

Garduino Upgrade, Now with more Twitter!

by [natantus](#) on November 14, 2009

Table of Contents

License: Attribution Share Alike (by-sa)	2
Intro: Garduino Upgrade, Now with more Twitter!	2
step 1: Gather your materials	2
step 2: Build your Garduino	3
step 3: Upgrade #1: Remote Sensors	3
File Downloads	4
step 4: Update #2: Relay Boxes	5
step 5: Upgrade #3: New Software	6
File Downloads	7
step 6: Upgrade #4: Wireless Control	7
step 7: Update #5: Twitter your Garden	8
step 8: Useful project notes	9
Related Instructables	10
Advertisements	10

Intro: Garduino Upgrade, Now with more Twitter!

A couple months ago I came across two great instructables. The first was the [Garduino](#), an arduino controlled garden to help you grow plants at home. The second was the [Tweet-a-Watt](#), a project that teaches you how to monitor your home power usage using Xbees and Twitter. I read about both these projects here at Instructables and in [Make Magazine, Vol 18](#).

I thought it would be great to combine both these projects and build myself an indoor garden that I could monitor from work via Twitter. Thus began an adventure in gardening and electronics that taught me a lot and took me much longer than perhaps it should have. Fortunately for you I'm going to write down all the steps so you can get started right away. Maybe you'll follow up with this project and upgrade your garden or use this as a guide to start on a similar project. Either way, I hope you'll let me know what you get up to.

If you're ready then head to the next step and begin the process!

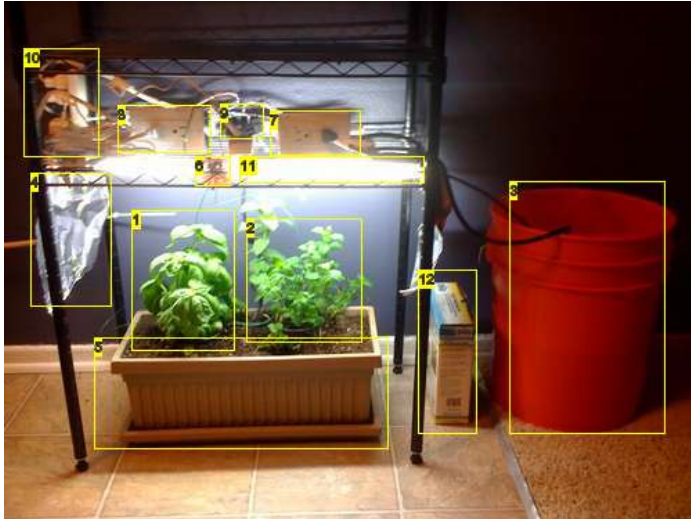


Image Notes

1. Basil plant good for soups
2. Mint good for ... what isn't it good for?
3. Water bucket with pump
4. Foil keeps the light inside
5. Planter with lots of yummy soil
6. SEnsor board
7. Pump relay box
8. Light relay box
9. Arduino running it all
10. Power to all the good stuff here
11. Light for the plant from the "Hydroponics Store"
12. Starter kit for garden drip system

Image Notes

1. Hadn't thought of the foil idea yet
2. Plants! They grow!
3. All the good upgrades for this project
4. Water ... note that it's not on my kitchen floor yet
5. Box of goodies

step 1: Gather your materials

This project is really two projects in one. It is both a gardening project and a wireless project. But, as is true with any project, getting your materials will probably be the hardest part. This is because building can move along pretty fast once you have everything, but it's finding out you're missing the one crucial piece that always holds things up. Below is a list of web pages from my wiki with a list of all the parts you'll need. Go ahead and take a look, I've listed them with as much detail as I could and if I've left something out just let me know so I can add it to the list.

Various Parts Lists:

- [Garduino Parts List](#)
- [Relay Box Parts List](#)
- [Arduino Xbee Adapter Parts List](#)

I highly recommend you buy a couple extra parts here and there. You'll find, as I did, that they come in handy when you make a mistake. For the majority of the parts nothing is particularly expensive so buying a few extra things won't hurt.

step 2: Build your Garduino

The Garduino Instructable is probably your best resource for starting this project. It has tons of great insights that will help you get started with your own arduino controlled garden. He has also put together a website with more pictures and a kit that you can get, but I chose to build this project with my own parts.

When you get started you'll want to get some plants and a place to plant them. I recommend a local gardening store where possible. You'll pick up some healthy plants there and probably a few good tips on how to care for them. I picked up basil and mint, a couple plants I'd find useful to have for cooking. I also got an 8" x 18" planter with a matching dish to catch water. You can certainly go out and use tubs or custom pots, just choose what's right for you and your plants. Also, pick something that will look nice around your house if that's important to you.

