2021-08-16
Byproduct#9084

Automatic plant watering with Raspberry Pi and Python

For up to 10 plants

Introduction

What does this thing do?

- It checks on your plants at regular intervals, for example once per day. If the sensors detect that the soil is dry, the plant gets some water.
- The software is configured using a text file, in which you specify the RasPi GPIO pins for each sensor and pump, and the amount of water (pumping time) given to each plant.
- It writes a daily log that you can read while nodding and humming approvingly.
- It checks for errors: for example, if attempted water pumping did not result in moist soil, operation is stopped until you go and see what's going on.
- (Optional) discord bot sends you a message if something needs to be done.
- (Optional) debug mode prints out operation in detail (each pump and sensor etc) for easy troubleshooting.
- …and more as I code features – or as you code features! It's open source.

Is this in any way practical?

Probably not. :D

It's mainly for learning and just geeking out with plants for fun.

It might save you some time (from manual watering) over a long period, especially if you have a lot of plants, but don't count on it.

It might also allow you to be away from home for a longer time without your plants dying. Like a watering bulb, but geekier, with a lot more capacity and drastically worse reliability!

> What's probably going to happen is that everything will work smoothly for several months with zero issues, but when you leave for your holiday, something will break the next day. You can monitor the system remotely though!

Is this going to be expensive?

Depends on the number of plants. Roughly 50€ for the RasPi, around 100€ for the other parts. Gardening stuff not included.

How many tomatoes do I need to grow to cover the costs?

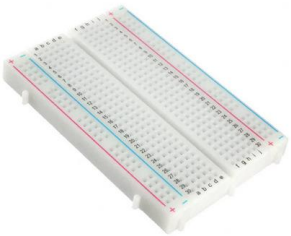
I could not be reached for comment at this time.

Things you'll need

Raspberry Pi and basic knowledge on how it works. Optimally one with 40 pins and wifi, so models 3 and up.

Lots of jumper cables (male-to-female, M-M and F-F).

Moisture sensors that look like this. Widely available in the context of RasPi or Arduino. Or just search for moisture sensors. Probably a good idea to get one or two extra ones just in case.



A breadboard or two. The 3/5V pumps can be powered by the RasPi's power pin and the breadboards are used for that. If you prefer other pumps, you'll need an external power source for them.

Submersible pumps, 3/5 volts. Same deal as with the sensors.

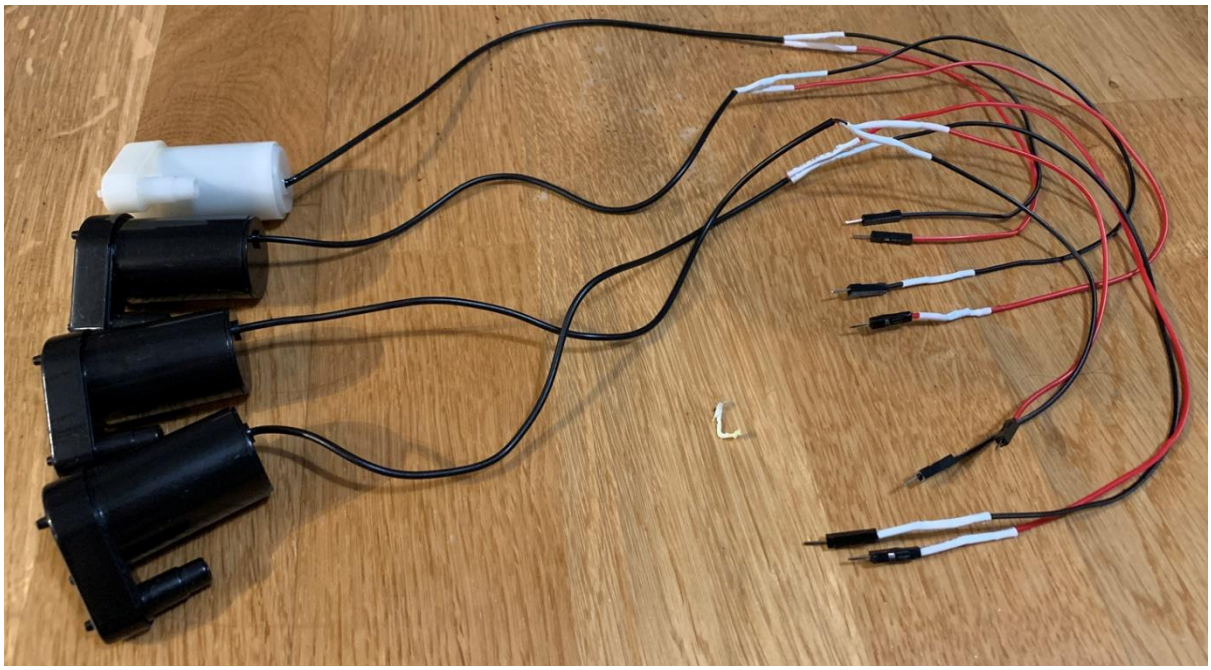
5V Relay modules. One module with one channel, and one module with as many channels as you have plants.

