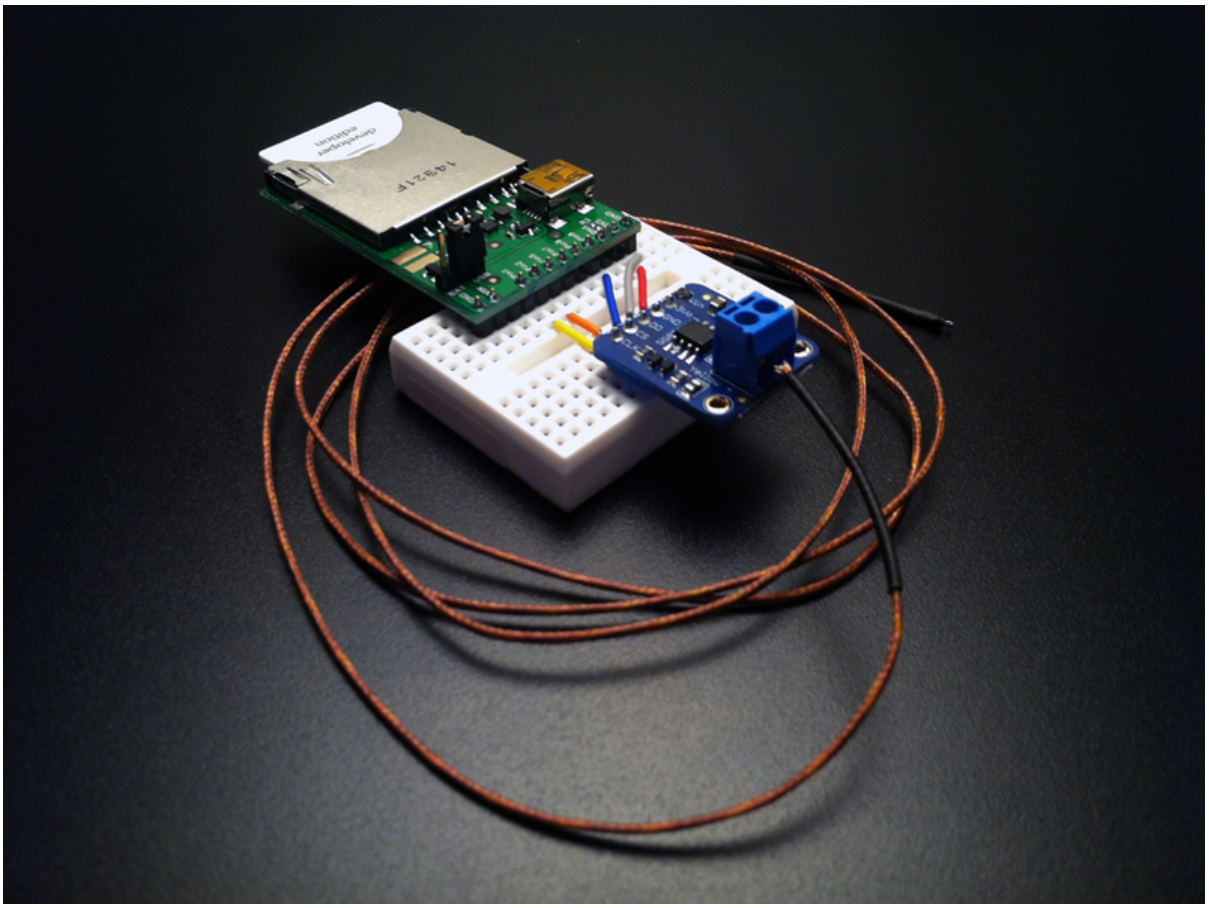
Connecting the MAX31855 Thermocouple Amplifier breakout to an Electric Imp

Created by Joel Wehr



https://learn.adafruit.com/connecting-the-max31855-thermocouple-amplifier-breakout-to-an-electric-imp

Last updated on 2024-03-08 01:51:48 PM EST

Table of Contents

Overview 3

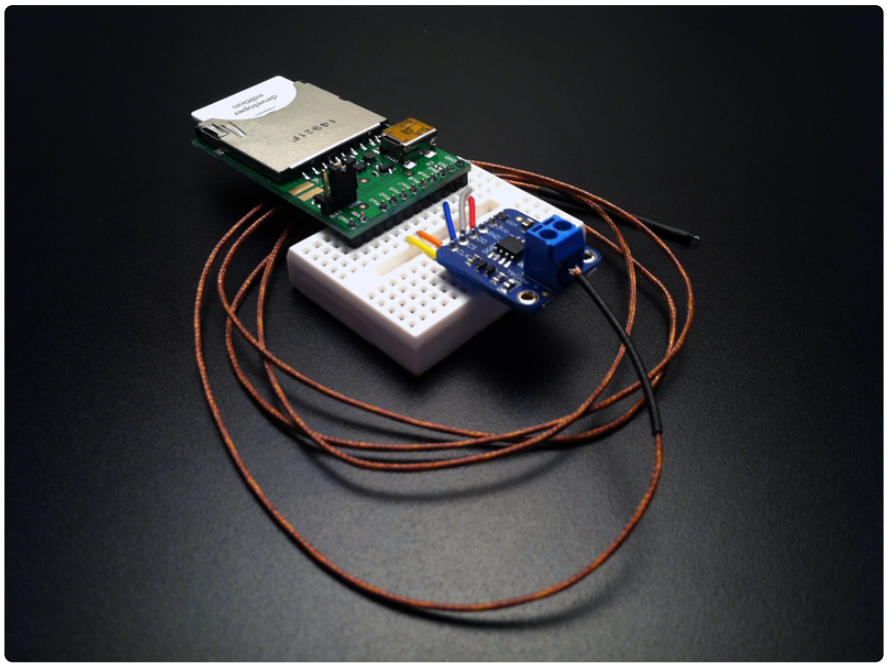
Breadboarding the circuit 4

Get the Device code 10

Configure the Agent for Xively 13

Configure the Agent for Twitter 19

Overview



If you are interested in Internet of Things projects, this tutorial is a simple and very practical way to get started. The Electric Imp platform was created specifically for Internet of Things devices, and measuring temperature is a great way to learn.

In this tutorial, we will be connecting an Electric Imp to the Adafruit MAX31855 Thermocouple Amplifier breakout board using one of the Serial Peripheral Interface Buses (SPI) available on the Electric Imp. Then we will connect a K-type thermocouple, and send the data to Xively and Twitter.

