Automatic Plant Watering

📝 Project Overview

This was my introduction to Arduino projects and really wiring/electrical work in general outside of typical PC plug and play components. First was a lot of research, probably a week or so of YouTube videos, Reddit threads and google searches.

I started off with the basics of working with electricity, the difference between AC and DC, voltage, current all that incredibly mind numbing Greek alphabet nonsense (in a not racist way). Once I had a basic understanding of these I moved onto the common parts in these types of small home projects, resistors, transistors, breakout modules and GPIO headers, that kind of stuff, this stuff was a lot more interesting to look into and led down many rabbit holes.

Once I started to feel like I could look at a prototype and identify a handful of the parts and roughly piece together the procedural logic behind the project, I started watching more 'full project walkthrough' type videos to try and get a better understanding of how these concepts holistically connect to form a finished product.

I kept most videos in the vein of basic automation and plant watering using microcontrollers to get a good understanding of the general flow that I'd be applying to my project.

And that was about it, I started researching local and international hobbyist stores as well as mainstream suppliers for components, looking up reviews and recommendations on good quality tools and ultimately have gone with Core Electronics. While they are based in (I don't know but somewhere in Australia that's not my state) they had great reviews, very reasonable prices and a really nice website (for whatever reason so many of these stores seem to have really clunky sites). You can feel the effort they put it in, so I tend to order in semi bulk from them when I need new things, haven't had an issue yet.

✅ Steps

1. I happened to get the microcontrollers second-hand off marketplace ($10 AUD) and had them before the rest of my parts arrived. So I started off with plugging it into my laptop and getting the Arduino IDE software installed. Using a Linux a laptop there was a slight problem getting started (I downloaded an incompatible version bc that's what happens when you don't read before you download). So, after downloading the correct version it was as simple as installing and using any other software.

2. With the IDE installed and the board connected, I looked to select and configure the few settings I needed to. Since I was using an international CH340 board for testing I had to install some dependencies to get my board recognised, then it was as simple as selecting the model and the designated COM port.
3. Once the parts arrived, I got to connecting the controller board to the breadboard and doing some basic tests to make sure power was being passed through the breadboard. During prototyping I used the laptop connection for power supply to keep testing simple and consistent.
4. Once I'd played around a little with some jumper wires and the multimeter and confirmed that connections were secure and power was being distrusted how I'd expected, I connected the moisture sensor. Very straight forward installation, the sensor had three labelled wires, power, ground and an analog output which receives the moisture sensors data. Connecting these to the rows connected to the board, I got the sensor connected.
5. Next up, testing that the sensor was working. This was were I actually had my first real problem (laughable rookie mistake), but that will come soon. To test it was working I wanted to set up a simple program to output or echo the readings to the IDE console (technically serial monitor). Simple as it is I shamefully used ChatGPT to spin up a quick test script (reoccurring theme - don't judge me I'm not a software dev) which took a couple minutes to debug and modify. Compiled and uploaded and script and got suspiciously consistent readings. Whether the sensor was in moist, bone dry soil, even a cup of water, I was getting ~350 no matter what. After more research than I'd care to admit, I realised I was missing the analog (A) identifier before declaring my controller analog input pin. Controller boards will vary however will typically have analog and digital pins, by default digital pins do not need an identifier (i.e. D), this is assumed by the IDE. Analog pins however do need an analog identifier, updating my input variable from <var = 0> to <var = A0> gave me the readings that I was expecting.
6. Next up was the pump which was a little more involved and had it's own problems attached. The pump itself was very simple, soldered red (vin) and black (gnd) wires, the pump remains fully submerged and when triggered it pushes water through the piping to the plant. First problem was, it needed a relay module to send an electrical current when the readings reached a certain level to trigger the water flow. Second problem was, the fucking relay was Chinese with less than helpful google translations.
7. Once the vin from the pump was wired to the relay and the pump gnd was connected to the breadboard rails, I got to testing and immediately had a problem, the pump wouldn't turn off... It took me a few minutes to work out the next steps but I figured the best way to find out the cause was trial and error. I could assume that the connections were fine, I had to figure out at what point in the circuit the pump was being passed a current and what was triggering the pump. I assumed it was the code, reviewed it and ran it through validators and debuggers but no luck. Fast forward a couple hours of testing different configurations I'd narrowed it down to the relay connections problem was I still had no hope of reading what

each port was for. Logically, I used a similar English relay module as comparison and connected as I would to that, still no luck. While I don't know for sure this is the reason, I'm quite sure the Chinese ports are in reverse order, meaning using the opposite ports to the English comparison had my pump functioning perfectly. Long story short, relay module had 3 output ports, one was definitely for a power connection to the controller, the other effectively determines whether the connection is constant or intermittent. Initially, the circuit was configured to leave the connection between the relay and pump open making the pump run continuously, switching to the closed port allowed for precise triggering.

