





Self-learning connected thermostat that optimizes heating and cooling of buildings

[WORK IN PROGRESS] This smart thermostat is an electronic, programmable, and self-learning Wi-Fi-enabled thermostat that optimizes heating and cooling of homes and businesses to conserve energy. It is based on a machine learning algorithm: for the first weeks users have to regulate the thermostat in order to provide the reference data set. The thermostat can then learn people's schedule, at which temperature they are used to and when. Using built-in sensors and phones' locations it can shift into energy saving mode when it realizes nobody is at home.

 Difficulty **Medium**

 Duration **6 hour(s)**

 Categories **Electronics, Energy**

 Cost **70 USD (\$)**

Contents

Introduction

Step 1 - Breadboard the hardware

Step 2 - Build an enclosure

Step 3 - Laser cut 3 acrylic disks

Step 4 - Solder the components

Step 5 - Software

Step 6 - End of programming

Step 7 - Connectivity

Step 8 - Putting it all together

Step 9 - Get excited, crazy things are possible.

Notes and references

Comments

Introduction

In January 2014, Google bought Nest, a connected devices company, for \$3.2 billion. This might seem like an ungodly sum for a company that makes thermostats and smoke detectors, but it makes absolute sense. Nest's products are beautifully designed, their team is overflowing with talent, and they were the first company to figure out what the "Internet of Things" means to consumers and deliver products that people actually want.

But in order to do this, Nest had to spend millions of dollars on R&D to build the basic infrastructure behind the product. The high cost made it impossible for anyone but the extremely well-capitalized to enter the market and create connected things.

Well, we want to change that. At Particle, we're making it easier to bring connected devices to market with the Particle Photon, our Wi-Fi development kit, and the Particle Cloud, our cloud service for connected devices. And to prove it, we built our own approximation of the Nest Learning Thermostat in one day — and we've open sourced everything. In this process, we've come to respect the incredible technical challenges that Nest has solved while also coming to understand how much the game has changed since they first started.

Materials

- The Particle Photon served as our connected brain.
- We display the temperature on a few Adafruit 8x8 LED matrices. The interface for the displays is a common I2C bus.
- The primary sensor is a Honeywell HumidIcon temperature and humidity sensor, which shares the I2C bus with the displays.
- For our MVP, we decided a couple LEDs could represent whether the heat and fan were on. In the end the same pins would be connected to relays instead of the LEDs.

If you want to save energy when a person's not home, then you need a way to know when they are home so you can err on the side of comfort again. We added a Panasonic PIR motion detector.

🔗 <https://github.com/spark/thermostat>

Step 1 - Breadboard the hardware

First, you need hardware. In our case, that means sensors for temperature and humidity, plus a motion sensor to figure out whether you're home, and relays to control the furnace and the fan. We also need a display so you can see the current temperature, and an enclosure to protect the messy bits.

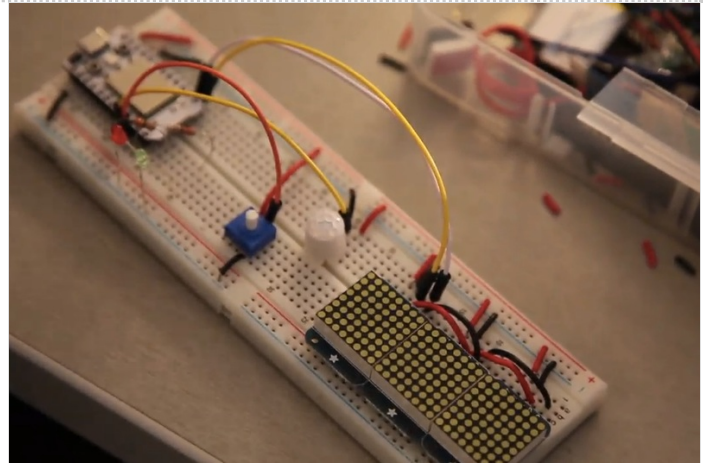
Breadboard the hardware (breadboarding is a non-permanent way to create an early hardware prototype).

We chose a number of components for this product (see the list of components above).

All in all, it took about an hour to throw together this breadboarded prototype, although we had to order the components a couple of days beforehand. It took another couple of hours to pull together working firmware (see the software section below).