Our maker objectives:

- Assemble the breakout boards.
- Breadboard the circuit, and attach the thermocouple.
- Load the Electric Imp device code, and log temperature data in the Web IDE.
- Create a Xively.com developer account, and a Feed and Channel to push data to.
- Load/Configure the Electric Imp agent code for Xively, and push temperature data.
- Optionally, download the iPhone app "Pitchfork", and subscribe to our Xively Feed.
- Create a Twitter developer application and configure it to tweet from the Imp.
- Load/Configure the Electric Imp agent code for Twitter and tweet our temperature data.

Our learning objectives:

- Understand how to configure and use SPI on the Electric Imp.
- Electric Imp pin configuration: http://devwiki.electricimp.com/doku.php?id=imppinmux (https://adafru.it/cEB)
- Read up on SPI here: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus (https://adafru.it/cEC)
- Understand how to connect to and read from the MAX31855 thermocouple amplifier.
- MAX31855 datasheet: http://www.adafruit.com/datasheets/MAX31855.pdf (https://adafru.it/aLh)
- Understand how thermocouples work. http://learn.adafruit.com/thermocouple/overview (https://adafru.it/cED)
- Learn how to create a Xively feed and push data to it.
- Learn how to create a Twitter developer application and tweet from the Electric Imp.

Thermocouples are really great for extreme temperatures! Here are some projects ideas.

- BBQ/Grill/Smoker temperature monitor
- Oven temperature monitor
- Homebrewing temperature monitor (Mashing, chilling, boiling, fermenting, ect)
- SMT reflow oven temperature monitor/control
- Fridge/Freezer/Kegerator temperature monitor
- Soldering Iron temperature monitor

Breadboarding the circuit

Let's get started creating our Internet of Things temperature monitor!

FAQ: Are there two versions of the Electric Imp?

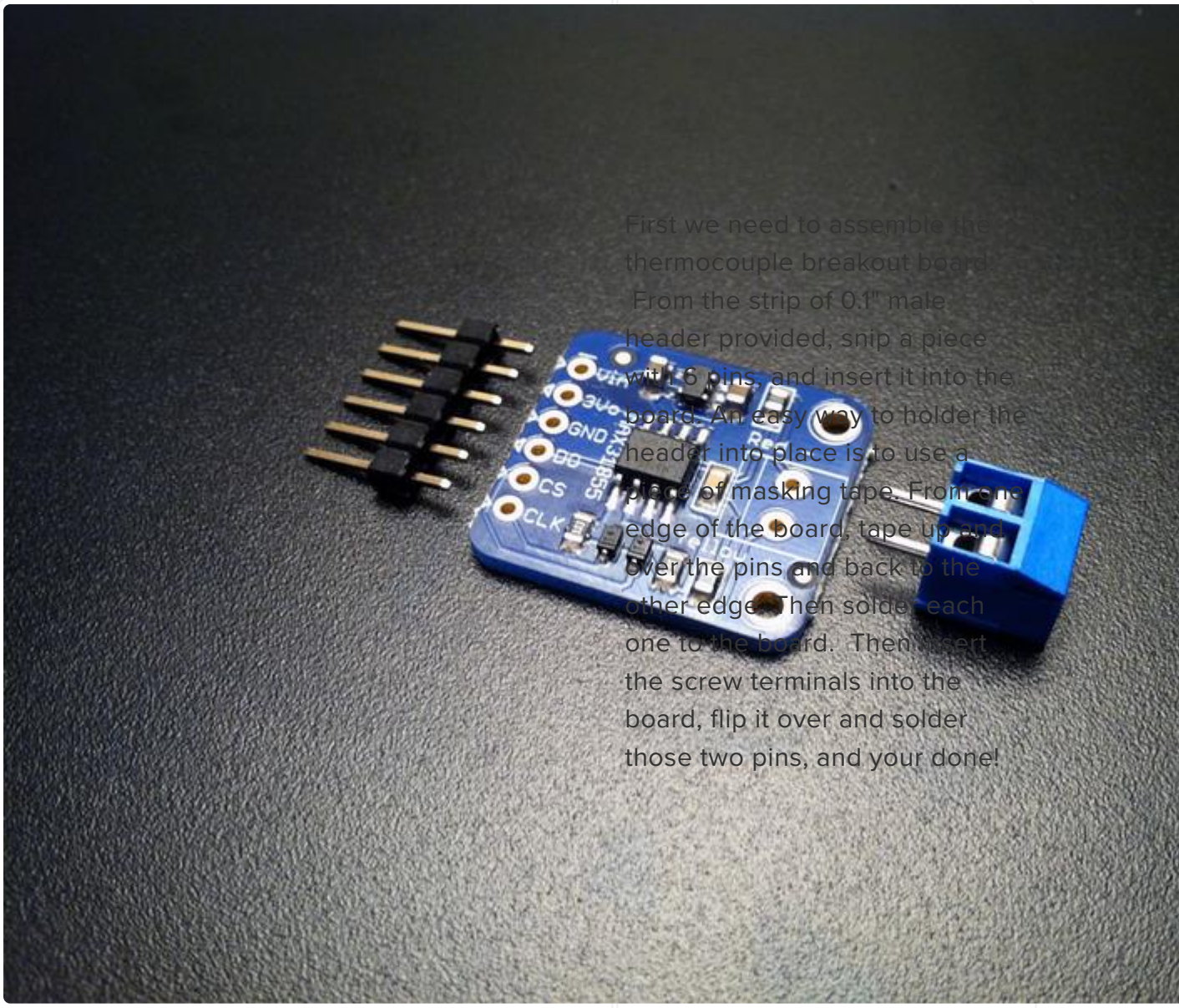
Yes. Pictured below are the imp001 on the left, and the imp002 on the right. The imp001 fits into most SD card sockets and contains all of the hardware to "Blink-Up" or connect to your wireless network. The imp002 is a solder down version and allows the developer to create a custom "Blink-Up" circuit. All 12 pins are available with the imp002. We will be using the imp001 for this tutorial, since it contains all of the circuitry to get us going.

This tutorial assumes that your Electric Imp is connected to your WiFi network (Blinked-Up), and that you have access to the [illegible] which is [illegible] haven't, head on over to www.electricimp.com, sign up and follow the commissioning process directions. Alternatively, check out this really nice instructable by Matt Haines from Electric Imp.