8. Next problem with the pump and the relay, the factory wires were pathetically small and frail. I now understand this is because you're expected to put your own connections on the end which actually had me stumped at first and sent me back to google. Fast forward and I'd settled on dupont connectors, they were cheap to buy a variety pack good for what I need and future use and connect easily both with the breadboard and the final Arduino Uno R3 which had built-in female GPIO inputs. Along with the dupont connectors, I got an average wire crimper, and added some basic connections on the end of the wires helping to secure connections much easier. Initially I was pretty lazy and spent time looking for a bulk pack of various connection forms to avoid doing it myself... I was wrong, learn it, it's easy and useful, learn it.
9. With everything connected, next up was writing the final program that could incorporate all parts and effectively keep a plant healthy. Again, using ChatGPT to spin up my program, I was much better with debugging at this point and it didn't take long to get the few things changed to how I wanted. Surprisingly, from here was pretty straight forward. Program ran fine which will be attached below. It's worth noting that the current program can absolutely be optimised for improved efficiency and accuracy.
    - Currently, the moisture sensor takes constant readings with a 1 second delay effectively meaning one reading per second. While so far I haven't experienced any inaccuracies which cause for immediate change, I have been considering changing the delay to something like ~5 seconds and taking the average over that delay period (which would be average of reading per x seconds) to prevent the pump being momentarily triggered over longer periods of time leading to over watering.
10. With the program functioning correctly and all connections secured, I was ready to transfer to a permanent board. I will note that initially, I planned to use a micro controller board without GPIO headers attached and splicing the jumper wires and soldering directly to the controller board. The intention was to keep the final product as compact as possible however, with minimal soldering experience and no practical casing solution for a microcontroller board once I had soldered all connections, I opted to use the Uno R3 board. Reason was simple, there was $10 acrylic case made for the R3 about 10 minutes from my house, lazy yet problem solved.
11. Moving the connections over to the R3 was much easier than expected. The R3 board has female GPIO inputs making it really easy to make sure you're a) connected to the right port

(yes, believe it or not, if you haven't used breadboards before it's very easy to mistakenly connect to the above/below pin) and b) connection is secure (it either fits nicely in the port or it doesn't). With the wiring planned out, I put the board inside the case, attached the wires and threw a couple small zip ties on to make it a little easier to look at. If you're familiar with the basic kind of acrylic case you might be wondering how I got a PSU and the relay attached since there's no native housing, -remember this is a safe space... - I electrical taped it on the top. If you throw aesthetics out the window it is very practically viable solution.

12. Ironically, the homemade explosive esqe battery pack was redundant since I miscalculated the power usage and fried the battery in two days. I actually suspect that the sensor was running overnight outside of soil and humidity/env factors triggered the pump for long periods overnight and drained the battery (I only anticipated the pump being used for ~5-10 seconds per day maximum) however did not care to actually find out and instead opted to be lazy again and use the native power jack with a USB adapter.

🛠 Tools

- Digikey CH340 Microcontroller - Prototype Testing Board
- Arduino UNO Controller Board - Production Controller Board
- Solderless Breadboard - 830 Tie Point
- Diagonal Cutters
- Tubing for Submersible Pumps - PVC 6mm ID - 1 Meter Long
- Capacitive Soil Moisture Sensor (v2.0)
- 5V Single Channel Relay Module 10A
- Needle Nose Pliers
- Jumper Wires (Various)
- Dupont Connectors
- Wire Crimper
- Arduino IDE
- Digital Multimeter
- 9V Battery
- 9V Battery Clip (3.5mm Power Jack Adapter)
- USB Power/Data Cable

🔗 References

- Core Electronics - DIY Electronics Hobbyist Store