Tools

- CNC machine
- Laser cutter
- Personal computer (for programming)
- Soldering iron

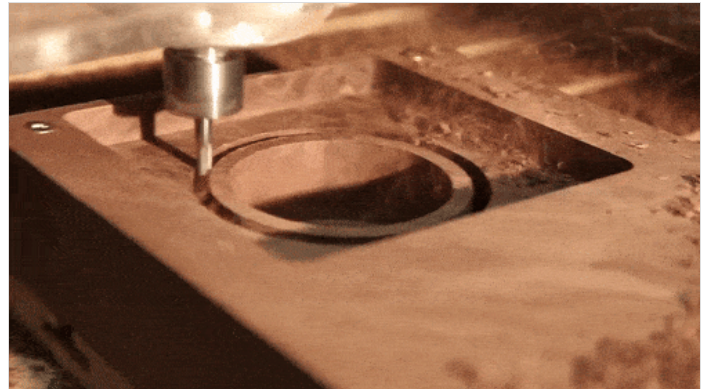


Step 2 - Build an enclosure

The Nest enclosure is glass and aluminum, which are both very pretty but not very handy for prototyping. Instead, we chose acrylic and wood.

First, we CNC milled two wooden rings: one to act as a stationary base, and other to spin freely as a temperature controller.

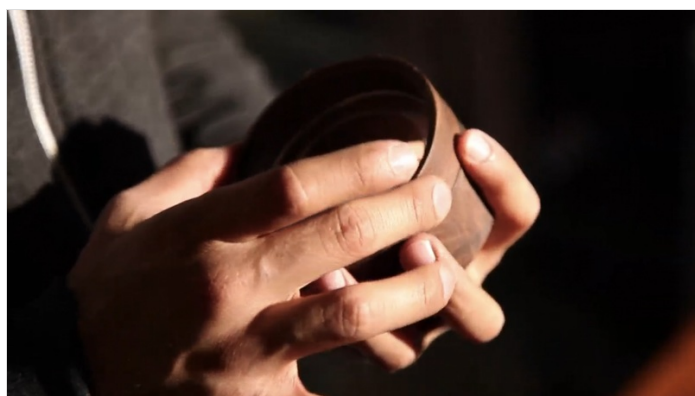
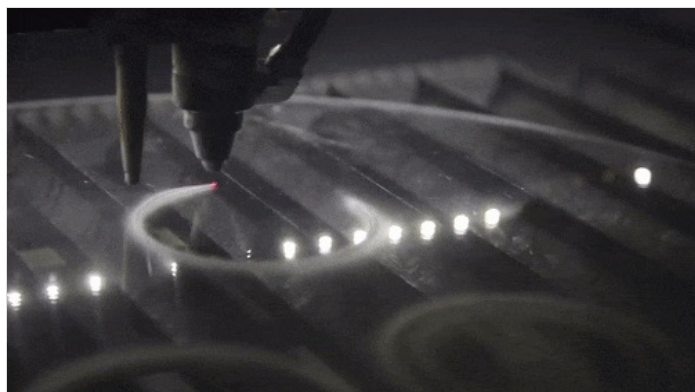
i Turn the ring clockwise to increase the temperature, and counterclockwise to decrease the temperature.



Step 3 - Laser cut 3 acrylic disks

Next, we laser cut three acrylic disks:

- one to act as the faceplate (which we later sanded to make it frosty);
- a second to act as the wall mounting plate;
- and a third connects the spinning wooden ring to a potentiometer.



Step 4 - Solder the components

Once we completed the enclosure, we converted our breadboarded circuit into a more permanent design by permanently soldering the components.

Step 5 - Software

Next, you need software. Some of this software runs on the thermostat (often called 'firmware'), reading data from the sensors, controlling the relays, and displaying data on the screen. But since this is a connected thermostat, we also want a web interface so that it can be controlled from your smartphone or computer. And since it's a learning thermostat, we also want to do some machine learning so that we can over time improve your comfort and energy efficiency. This software will run in the cloud.

Pin *Firmware* is called firm because it's traditionally more locked down than software, since it runs on a little chip that usually is never accessed again after the product is delivered to the customer. But adding an internet connection changes things pretty significantly. Firmware is no longer firm when you can update it from anywhere with the click of a button. With a Particle Photon, you can flash new code onto your chip using our web IDE.