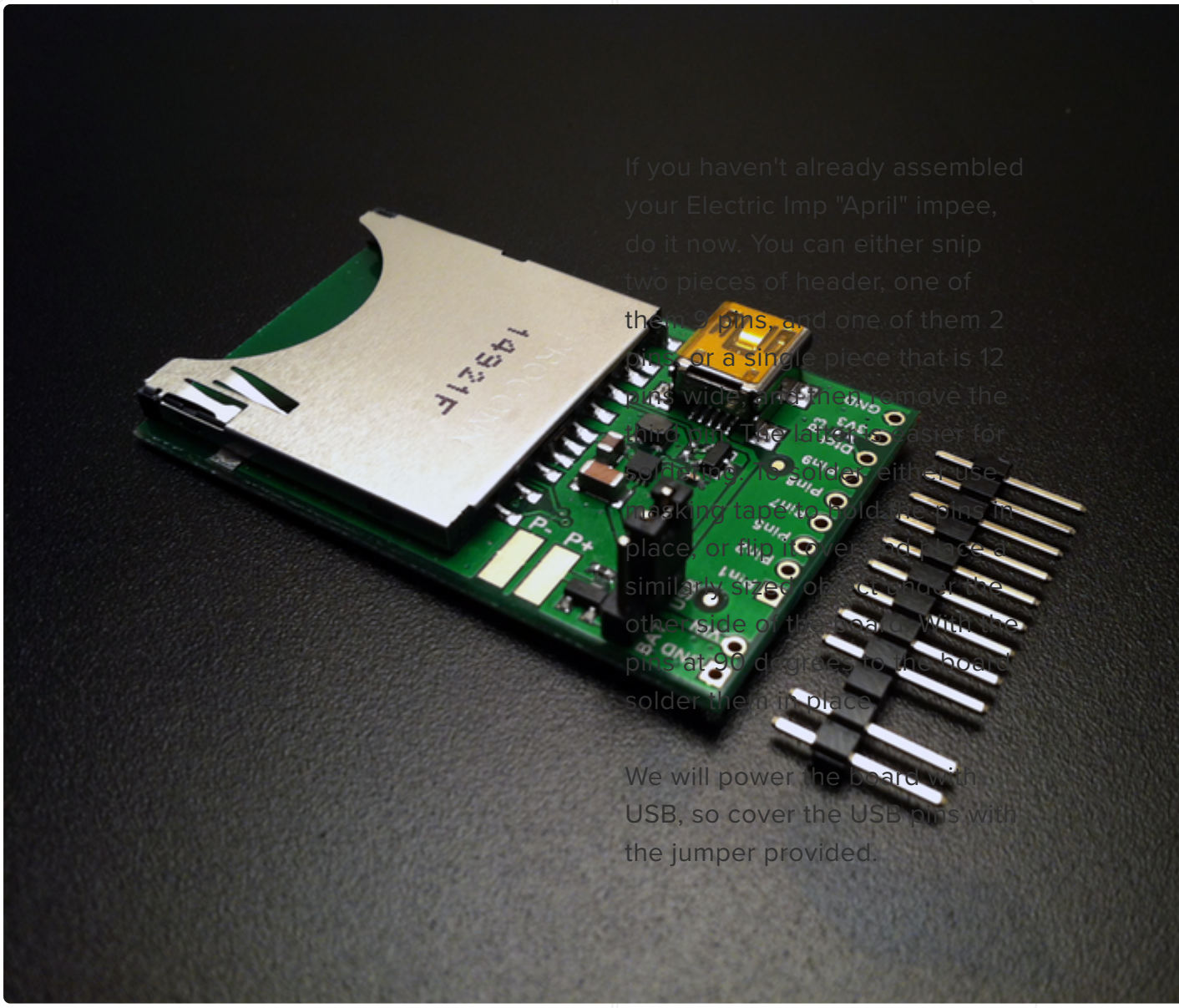
http://www.instructables.com/id/Getting-Started-with-Electric-Imp/

[illegible] happily blinking green, you are ready to get started.

If for some reason you have trouble getting connected, the best way to get help is to post on the Electric Imp forums from your developer account. Electric Imp staff and forum members post and reply day and night, so you'll get help right away.



IMPORTANT! You can solder the pins to these boards facing either up or down. If you want to plug or solder it directly into a breadboard, solder the pins facing down. If you want to lay it flat, and connect it to another device with jumpers, solder it with the pins facing up.