Some water tube. The pump you found will probably have a link to the right kind of tube. Mine is 8.20mm outside and 5.54mm inside diameter.

Some friendly plants.

Building

Extending pump wires



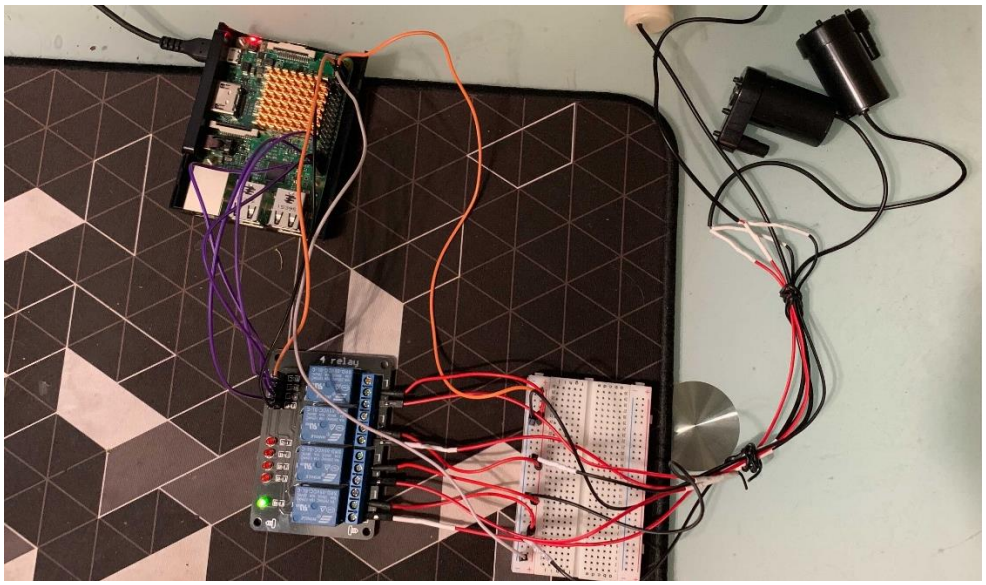
All gardening projects start with some soldering, right?

The pumps came with short wires, so I extended them with jumper cables. That way they're easier to plug in and the RasPi setup can stay farther away from water. After all, you'll want to irrigate only your plants and not your electrical components.

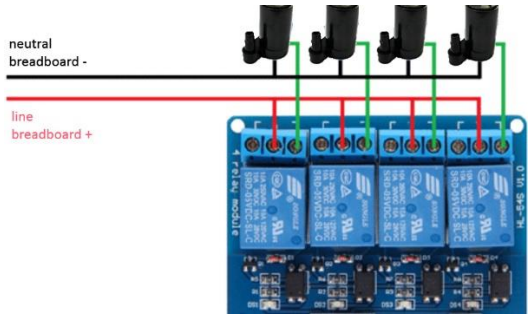
If you're not into soldering, you can twist the wire strands together and cover them with electrical (insulating) tape.