Step 6 - End of programming

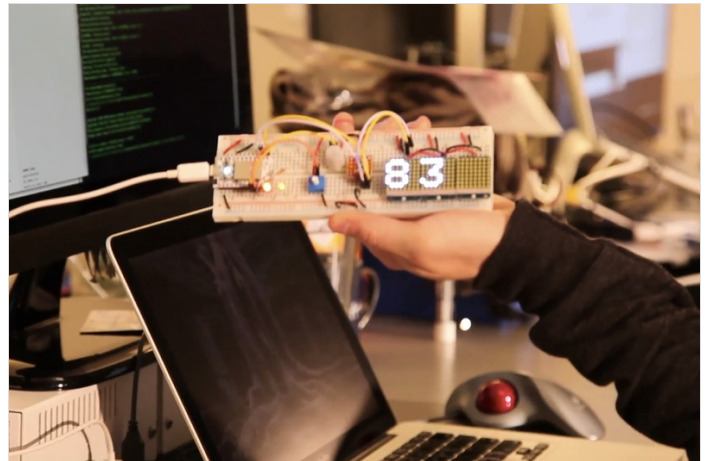
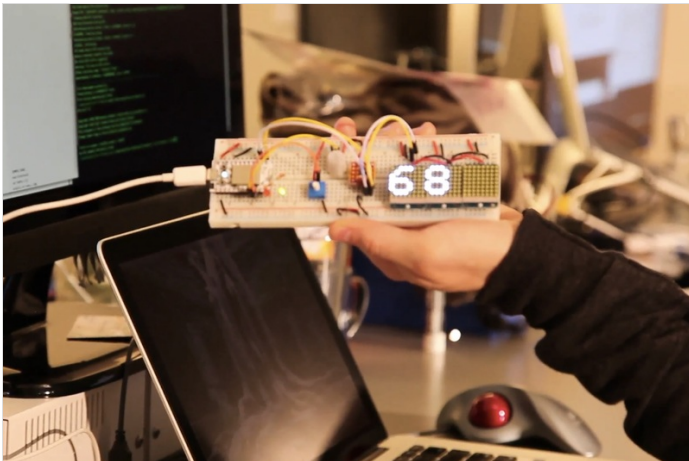
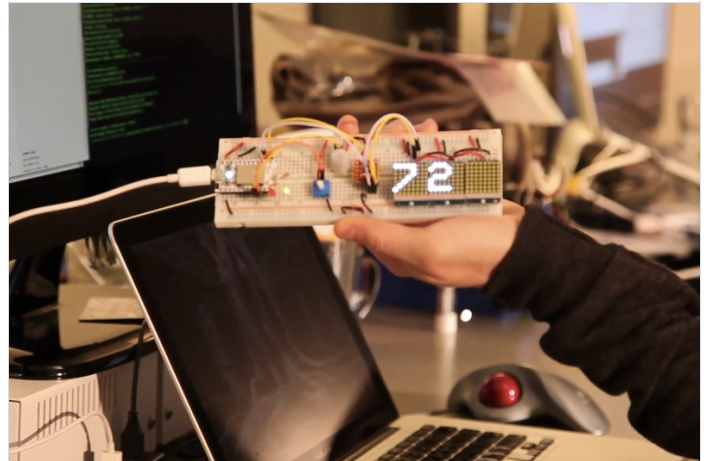
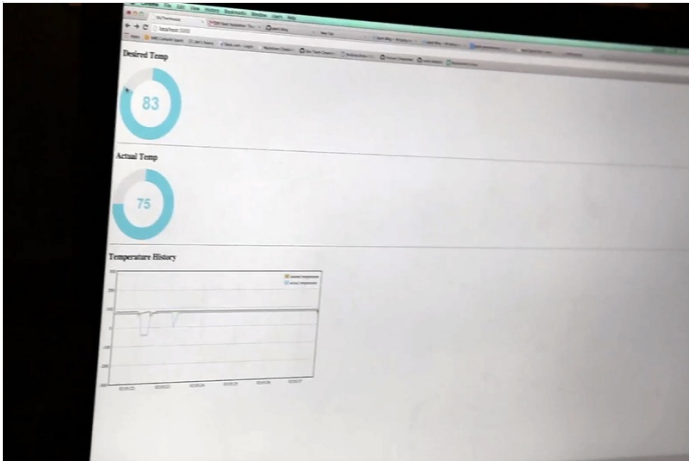
Our thermostat is complemented with a cloud-based web app that handles all of the complex logic of the thermostat.

💡 By doing this in the cloud, we can iterate faster using high-level programming languages and frameworks like Ruby on Rails rather than low-level embedded C.