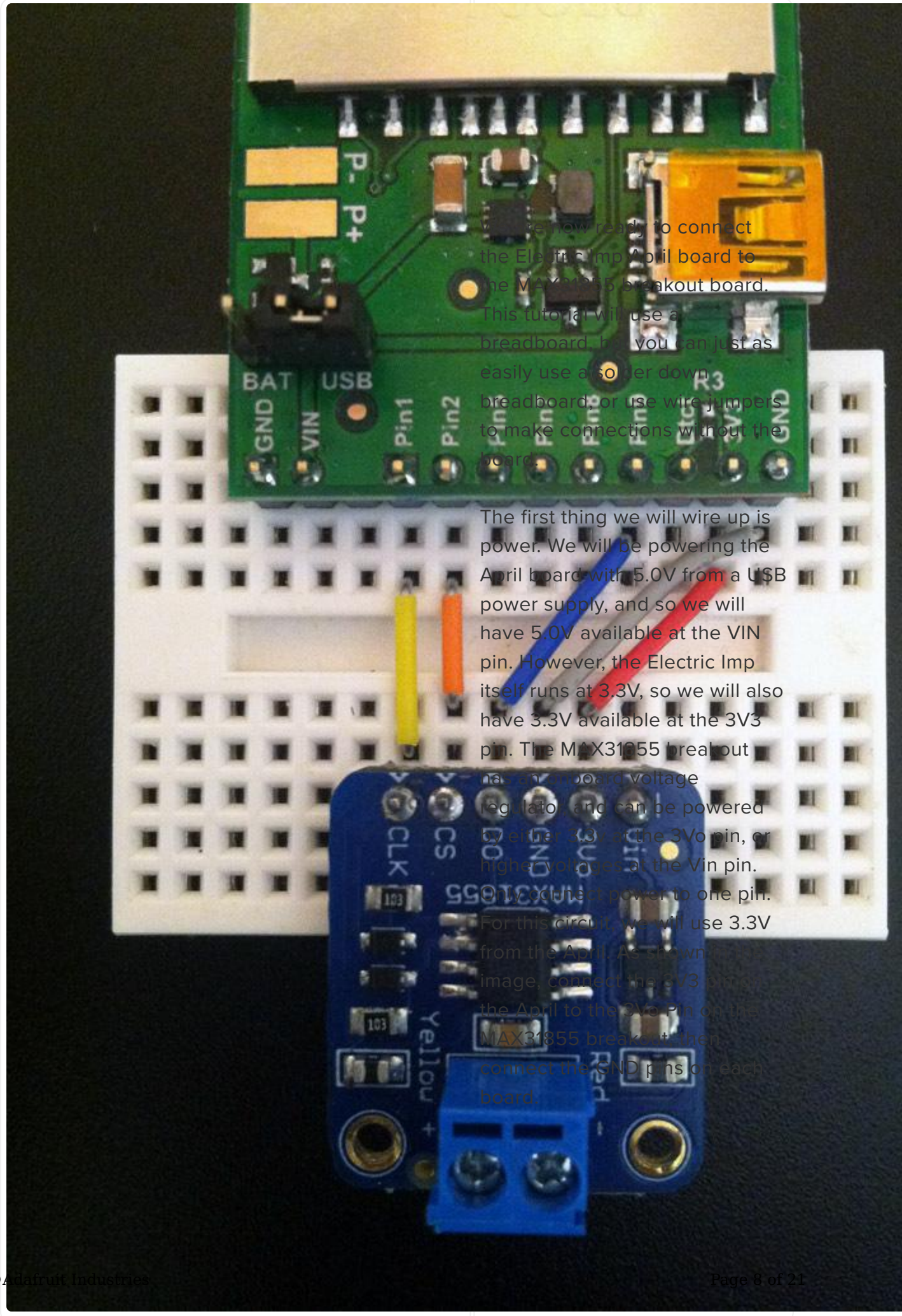


FAQ: What is an impee?

Any device or breakout board that is powered by an Electric Imp is called an "impee". The April impee is one of the reference designs created by Electric Imp for developer use. You can check out the other designs here:

http://devwiki.electricimp.com/doku.php?id=boards:start (https://adafru.it/cEV)

These designs also include a Bill of Materials (BOM) and Gerber files, in case you want to try your hand at printing and assembling the boards yourself.



We're now ready to connect the Electric Imp April board to the MAX31855 breakout board. This tutorial will use a breadboard, but you can just as easily use a solder down breadboard or use wire jumpers to make connections without the board.

The first thing we will wire up is power. We will be powering the April board with 5.0V from a USB power supply, and so we will have 5.0V available at the VIN pin. However, the Electric Imp itself runs at 3.3V, so we will also have 3.3V available at the 3V3 pin. The MAX31855 breakout has an onboard voltage regulator and can be powered by either 3.3V at the 3Vo pin, or higher voltages at the Vin pin. Only connect power to one pin. For this circuit, we will use 3.3V from the April. As shown in the image, connect the 3V3 pin on the April to the 3Vo pin on the MAX31855 breakout, then connect the GND pins on each board.

The Electric Imp has two sets of pins for SPI. Those sets are pins 1,8 & 9, and pins 2,5 & 7. For this tutorial we will be using SPI189 as it is referred to in hardware configuration. When you configure SPI189 you get the following pin configuration in the Electric Imp.

- PIN 1 - CLK or Clock
- PIN 8 - MOSI (Microcontroller Out/Sensor In)
- PIN 9 - MISO (Microcontroller In/Sensor Out) or Data Out

This line of code configures SPI189 in the device:

hardware.spi189.configure(MSB_FIRST | CLOCK_IDLE_LOW , 1000);

Since we will only be reading data from the MAX31855 board (sensor), we only need to connect the MISO pin, which is pin 9, and not MOSI. We also need Pin 1 to provide a clock signal, and one more pin to act as a Chip Select (CS) pin. In order to tell the MAX31855 that we want to read SPI data, we pull the CS pin LOW, read the data, and then set the pin back to HIGH. We can configure any of the other pins as a CS pin, so we will use Pin 2, since it is directly across from the CS pin.

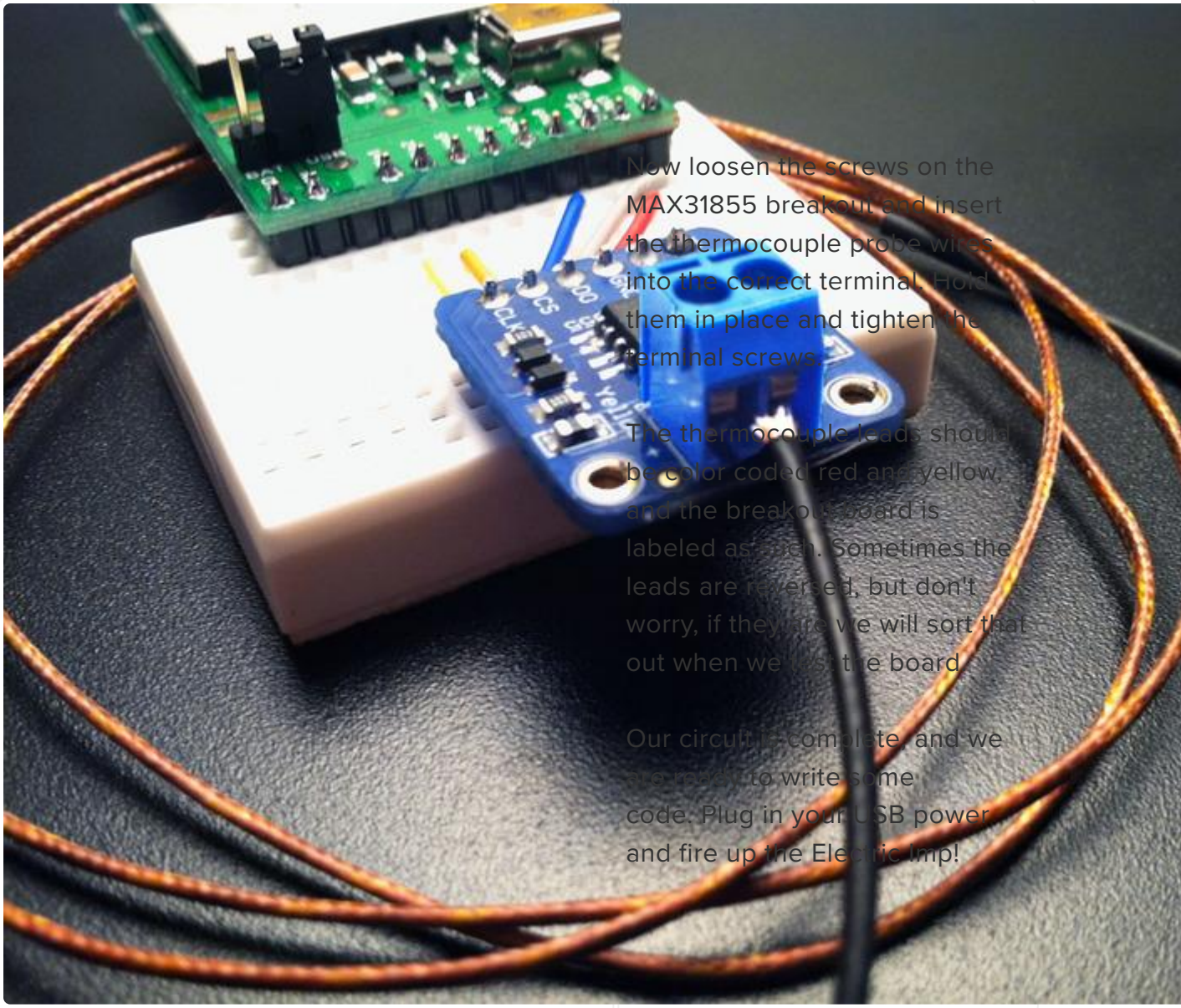
This code reads SPI data from the chip that is connected to Pin 2:

hardware.pin2.write(0); //pull CS LOW to start the transmission of temp data
local temp32=hardware.spi189.readblob(4);//SPI read is totally completed here
hardware.pin2.write(1); // pull CS HIGH

Note: You can read as many chips as you like from a single SPI bus, as long as you have a Chip Select Pin for each. Using just the April board, you could read up to four MAX31855 chips. This is very handy for a project with multiple thermocouples.

To complete the wiring, connect these pins as shown above.

April - MAX31855
PIN1 - CLK
PIN2 - CS
PIN9 - DO
3V3 - 3Vo
GND - GND



Now loosen the screws on the MAX31855 breakout and insert the thermocouple probe wires into the correct terminal. Hold them in place and tighten the terminal screws.

The thermocouple leads should be color coded red and yellow, and the breakout board is labeled as such. Sometimes the leads are reversed, but don't worry, if they are we will sort that out when we test the board.

Our circuit is complete, and we are ready to write some code. Plug in your USB power and fire up the Electric Imp!

Get the Device code

Electric Imp: The agent and the device

One of the really interesting things about the Electric Imp is that you are able to program both the device, and something called an "agent", which is a secure, programmable web service that you use to interface with your Electric Imp device. If you are coming from a platform like Arduino, you may be wondering why you can't talk directly to the Electric Imp itself. That question is beyond the scope of this tutorial, but I have found that the agent is a very handy service to have. Here are a few great features of agents:

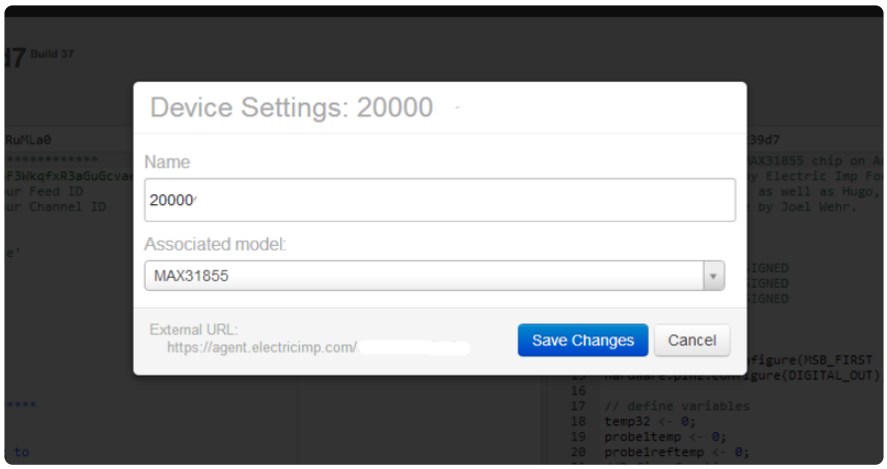
- Communicate using HTTPS

- Additional memory and processing power for code
- If you device goes offline, your agent will continue to execute code and can inform you.

As an example of the last point, if you use an Electric Imp in a security system, and a thief disables your connection or the device, your agent can still alert you that there is a problem.

The device can operate without any programming in the agent, so the first thing we need to do is program the device to read data from our thermocouple amplifier via SPI.

Log into your Electric Imp account and open the Web IDE. Find the Electric Imp that you will be using by its hardware address in the list of your Devices on the left hand side. Mouse over it, and click the gear symbol to create new Device Settings. Click the down arrow beside associated model, and then type a new model name, such as "MAX31855".



Note: While the web IDE is still in Beta, you may find that the screenshots and commands listed here are slightly different from the version that you are using.

The Electric Imp code for this tutorial is available at the GitHub page listed below. Typically, you store code for the device in a file called "**device.nut**", and code for the agent in a file called "**agent.nut**" The Electric Imp programming language, "ImpOS" is a slightly modified version of the Squirrel programming language.

https://github.com/joel-wehr/Tutorial_Electric_Imp_MAX31855 (https://adafru.it/cEW)

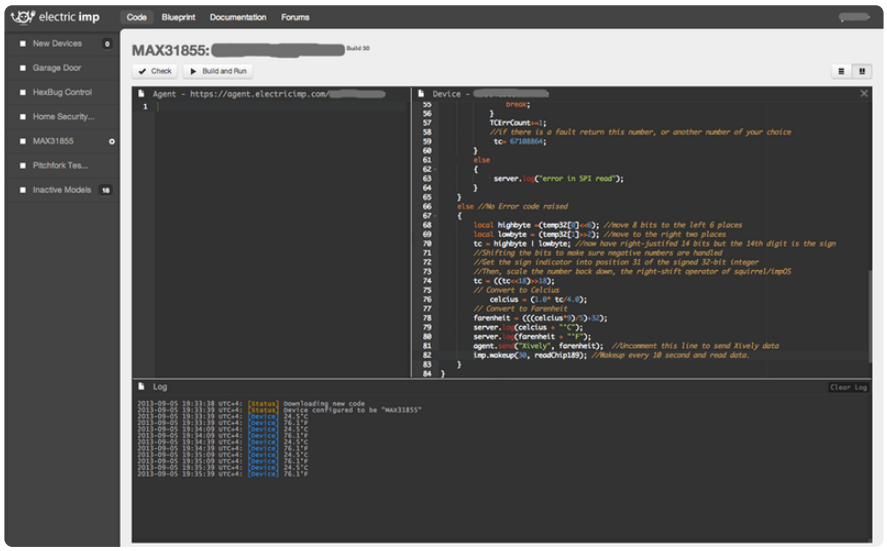
Credit for this code goes to a number of members of the Electric Imp forum community as well as members of the Electric Imp team. Big thanks to beardedinventor, bodinegl, rivers, and mjkuwp94. Check out the Electric Imp GitHub

page for example and reference code.