The first thing that is different about my project is that I was building an indoor garden. My apartment doesn't have a lot of light from any of the windows, nor does any good light fall on the patio. This changed a lot of the parameters for my garden. For one, I didn't need a light or temperature sensor, something I discovered after the fact, as my light will be constant from the fluorescent bulb and my temperature inside the apartment is pretty stable.

I also learned that whether you choose to keep your plants inside or outside you should be aware of bugs. My first plants died because of spider mites, something I hadn't even thought of when starting this project. I found the solution was actually pretty easy, involving a spray bottle, a tiny bit of soap and water. It's a much cleaner solution, especially if you plan to eat your plants, and doesn't hurt the environment either.



Image Notes

1. Basil plant good for soups
2. Mint good for ... what isn't it good for?
3. Water bucket with pump
4. Foil keeps the light inside
5. Planter with lots of yummy soil
6. SEnsor board
7. Pump relay box
8. Light relay box
9. Arduino running it all
10. Power to all the good stuff here
11. Light for the plant from the "Hydroponics Store"
12. Starter kit for garden drip system



Image Notes

1. Defense against spider mites!

step 3: Upgrade #1: Remote Sensors

One of my early upgrades to this project was to take the sensors off the arduino and put them closer to the plants. I wanted to do this primarily to keep the arduino as far away from the water and water pump as possible. The nice thing about this is that you can put your arduino somewhere accessible and safe and keep the sensors you want closer to the plants, where they are needed. Is this a good idea? Maybe, but I was happy to do it.

My sensor was designed to have the temperature, light and moisture sensors onboard. Also, I included 2 red LEDs so that I could indicate if the plants were being watered or if the temperature went out of an acceptable range. Of course this was before I had decided to keep the plants indoors, so in retrospect I'd probably ditch the temp and light sensor and leave the LED on the arduino. Better yet I could have just used the onboard LED connected to pin 13 on the arduino. Then all I'd have is two wires leaving the arduino to act as the moisture sensor for the plant. Regardless, this was a fun part of the project and I learned a lot.

You can download the schematic and board layout files I created. I decided to build the board by hand instead of having it manufactured for me primarily due to cost.



Image Notes

1. All the sesors and lights that tell my arduino what the plant is up to
2. Down here are the nails that act as the moisture sensor
3. Nails act as the moisture sensors for the plant
4. The light was thankfully turned off for this photo

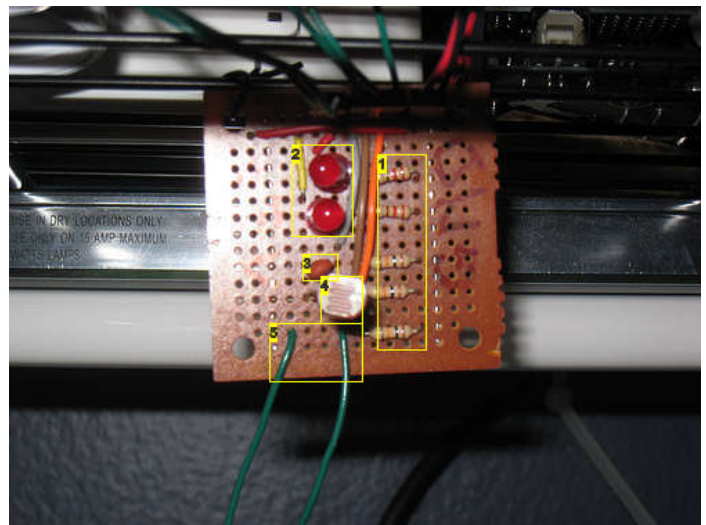
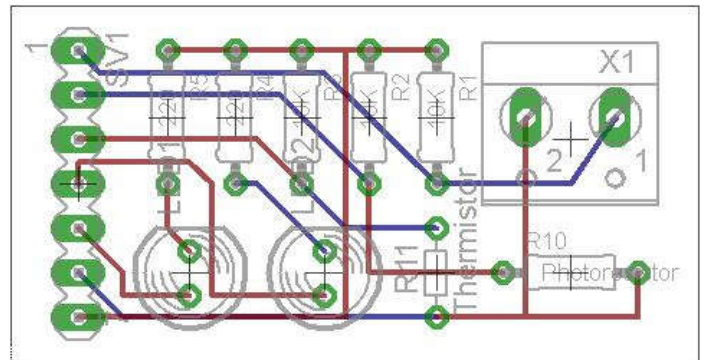
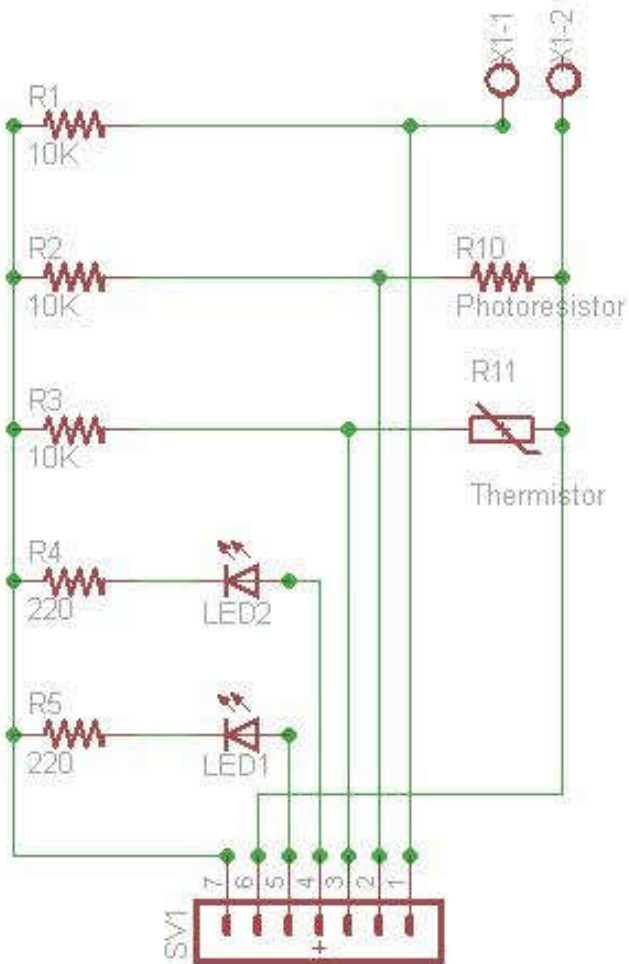