Building

Connecting pumps



Here's how the pumps are connected. Connect a RasPi 5V power pin and ground to breadboard + and -, and the breadboard becomes your power source. It may look like spaghetti but isn't complicated. The image below (kinda stolen from here) may help make sense of it.




```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(21,GPIO.OUT)
GPIO.output(21,GPIO.LOW)
time.sleep(5)
GPIO.output(21,GPIO.HIGH)
GPIO.cleanup()
```

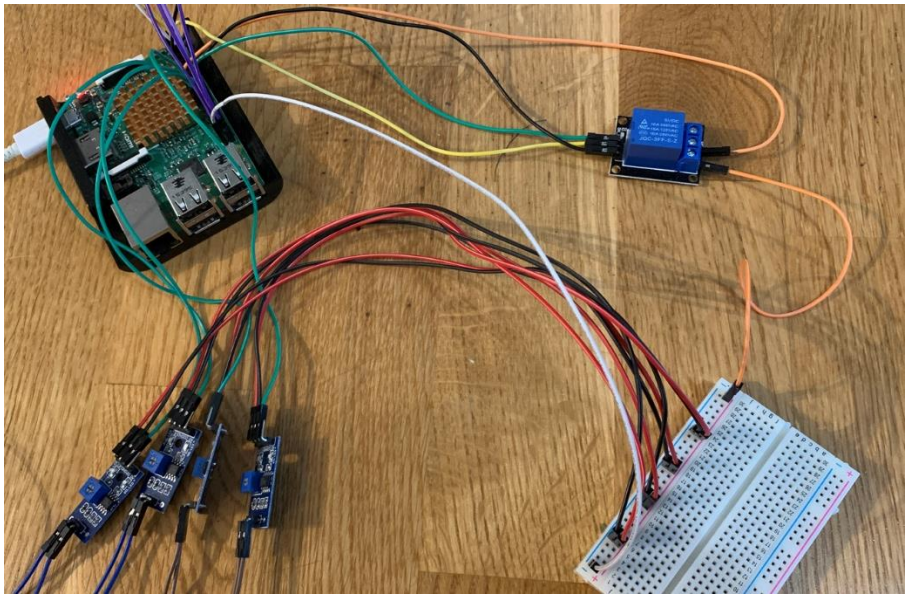
RasPi 5V power pin (not GPIO pin!)	-> breadboard +
RasPi ground	-> breadboard -
RasPi 3V power pin	-> relay power
RasPi ground	-> relay ground
Raspi GPIO pins	-> relay in1… in4
pumps +	-> relay NO ports
pumps -	-> breadboard -
relay COM ports	-> breadboard +

In the breadboard, you can use any slot along the same + or - rail.

The pumps will now run when you set the GPIO of the corresponding relay to output GPIO.LOW, and will stop when you set it to GPIO.HIGH. A simple test program **`pumptest_simple.py`** is included. You will have to manually change the GPIO pin number (21) in the code. Put the pumps under water when you test them.

Building

Connecting sensor chips



Here's how to connect the moisture sensor chips. I assumed I'll just simply connect them to power and GPIO, but I've been told the sensors will corrode quickly if they're always on. A moisture sensor that gets destroyed by moisture, what a concept! Another relay is therefore needed, but this time just one channel is enough.

A RasPi power pin provides power to the breadboard, the sensors get their power from the breadboard, and the breadboard is turned on and off with a relay.

So: turn on relay -> get all four measurements -> turn off relay. The sensors will live a longer and more meaningful life and will be grateful to you.