Download the .zip archive.

https://adafru.it/cEX

Copy the code from device.nut into the Device panel in the web IDE.



Click "Build and Run"

If all is well, you should see the Imp download the new code, and start logging data every 10 seconds like this:

2013-09-04 21:33:28 UTC+4: [Status] Downloading new code
2013-09-04 21:33:28 UTC+4: [Status] Device configured to be "MAX31855"
2013-09-04 21:33:28 UTC+4: [Device] 24.5°C
2013-09-04 21:33:28 UTC+4: [Device] 76.1°F

Nice! The Electric Imp is reading our thermocouple data over SPI!

If you haven't read through LadyAda's tutorial on thermocouples, amplifiers and the Seebeck effect, I highly recommend that you do. Working with thermocouples will make much more sense.

http://learn.adafruit.com/thermocouple/overview (https://adafru.it/cED)

Troubleshooting: If you are getting unexpected data from your thermocouple, first make sure that the leads are fully secured in the screw terminals. Hold the end of the thermocouple in your hand, and check to see if the temperature

reading rises. If it doesn't, or goes lower, swap the red and yellow thermocouple leads and try again.

If you "Check" your code, it will attempt to compile, and you will see an X in a red box by the lines of code that throw an error. If you "Build and Run" your code and errors occur, check the log for the line number that threw the error.

Configure the Agent for Xively

Please Note: Xively no longer has free developer access to their system, so this tutorial is only for historical research. Please check out our other IoT tutorials for alternative services!

The Electric Imp is a really great microcontroller in a tiny package, but its true purpose is to connect devices to the Internet, so lets get to it!

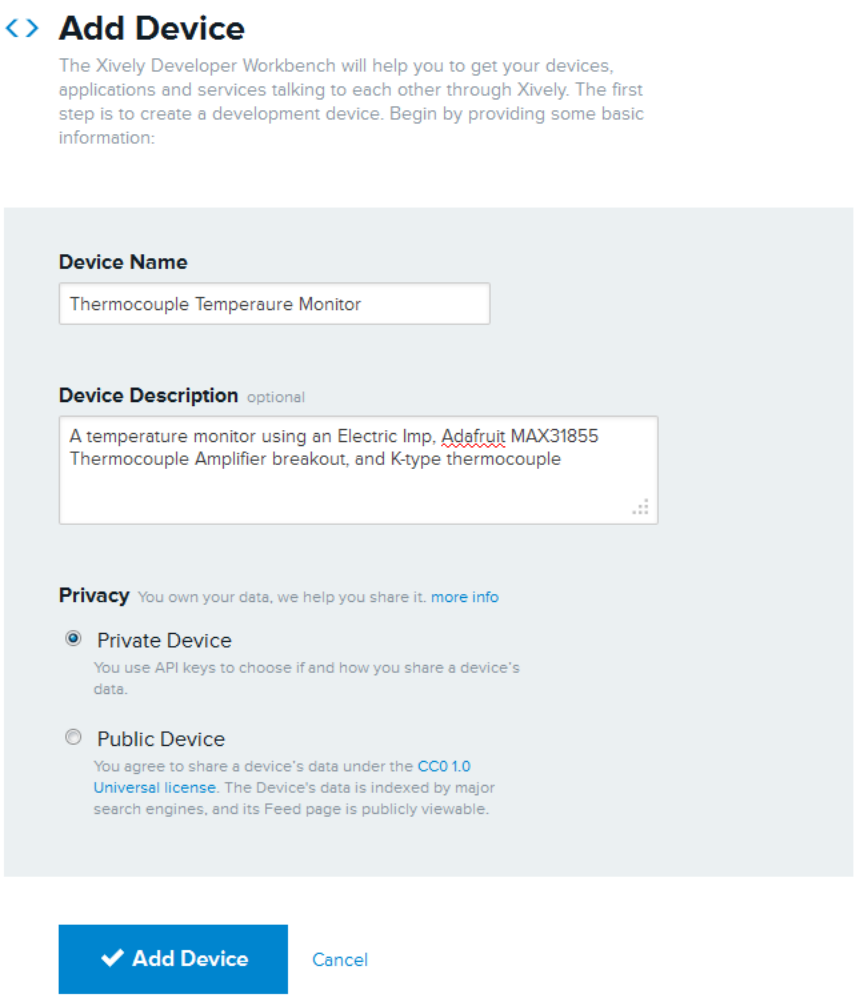
It seems that there are new services to push your data to every week! One of those has been around for a while and is now called Xively. You may have known it when it was Cosm, or Pachube. Let's set it up to log and graph our data.

If you haven't already, go to https://xively.com/signup/ (https://adafru.it/cEZ) and create a free developer account.

After you are set up and logged in, Click the DEVELOP tab, and +Add Device.

Give your device a name, a description, and then choose a privacy setting. For now, choose Private Device as we will be working with API keys.

Click Add Device.



Take a look around at your new Xively feed. The first thing we need to do is create a channel to push our thermocouple data to. Click the blue "+ Add Channel" bar.

Fill out the channel information.

