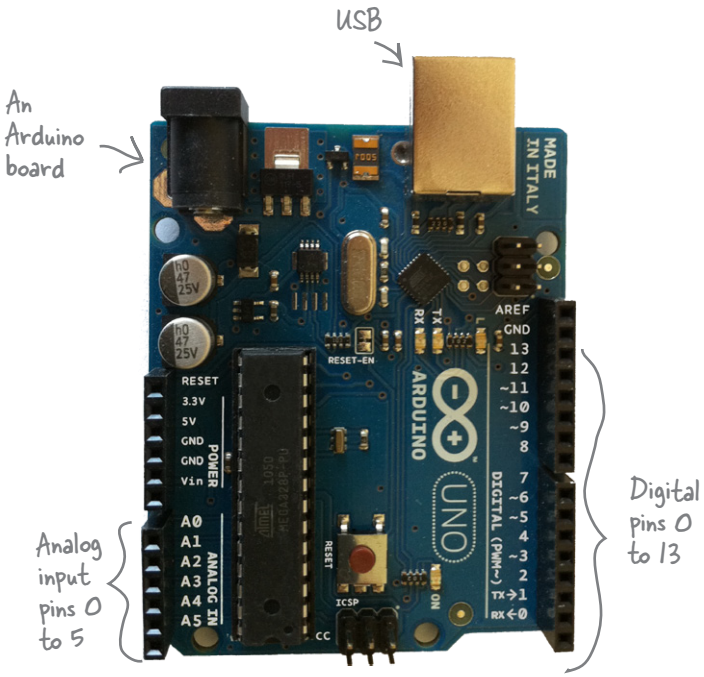
The Arduino

The brains of the plant monitor is an **Arduino**. An Arduino is a small micro-controller-based open source platform for electronic prototyping. You can connect it to sensors that pick up information about the world around it, and actuators that respond. All of this is controlled by code you write in C.

The Arduino board has 14 digital IO pins, which can be inputs or outputs. These tend to be used for reading on or off values, or switching actuators on or off.

The board also has six analog input pins, which take voltage readings from a sensor.

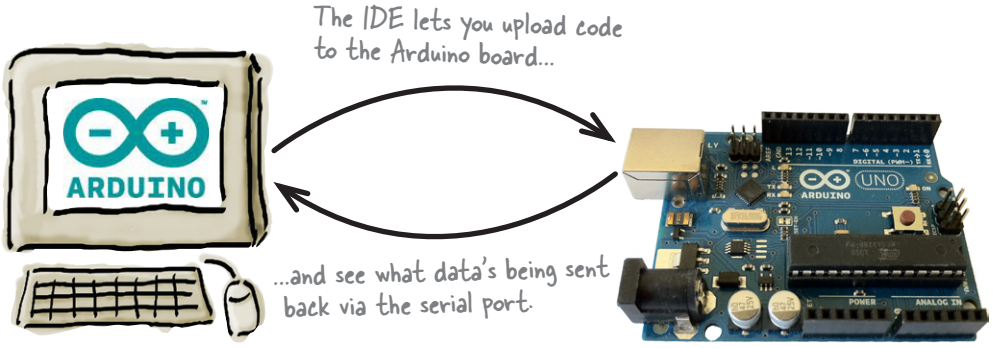
The board can take power from your computer's USB port.



The Arduino IDE

You write your C code in an Arduino IDE. The IDE allows you to verify and compile your code, and then upload it to the Arduino itself via your USB port. The IDE also has a built-in serial monitor so that you can see what data the Arduino is sending back (if any).

The Arduino IDE is free, and you can get hold of a copy from *www.arduino.cc/en/Main/Software*.



Build the physical device

You start by building the physical device. While this bit's optional, we really recommend that you give it a go. Your plants will thank you for it.

We used an Arduino Uno.

You will need:

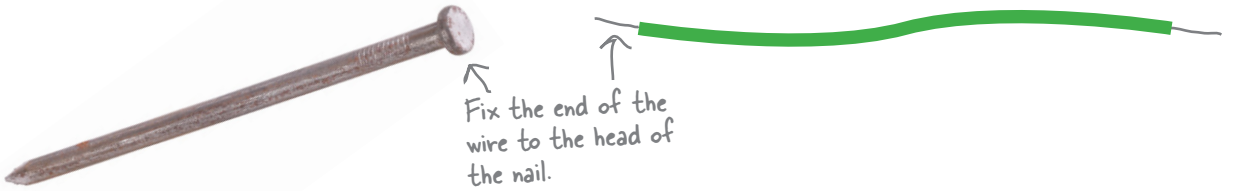
- 1 Arduino
- 1 solderless breadboard
- 1 LED
- 1 10K Ohm resistor
- 2 galvanized nails
- 3 short pieces of jumper wire
- 2 long pieces of jumper wire

Build the moisture sensor

Take a long piece of jumper wire and attach it to the head of one of the galvanized nails. You can either wrap the wire around the nail or solder it in place.