Image Notes

1. A ton of resistors
2. LEDs tell me about the temperature and the pump
3. Temperature sensor
4. light sensor, pretty useless right here
5. Moisture Sensor wires



File Downloads



garden_sensor.brd (10 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'garden_sensor.brd']

<http://www.instructables.com/id/Garduino-Upgrade-Now-with-more-Twitter/>



garden_sensor.sch (69 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'garden_sensor.sch']

step 4: Update #2: Relay Boxes

The most important upgrade I think I made was to build real relay boxes to control my power to the light and the water pump. With water so close to the electricity I figured a safer approach was needed than the one used in the original Garduino instructable. Fortunately the author pointed out a great resource at SparkFun that I used to do the project.

Finding the whole thing incredibly useful I wrote up another instructable on how I built my relay boxes and made it more generic for other projects where you might want an arduino to control power. Check out the [Arduino Controlled Relay Box Instructable](#) when you get a chance.

At this point I decided to add on an arduino protoboard to my project so that I could have access to more power lines without having to solder together a new breadboard. You may choose to do something different, but this made things significantly easier, especially if I decide to do more modification later.

If you don't do anything else with your Garduino definitely do this upgrade. It's definitely the most important.

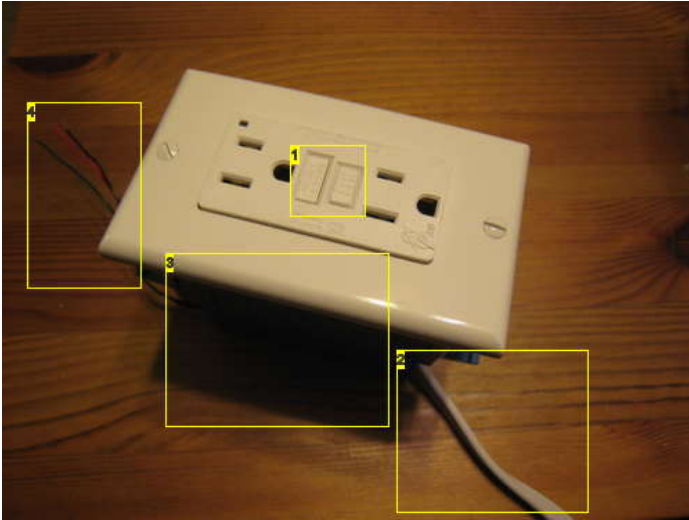


Image Notes

1. Test and Reset buttons
2. Extension cord wires
3. Nail mount housing hides in the shadows!
4. Control wires

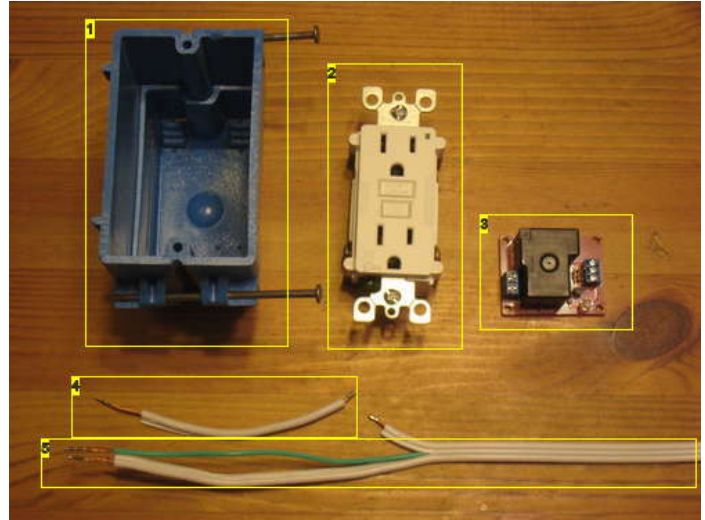


Image Notes

1. Nail mount housing
2. GFCI Outlet
3. Relay circuit completed
4. Hot wire disconnected from the extension cord
5. Extension cord

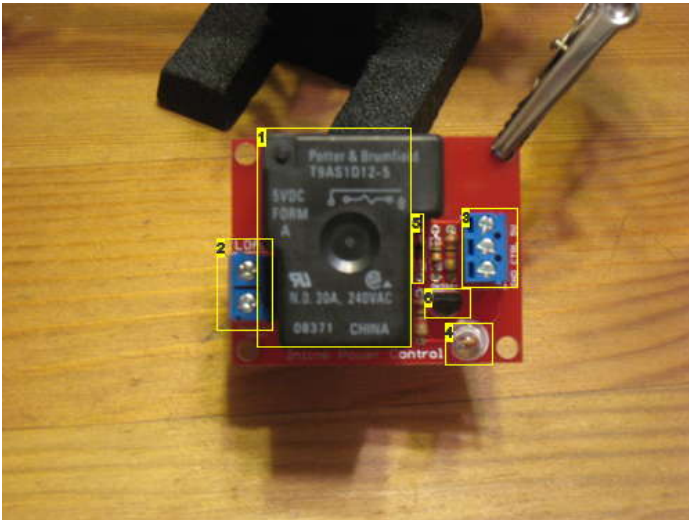


Image Notes

1. Relay - 240/120V, 30A
2. Load connectors
3. Control connectors to Arduino
4. LED tells you its on or off
5. Diode to protect microcontroller
6. Transistor does all the hard work for no pay!

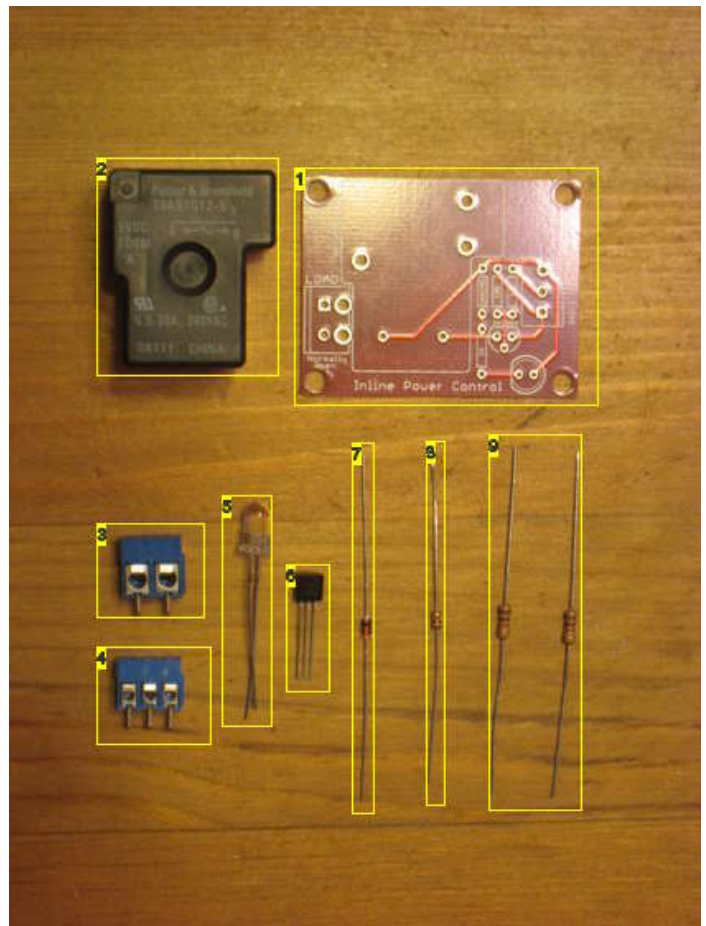
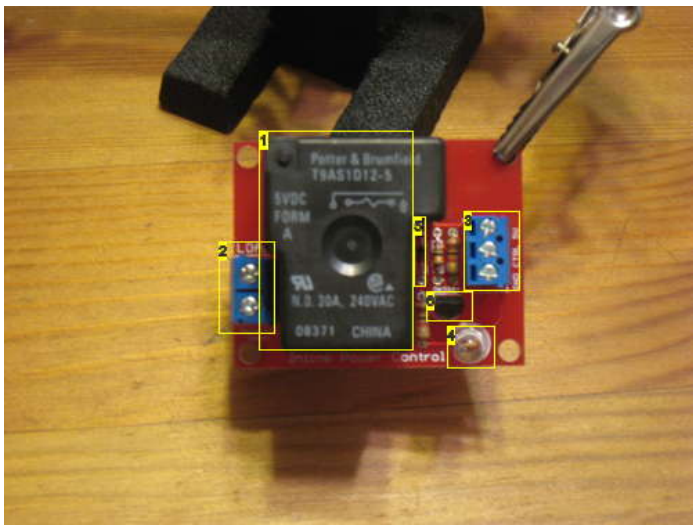