RasPi 5v power pin	-> relay power
RasPi ground	-> relay ground
RasPi GPIO pin	-> relay in
RasPi 3V power pin	-> relay COM port
Relay NO port	-> breadboard +
RasPi ground	-> breadboard -
sensor chips +	-> breadboard +
sensor chips -	-> breadboard -
RasPi GPIO pins	-> sensor chips digital out

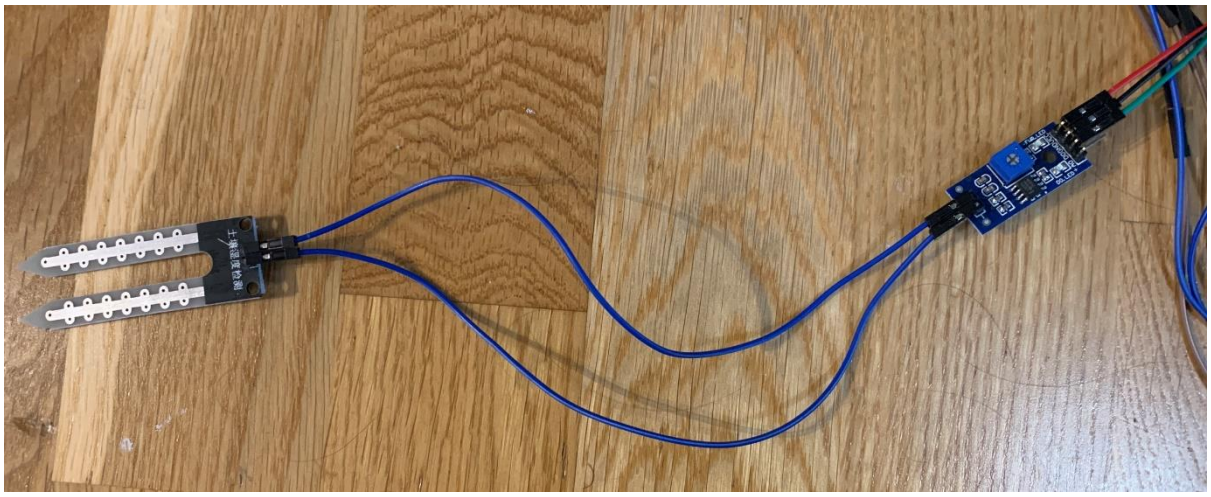
You might wonder why one relay/board setup is 5V and the other is 3V – well, I don't know! These relays are all sold as 5V relays, but one worked only with 3V and one with 5V. 🤷 So if your setup doesn't work, try switching between 3V and 5V for relay power.

The pumps and sensors should work with either 3V or 5V. The pumps will probably be more powerful with 5V though.

You could use just one breadboard, because they usually have two separate power rails. I used two separate boards to keep the cable spaghetti under control.

Building

Connecting sensors



Connect the sensor chips to the sensors with the two (probably unlabeled) pins. It doesn't matter which way they are connected.

Some of your plants may be farther away than others. You can easily extend these sensor cables with male-to-female jumper cables.

This project uses the digital output from the sensor chip, which gives a reading of 0 if the soil is moist, and 1 if the soil is dry. The threshold for this can be adjusted with the small screw on the chip, but the default setting seemed fine to me.