The Particle Cloud exposes your connected device through a REST API. That means that you can interact with it from any language that can generate HTTP requests, which is basically anything.

The beauty of a connected device is that it can be constantly improving, whether it's by updating the firmware, updating the cloud software, or by using machine learning to optimize and improve the logic of the device.

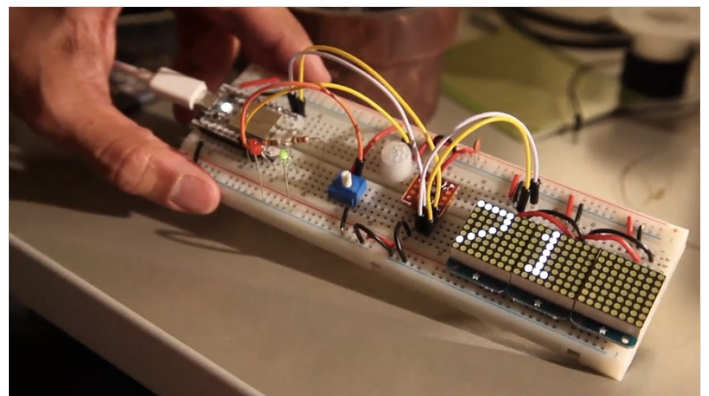
📌 Our user interface is a simple web app with a javascript knob that lets you select the desired temperature. The UI also includes a graph of historical temperatures, because data.



Step 7 - Connectivity

Once you've got your hardware and your software, it's time to link the two worlds.

Somehow you've got to get your thing online, and there are dozens of ways to do this. The simplest method is by adding a Wi-Fi module, so your product can act as a client on your local Wi-Fi network. The Particle Photon has a Wi-Fi module built in, and because it's integrated with the micro-controller, 'connectivity' is made easy. The Core automatically connects to the Particle Cloud through an encrypted tunnel, so you've got a secure connection to a cloud gateway out of the box. No programming the Wi-Fi module, and no finding or building communications protocols.

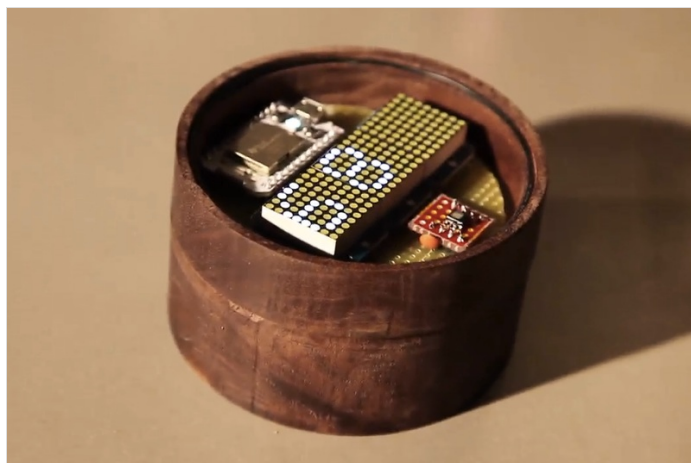


Step 8 - Putting it all together

Once our thermostat was complete, it was time to assemble it all together into the final package and mount it on the wall.

All in, we spent about \$70 on components to put this together (including \$39 for the Particle Core); the wood and acrylic were free.

We started working at 10am and finished at 3am, with 3.5 engineers involved (one went to bed early), and the only work we did in advance was order the electronic components.



Step 9 - Get excited, crazy things are possible.

We're not saying that you can build a \$3.2 billion company in a day. But we are saying that you can build a \$3.2 billion company, and it's easier now than it's ever been before.

Connected devices/Internet of Things/M2M/Industrial Internet is a certified big deal, and the Nest acquisition proves it. It doesn't matter whether you're a software developer with no hardware experience, an embedded designer with no web experience, or a psych major with no experience whatsoever. This is the next frontier, and it's time to check it out.

Your billion dollar company starts with a million dollar product, and your million dollar product starts with a hundred dollar prototype.

So what are you waiting for?



i Fair warning - we're not claiming to have matched the Nest thermostat in a day; far from it. But remember — every polished product starts as a rough prototype. As Alexis Ohanian said, "The first version of everything you love is janky!"

Notes and references

- To download the open source files for this thermostat, visit our [Github page](#).
- To get the Photon tiny Wi-Fi development, visit our [store](#)