Image Notes

1. Breadboard for circuit
2. Beefy Relay
3. Load connector
4. Control connector
5. Bright red LED
6. Transistor
7. Diode
8. Resistor
9. Resistors, same type

step 5: Upgrade #3: New Software

Now it's time for the software upgrade. With an indoor garden I no longer had the need to have the entire project be automated within the arduino. The original arduino kept an internal clock and counted the amount of daylight from the light sensor against the number of seconds in the day. I found this to be more frustrating than it was worth once I moved inside and stopped using the light sensor. This was primarily because I found that the light cycle revolved around when I power cycled the arduino, having no way to set the internal clock myself.

In fact I thought I could do a better job regulating the time from my laptop. Not only would the light only be on at night and in the morning when I wouldn't care, but I could reset the arduino power at any time and it wouldn't change this schedule. The added advantage to this method would be that I'd also be able to record the sensor readings from the garden to look at them later.

To get started you have to know that the arduino can talk over a serial connection to your laptop. You probably already knew this, but what you may not know is that you can use the Python programming language to read and talk to the arduino. This is great because it opens up all sorts of tools for you to use when interacting with your arduino.

For this you'll need to download and install the following:

- Arduino IDE (0017 or later)
- Python (preferably 2.6.X or later)
- PySerial Library

At this point I also want to direct you to the [Arduino Controlled Servo Robot project](#) by Oomlout. My methods roughly follow what I learned there. The basic premise is that you'll be sending a command from the laptop to the arduino every 15 seconds. The arduino will decode this message, check that it is a correct message, and then the arduino will use the commands to manage the garden. If no message is received then no new actions will be applied to the garden.

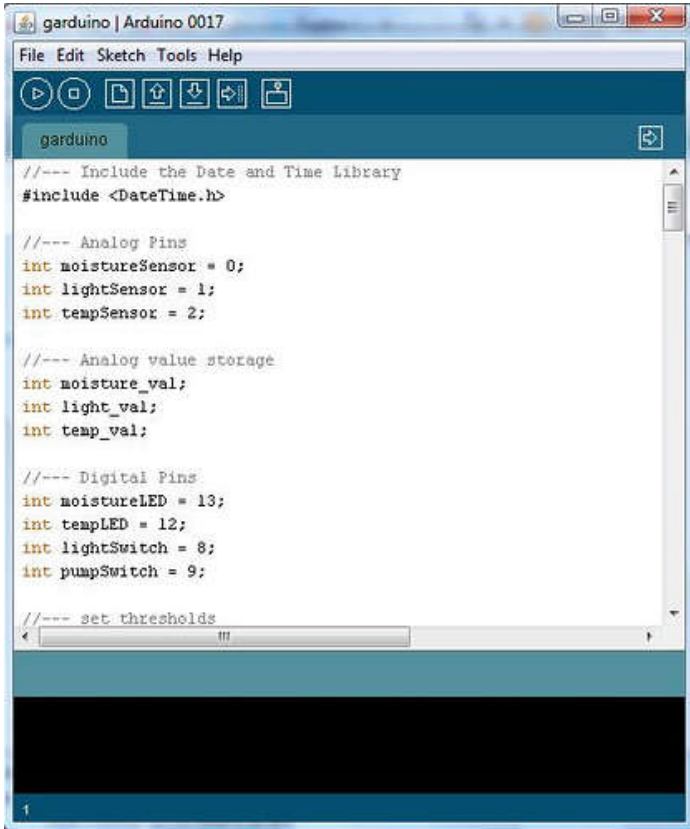
I choose to send a command that looks like this: "+++lw". The arduino can tell if the message is correct by reading that I have included the "+++". Then I pass the letters 'l' and 'w'. If the 'l' is lower case then it tells the arduino to turn off the light. If the 'l' is an uppercase 'L' then the light will come on. Easy, right?