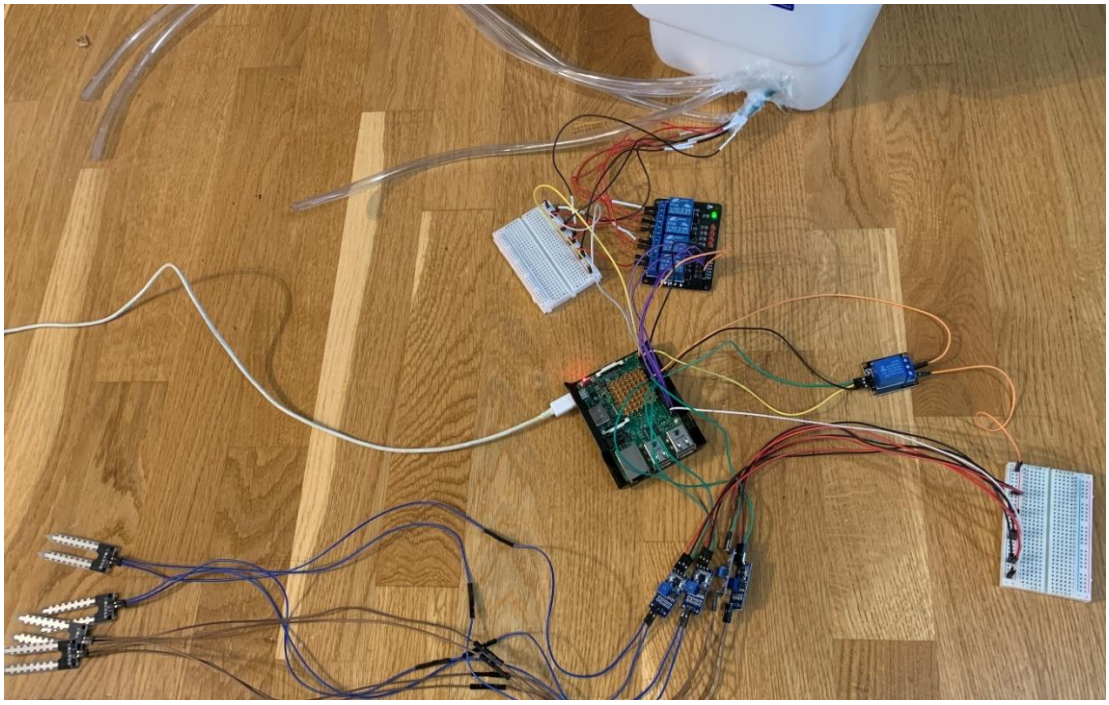
The sensor chip also has a fourth pin. It is an analog output which outputs a voltage based on *how moist* the soil is, but this project doesn't need it.

Two test programs are included: `sensortest.py` takes one reading, and `sensortest_long.py` turns the sensors on for one minute and reports at 10 second intervals. The software must be configured for these to work (see `config.txt`).

When the power is on, the sensors have helpful lights: one for indicating power and another for indicating moisture.

Building

Final setup



Here's a photo of the project almost ready to meet the plants. The tubes are easy to position with clothes pegs. Place the sensors so that a good amount of water hits them, and stick them all the way into the soil.

RasPi 5v power
RasPi 3v power
RasPi GPIO
RasPi GPIO
RasPi ground
RasPi ground and breadboard -
sensor and pump +

I thought I'll use this water container with a hole near the bottom, but that turned out to be a bad idea. Get something that is open from the top instead, like a bucket.

I falsely assumed a bottom hole would be necessary because the pumps use an almost non-existent amount of power (a single watt or even less). However, it turns out they can pump water some way up the tube just fine! (See the first page for an example.) Lesson learnt, and I now have a newfound respect for the power of one watt.

Moreover, these pumps can't stop water flow. That's why **the end of the tube must be higher than the water level in the container.** If the end of the tube is below the water level, the siphon effect will take all the water out, gruesomely drowning your plants.

Obviously, a DIY setup like this should be done with a finite water source and in a place where a leak does not result in water damage and deep regrets. My setup is on my balcony.

Using the software

- Grab the entire project as a zip from **https://github.com/Byproduct/RasPi-plant-waterer** and extract it in its own directory somewhere in your RasPi.
- Go to the directory you created, and give full permissions to all files in it:
  `chmod 777 *.*` (or be more selective as you please, keeping it simple here)
- Your RasPi probably has python 3 already. If not, install it:
  `sudo apt-get upgrade`
  `sudo apt-get install python3`
- Install the python discord extension:
  `pip3 install discord.py`
- Open **`config.txt`** and type in your settings. Instructions are inside.
- Test the program: `python3 irrigation.py`.
  If you need to troubleshoot, open **`config.txt`** and set `Debug_mode;true`
- If the script works as intended, you can set it run automatically, for example once per day. If you haven't used cron (scheduler) before, here's a tutorial .