Once you've done that, attach another long piece of jumper wire to the second galvanized nail.

The moisture sensor works by checking the conductivity between the two nails. If the conductivity is high, the moisture content must be high. If it's low, the moisture content must be low.

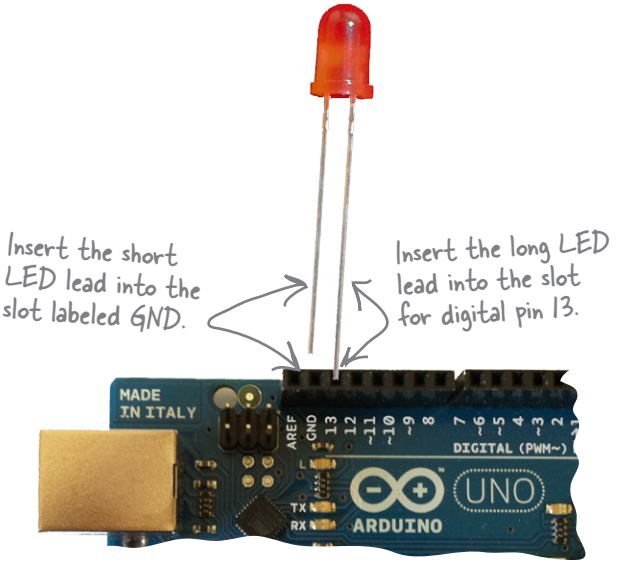


Connect the LED

Look at the LED. You will see that it has one longer (positive) lead and one shorter (negative) lead.

Now take a close look at the Arduino. You will see that along one edge there are slots for 14 digital pins labeled 0–13, and another one next to it labeled GND. Put the long positive lead of the LED into the slot labeled 13, and the shorter negative lead into the slot labeled GND.

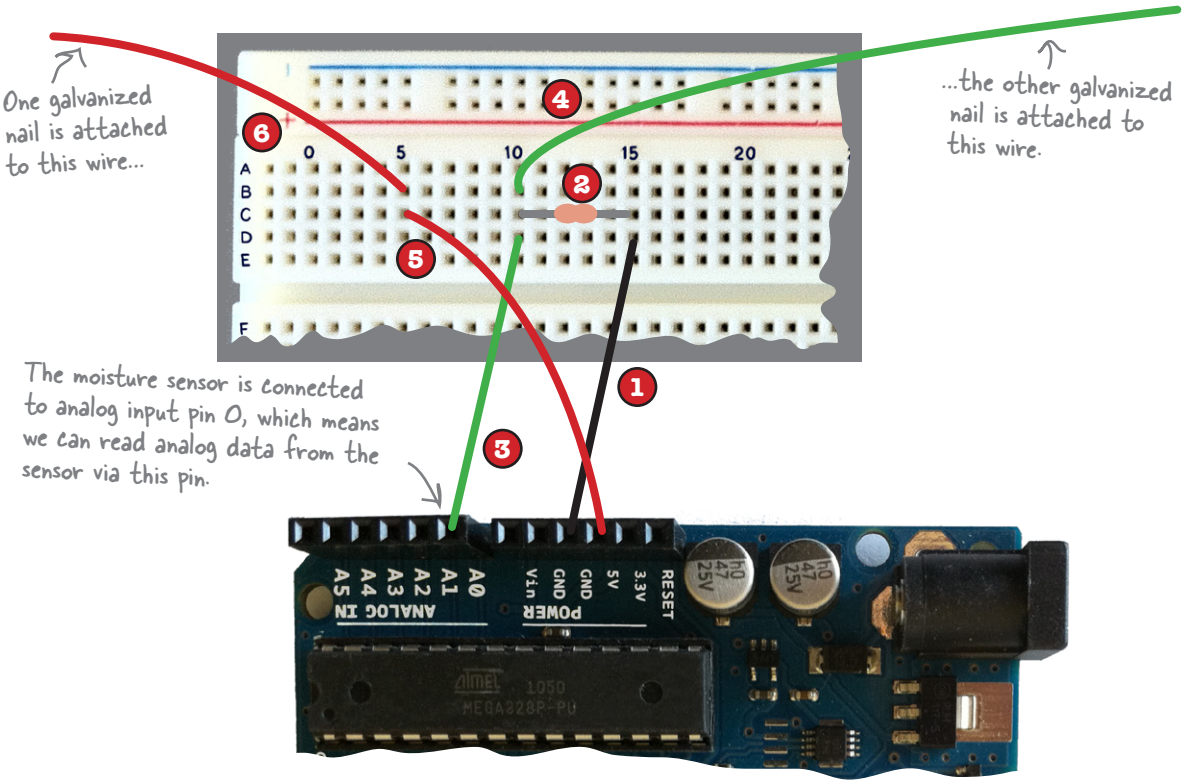
This means that the LED can be controlled through digital pin 13.