The 'w' is trickier. I have included some safety protocols in my software. The arduino will only turn on the pump for a maximum of 5 times for 5 seconds each time. You may wonder how a plant will live on only 25 seconds of watering. Well I reset the counter every 4 hours. This way the plant will continue to get water and my kitchen will not continue to get flooded. The 'w' as a lower case tells the arduino nothing useful, but if I change the 'w' to an uppercase 'W' then the pump counter will reset to zero

<http://www.instructables.com/id/Garduino-Upgrade-Now-with-more-Twitter/>

and the arduino can again water my plant if it's necessary. Make sense? You may want to something slightly different, but this worked for me.

Instead of including the code in the text here I've decided to include files. You should be able to open these with your Arduino IDE for the arduino code and with IDLE or a text editor for the Python code.



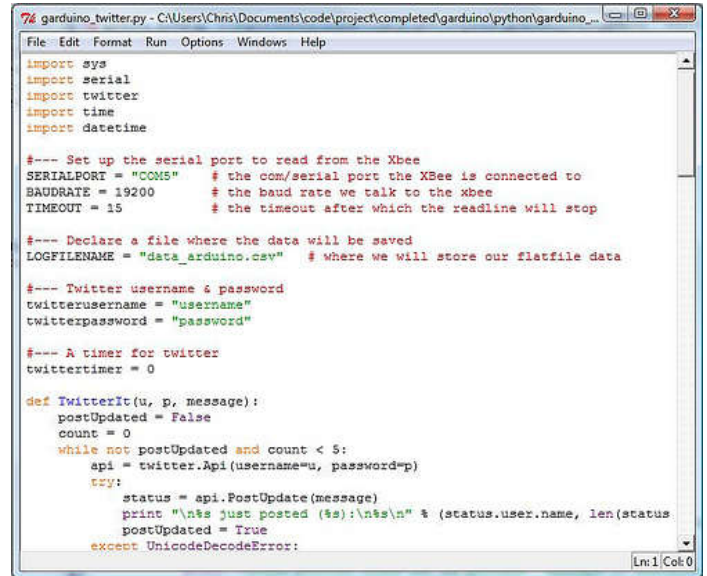
```
File Edit Sketch Tools Help
garduino
//--- Include the Date and Time Library
#include <DateTime.h>

//--- Analog Pins
int moistureSensor = 0;
int lightSensor = 1;
int tempSensor = 2;

//--- Analog value storage
int moisture_val;
int light_val;
int temp_val;

//--- Digital Pins
int moistureLED = 13;
int tempLED = 12;
int lightSwitch = 8;
int pumpSwitch = 9;

//--- set thresholds
```



```
File Edit Format Run Options Windows Help
garduino_twitter.py - CAUsers\Chris\Documents\code\project\completed\garduino\python\garduino_...
import sys
import serial
import twitter
import time
import datetime

#--- Set up the serial port to read from the Xbee
SERIALPORT = "COM5" # the com/serial port the Xbee is connected to
BAUDRATE = 19200 # the baud rate we talk to the xbee
TIMEOUT = 15 # the timeout after which the readline will stop

#--- Declare a file where the data will be saved
LOGFILENAME = "data_arduino.csv" # where we will store our flatfile data


#--- Twitter username & password
twitterusername = "username"
twitterpassword = "password"


#--- A timer for twitter
twittertimer = 0

def TwitterIt(u, p, message):
    postUpdated = False
    count = 0
    while not postUpdated and count < 5:
        api = twitter.Api(username=u, password=p)
        try:
            status = api.PostUpdate(message)
            print "\n%s just posted (%s):\n%s\n" % (status.user.name, len(status.postUpdated = True
        except UnicodeDecodeError:
```

File Downloads

 [garduino.pde](#) (4 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'garduino.pde']

 [garduino_twitter.py](#) (5 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'garduino_twitter.py']

 [twitter.py](#) (45 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'twitter.py']

step 6: Upgrade #4: Wireless Control

Now comes the exciting part! It's probably not likely that you can leave your computer connected via USB to your arduino all day long in your kitchen. If you can that's great, but I need my laptop in my office where I find it useful. This is where my experience with the Tweet-a-Watt came in handy.