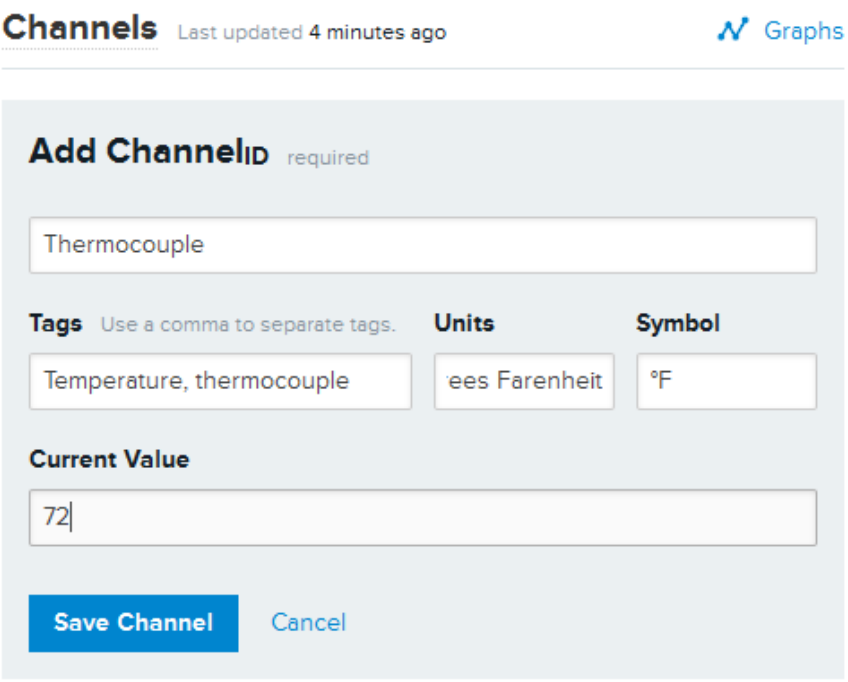
To type the degree symbol "°":

On a Mac type: Shift+Option+8 for the degree symbol
On a PC numeric keypad type: Alt+0176.

(or just copy and paste the character above!)

Save the channel.

If you like, you can set a Location, and Metadata.



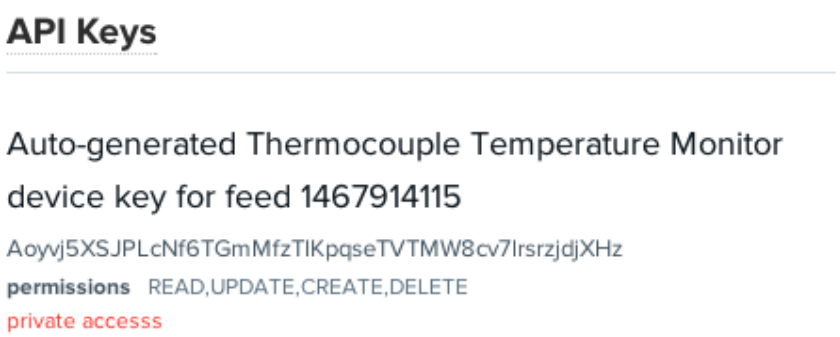
Now we need to configure the Electric Imp's agent to talk to our Xively channel. Copy the code from the "**agent.nut**" file that you downloaded from the GitHub tutorial repo. Then open the Electric Imp Web IDE and paste the code into the Agent panel for your MAX31855 Imp.

There are three sections in the agent code. The top section is code to talk to the Xively.com API. The next section is code to talk to the Twitter API, as well as SHA1 encryption code. Finally, at the bottom is a function that will be run each time the device executes the agent.send() function.

You'll need to enter three items from your Xively account into the agent code. Under the API Keys panel in your Xively developer workbench, copy the entire auto-generated API Key, and paste it inside the quotes for the API-Key variable at the top of the agent Xively code. Then find the Feed ID above the API Key, and copy and paste it to its variable. Lastly, type or paste the name of your channel into the Channel_ID variable. My channel was named "Thermocouple". This section of code looks like:

```
API_Key <- "YOUR API KEY"; //Type your Xively API Key
Feed_ID <- "YOUR FEED ID" //Type your Feed ID
Channel_ID <- "YOUR CHANNEL ID"; //Type your Channel ID
```

The Xively API Key sections looks like this:



Now, in the Device code panel, uncomment line 81 or the line that has this code:

agent.send("Xively", farenheit); //Uncomment this line to send Xively data

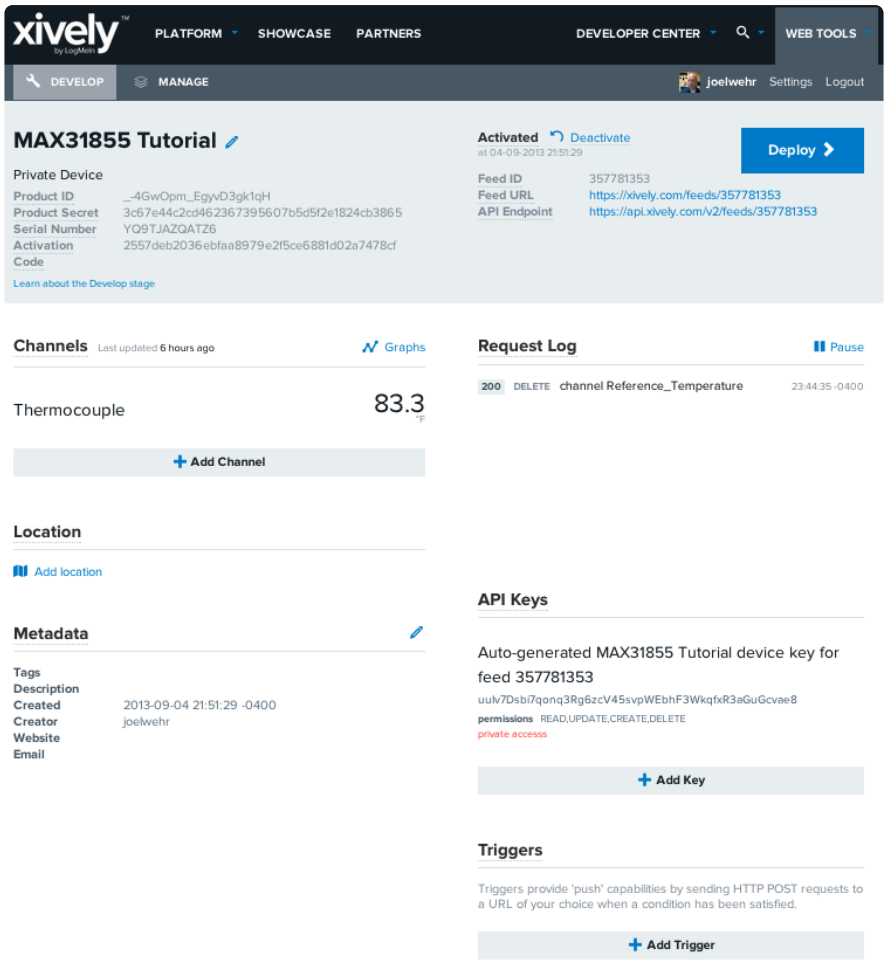
Build and Run your code, and you should see something like this:

2013-09-04 22:44:21 UTC+4: [Status] Device booting
2013-09-04 22:44:21 UTC+4: [Status] Device configured to be "MAX31855"
2013-09-04 22:44:21 UTC+4: [Device] 24.5°C
2013-09-04 22:44:21 UTC+4: [Device] 76.1°F
2013-09-04 22:44:21 UTC+4: [Agent] { "id": "Thermocouple", "current_value": 76.1 }

Take a look at your Xively channel. You should see the Electric Imp begin sending data to the feed in the Request Log, and the data should show up in your channel and on the graph.