Connect the moisture sensor

Connect the moisture sensor as shown below:

1. Connect a short jumper wire from the GND pin on the Arduino to slot D15 on the breadboard.
2. Connect the 10K Ohm resistor from slot C15 on the breadboard to slot C10.
3. Connect a short jumper wire from the 0 analog input pin to slot D10 on the breadboard.
4. Take one of the galvanized nails, and connect the wire attached to it to slot B10.
5. Connect a short jumper wire from the 5V pin on the Arduino to slot C5 on the breadboard.
6. Take the other galvanized nail, and connect the wire attached to it to slot B5.



That's the physical Arduino built. Now for the C code...

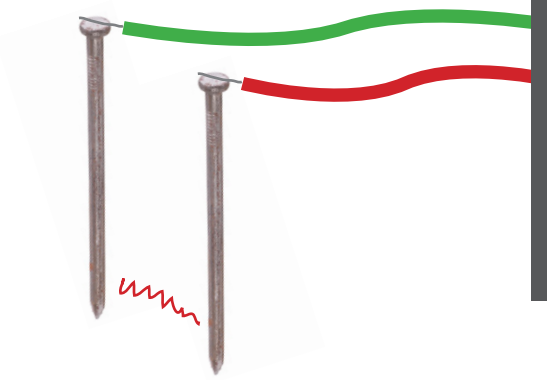
Here's what your code should do

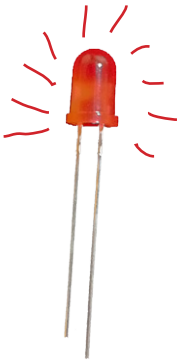
Your Arduino C code should do the following.

Read from the moisture sensor

The moisture sensor is connected to an analog input pin. You will need to read analog values from this pin.

Here at the lab, we've found that our plants generally need watering when the value goes below 800, but your plant's requirements may be different—say, if it's a cactus.





Write to the LED

The LED is connected to a digital pin.

When the plant doesn't need any more water, write to the digital pin the LED is connected to, and get it to switch off the LED.

When the plant needs watering, write to the digital pin and get it to switch on the LED. For extra credit, get it to flash. Even better, get it to flash when the conditions are borderline.

Write to the serial port

When the plant needs watering, repeatedly write the string "Feed me!" to the computer serial port.

When the plant has enough water, write the string "Thank you, Seymour!" to the serial port once.

Assume that the Arduino is plugged in to the computer USB socket.

Here's what your C code should look like

An Arduino C program has a specific structure. Your program must implement the following:

```
void setup()

{

/*This is called when the program starts. It
basically sets up the board. Put any initialization
code here.*/

}

void loop()

{

/*This is where your main code goes. This function
loops over and over, and allows you to respond to
input from your sensors. It only stops running when
the board is switched off*/

}
```

You can add extra functions and declarations if you like, but without these two functions the code won't work.

The easiest way of writing the Arduino C code is with the Arduino IDE. The IDE allows you to verify and compile your code, and then upload your completed program to the Arduino board, where you'll be able to see it running.

The Arduino IDE comes with a library of Arduino functions and includes lots of handy code examples. Turn the page to see a list of the functions you'll find most useful when creating Arduino.

Here are some useful Arduino functions

You'll need some of these to write the program.

`void pinMode(int pin, int mode)`

Tells the Arduino whether the digital `pin` is an input or output. `mode` can be either `INPUT` or `OUTPUT`.

`int digitalRead(int pin)`

Reads the value from the digital pin. The return value can be either `HIGH` or `LOW`.

`void digitalWrite(int pin, int value)`

Writes a value to a digital pin. *`value`* can be either `HIGH` or `LOW`.

`int analogRead(int pin)`

Reads the value from an analog pin. The return value is between 0 and 1023.

`void analogWrite(int pin, int value)`

Writes an analog value to a pin. *`value`* is between 0 and 255.

`void Serial.begin(long speed)`

Tells the Arduino to start sending and receiving serial data at *`speed`* bits per second. You usually set *`speed`* to 9600.

`void Serial.println(val)`

Prints data to the serial port. *`val`* can be any data type.

`void delay(long interval)`

Pauses the program for *`interval`* milliseconds.

The finished product

You'll know your Arduino project is complete when you put the moisture sensor in your plant's soil, connect the Arduino to your computer, and start getting status updates about your plant.

If you have a Mac and want to make your plant really talk, you can download a script from the Head First Labs website that will read out the stream of serial data: www.headfirstlabs.com/books/hfc