Early on I decided to use Xbee wireless devices to talk between my computer and the computer. Now I apparently did things the hard way because I thought you'd need to send the data through the Xbee controller from the arduino and then decode it on the other end with the Xbee Python library. This turns out to be completely untrue. You certainly can do it this way, but it's way harder than simply using the Xbee modules as a direct serial link to your computer.

It may have taken me hours, but this step shouldn't take you very long once you've set up your Xbees. Essentially you remove the USB cable between your computer and the arduino, hook up your Xbee modules (one to the computer via an FTDI cable and one to the arduino via tx/rx lines), and then continue as though you were still using the USB cable. Don't believe me? Try it. Maybe this was obvious to other people but it was pretty exciting when I figured it out.

Now I should tell you there are several steps to doing this whole wireless step. You need to first purchase and assemble the Xbee modules. Then you'll want to program each of the Xbee chips and finally connect them up. The best resource I found for setting up my modules was actually a tutorial about [Wirelessly Programming your Arduino](#).

I decided to set up my arduino to be wirelessly programmed so that I could easily update the code from my laptop when I needed to upgrade my arduino. This was an added bonus to being able to wirelessly talk to my arduino via serial and it just seemed to sweeten the whole project. I also found this made some of the debugging easier since I needed to do a lot of writing and rewriting to the arduino.

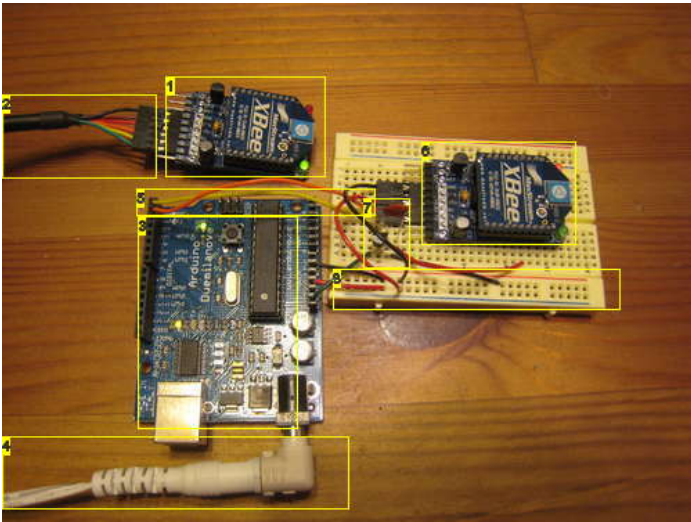


Image Notes

1. Xbee module for computer
2. FTDI cable to computer
3. Arduino Duemilanove
4. 9V power from wall
5. TX/RX lines to Xbee module
6. Xbee module for arduino
7. Electronics to program arduino remotely and hit the reset
8. Power!

step 7: Update #5: Twitter your Garden

Now you've got your garden all hooked up to your computer and talking wirelessly to the computer. You may even be recording your data to a file to look at it later. What you may not be able to do yet is monitor your plants while you're at work. To do this I employed Twitter, much like the Tweet-a-Watt did. I signed up for an account that I could use for testing, @chrisgilmertest. Then I hooked my code into the Python Twitter API, entered my user name and password, and voila! I was done.

I set up my twitter to send back the readings for the different sensors and the current time. I know you can track the time you posted, but I wanted the time from my computer. I also decided to have it tweet every half hour. When I originally was programming this I had it tweet every 5 minutes, which was good when I wanted to ensure it was actually working, but got annoying to follow pretty quick. I may actually have it tweet every hour or couple hours in the future. You get to decide what you want for yourself and go from there.

I have asked friends to help me come up with some silly messages for my garden to tweet, based on the sensor readings, so maybe I'll get some snarky comments up there soon.

You may have noticed that this code was already provided to you earlier in this Instructable. My hope was that you got excited and already started to play with it. If not, what are you waiting for?

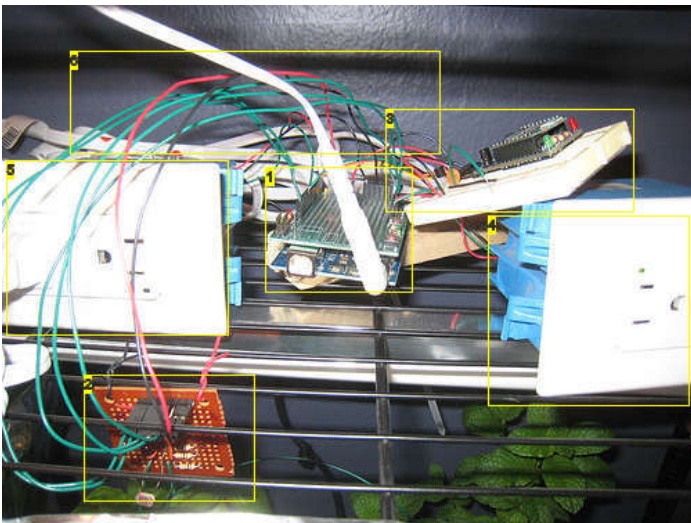


Image Notes

1. Arduino and protoshield
2. Sensor board
3. Xbee module set up temporarily on a protoboard
4. Pump outlet
5. Light outlet
6. Too many wires!