  Cron with RasPi may be a bit fiddly - do whatever works. I made it work using root instead of the user pi. That is, using `sudo crontab -e` instead of `crontab -e`.

  Place a line like this in the crontab, replacing the time, path to your python3 installation, and the path to the script accordingly.

  ```
  0 15 * * * /usr/bin/python3 /home/pi/plantproject/irrigation.py > /dev/null 2>&1
  ```

  This example runs the script every day at 15:00. `> /dev/null 2>&1` just means that text output is discarded. Once everything works nicely, you don't need the script's output because the script updates irrigation_log.txt for you instead.

  If you need to troubleshoot, you can print the output to a file too, for example:

  ```
  0 15 * * * /usr/bin/python3 /home/pi/plantproject/irrigation.py > /home/pi/log.txt
  ```

…and that's it! You now (maybe) have an automatic irrigation system for your green friends. Below is an example of the irrigation_log.txt it produces. Of course, you will still occasionally need to refill the water tank and care for the plants otherwise.

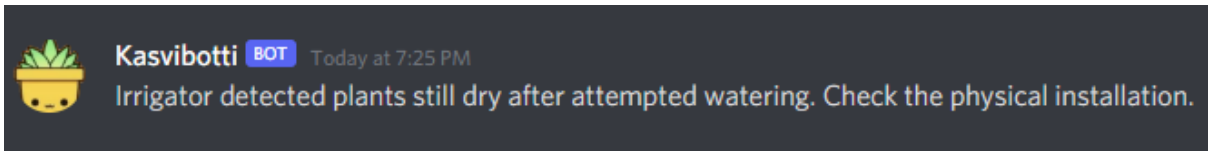
```
-----
2021-08-10, 12:00:10
Watered plants: Tomatoes
-----

-----
2021-08-11, 12:00:10
Watered plants: Chilis
-----

-----
2021-08-12, 12:00:10
All plants moist, did not pump any water.
-----

-----
2021-08-13, 12:00:10
Plants still dry after attempted watering.
Check the physical installation.
-----

-----
2021-08-13, 12:15:00
Watered plants: Tomatoes
-----

-----
2021-08-14, 12:00:10
All plants moist, did not pump any water.
-----
```

The project also includes RPi.zip which you can extract if you want to edit the code on your PC instead of on the RasPi. This is a fake library that doesn't do anything, but it allows the program to execute instead of erroring out. I've tested it only on Windows.

Using the discord bot

This step is completely optional. If you'd like to get notifications about your plants (and a way to check on them via discord message once that is implemented), you can.

This requires your RasPi to be connected to the internet.



Firstly, you'll need a discord bot. You can follow this tutorial to set one up. For this project you'll only need the token you'll get when the bot is created, not any of the code or functionality in the tutorial.

Also, you'll need the user ID of your discord account (yours, not the bot's). Here's how.

Open **`discordbot_config.txt`** and type in the bot's token and your ID.

On your RasPi, install screen if you don't have it already:
`apt-get install screen`

Then go to this project's folder and launch the discord bot inside screen:
`screen python3 discordbot.py`

You should see the bot starting up. Hit CTRL + A + D. Instead of quitting you detached from the screen, leaving it running in the background. Your discord bot is now always online and will notify you if anything is wrong with your plants. In fact, it will relay any messages you'd like to get notified about! It checks for **`messages.txt`** once per minute (or as often as you specify). You could combine it with your other projects.

If you want to terminate the bot you can resume it with `screen -r` and quit, or see running processes with `ps x` and kill the screen with `kill [process number]`. ☠

Greetings

If you have any questions or feedback, or just found this tutorial helpful, I'd be happy to know! I'm on Discord (Byproduct#9084). Email (byproduct@iki.fi) may or may not reach me due to the amount of spam these days.

Have fun!

Juhana