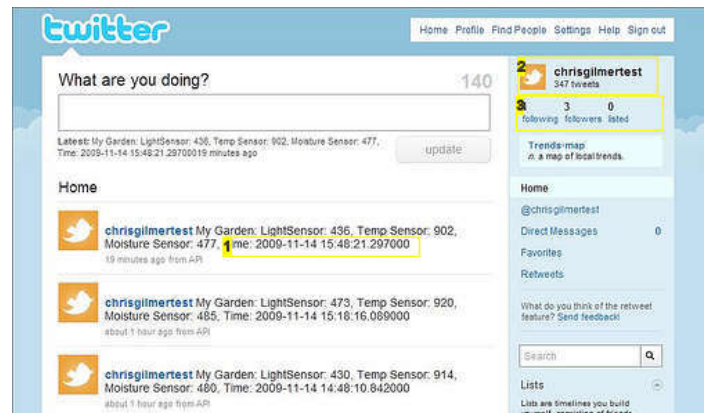


Image Notes

1. I like to time tag all of these
2. I think my garden has not put out more tweets than me
3. People are already following my garden!

step 8: Useful project notes

I learned a bunch of useful things while doing this project and I wanted to make sure I captured them here for you. Hopefully I've written them all down because they are mighty handy:

Plants:

When it comes to plants remember that they are indeed alive. They are also tasty, not just to you but often to other things as well. My first plants were eaten by cats and then destroyed by spider mites. I never would have thought about this during the planning stage of the project and it definitely cost me a few extra bucks to replace them.

Lighting:

Plants need light and it can be tough finding the right set of lights. There is plenty of information out there about lights for growing plants but it turns out to be incredibly difficult to find these lights at a local hardware store. It took me a while but after asking around I was directed to a Hydroponics Store. You know the kind (wink, wink), and they hooked me up in no time at all. Pun partially intended. The people were great at the store and definitely reminded me of why I like to buy local when possible.

Water:

You've got to be really careful when you get started with automatically watering your plants. The soil resistivity will change dramatically as you first start to water it, meaning your calibration of the sensors will be difficult. I found this out the hard way when my pump decided to water not only my plant but my entire kitchen. It's better to program in some safety catches in the software to prevent this. My plants will only water for up to 30 seconds at four different times in the day and no more. It may be different for you, but be warned.

Wireless Programming:

You may have set up your arduino to be wirelessly programmed. I did and I found the only annoying thing is that I'd reset my arduino when I reset my computer. Turns out I was flipping the reset line on the arduino via the Xbee. The easiest solution to fix this was to disconnect the reset line whenever I didn't want to reprogram. I could still leave the other electronics in place, but removing the reset line fixed the problem without having to disassemble anything else.

Final Comments:

I had a lot of fun building this project. I learned a lot along the way and I hope to have some delicious plants for eating after all this work. I hope you found this project interesting and useful for upgrading your own Garduino. If you do follow any of the upgrades in this project let me know and tell me how they worked out for you.

You can always ask for more information or check out my website for more information. I keep a [blog](#), a [project wiki](#), and an [svn repository](#) with the latest code. I also have a set of [photos](#) you can look at. Hopefully it's all up to date and full of resources to help you out. Good luck with your project!



Image Notes

1. Basil plant good for soups
2. Mint good for ... what isn't it good for?
3. Water bucket with pump
4. Foil keeps the light inside
5. Planter with lots of yummy soil
6. SEnsor board
7. Pump relay box
8. Light relay box
9. Arduino running it all
10. Power to all the good stuff here
11. Light for the plant from the "Hydroponics Store"
12. Starter kit for garden drip system

Related Instructables



Tweet-a-watt - How to make a twittering power meter... by adafruit



Watts-your-consumption? - Wireless Power Meter by mcloke74



XBee adapter by adafruit



The Twittering Office Chair by randofa



Arduino gift guide (slideshow) by pt



Properduino - When an Engineer Gardens - Part 1 - Le Propergator Watering System by grahamslawson



Garduino: Gardening + Arduino by liseman



Tweet-A-Temp by Z0t