Nice! You are sending thermocouple data to Xively!

Troubleshooting: Make sure you copied all the information correctly. Check the API Key to make sure it has "create" privileges. It should by default, but if it doesn't you'll need to create one that does. "Read" privileges are not enough.

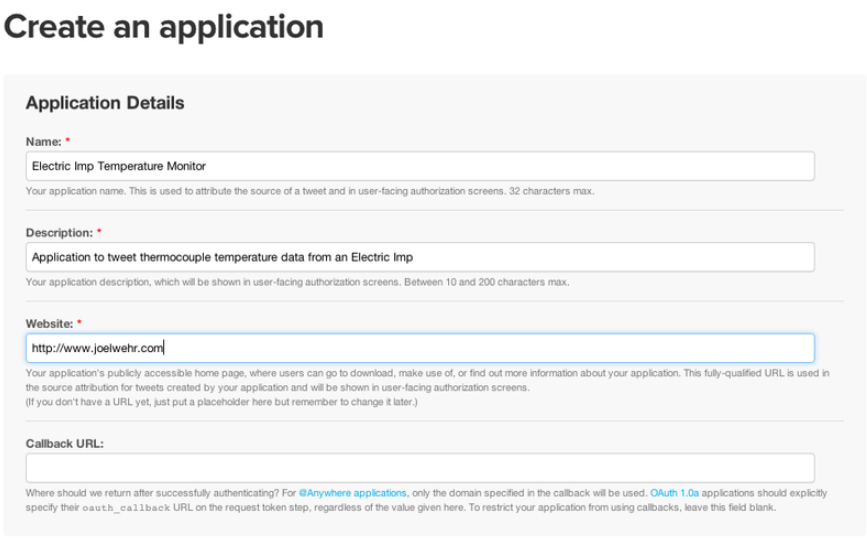

xively
PLATFORM
SHOWCASE
PARTNERS
DEVELOPER CENTER
WEB TOOLS
DEVELOP
MANAGE
joelwehr
Settings
Logout
MAX31855 Tutorial
Private Device
Product ID _-4GwOpm_EgyvD3gk1qH
Product Secret 3c67e44c2cd462367395607b5d5f2e1824cb3865
Serial Number YQ9TJAZQATZ6
Activation Code 2557deb2036ebfaa8979e2f5ce6881d02a7478cf
Learn about the Develop stage
Activated
at 04-09-2013 21:51:29
Deactivate
Deploy
Feed ID 357781353
Feed URL https://xively.com/feeds/357781353
API Endpoint https://api.xively.com/v2/feeds/357781353
Channels
Last updated 6 hours ago
Graphs
Thermocouple
83.3
°F
Add Channel
Location
Add location
Metadata
Tags
Description
Created 2013-09-04 21:51:29 -0400
Creator joelwehr
Website
Email
Request Log
Pause
200 DELETE channel Reference_Temperature 23:44:35 -0400
API Keys
Auto-generated MAX31855 Tutorial device key for feed 357781353
uulv7Dsbi7qonq3Rg6zcV45svpWEbhF3WkqfxR3aGuGcvae8
permissions READ,UPDATE,CREATE,DELETE
private accesss
Add Key
Triggers
Triggers provide 'push' capabilities by sending HTTP POST requests to a URL of your choice when a condition has been satisfied.
Add Trigger

Carrier
5:45 PM
Xively Datastream
last updated: 21:45 2013/08/22
Grill_Temperature
73.2°F
degrees Farenheit
82.65
MIN VALUE
43.5
MAX VALUE
Food_Probe_Temperature
73.2°F
degrees Fahrenheit
84.0
MIN VALUE
28.65
MAX VALUE
Pitchfork
Now, what if you wanted to monitor this data on your iPhone or iPad? Search for an app called Pitchfork in the App Store. This app was created to work with the Electric Imp and Xively. Find the Xively Data panel, and enter the settings for your API key, Feed and Channel into the app. Now, everytime you open that panel, the app will subscribe to your feed and update every time there is new data. The app allows you to configure a single feed with up to two channels. Now you can keep an eye on your temperature from anywhere!
When you are comfortable with Xively, take a look at Triggers. You can set triggers for each channel to enable actions on events. For more fun, take a look at Zapier.com. You can hook Xively triggers to Zapier "Zaps" and perform all kinds of cool actions.
Control Panel
Xively Data
Garage Door
Security System

Configure the Agent for Twitter

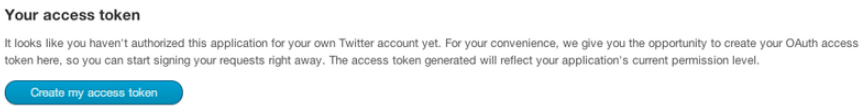
Next, let's set up our Electric Imp to tweet its temperature! You'll need a Twitter account, so go ahead and create one of you haven't already. In order to tweet from our Imp's agent, we will need to access the Twitter Developer Center. https://dev.twitter.com/apps/new (https://adafru.it/cF0)

Fill out the Name, Description, and Website information. Agree to the terms and conditions, and complete the Captcha. If all is well, Twitter will create an new application for you.

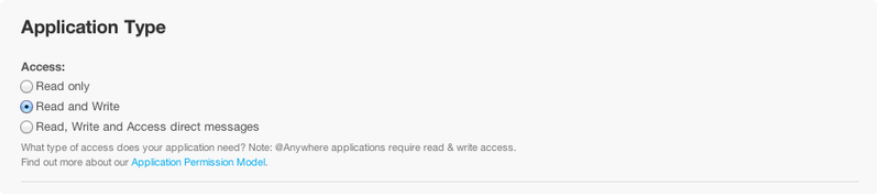


Take a look at what is on the page. At the bottom, click "Create my Access Token". Twitter will create Access tokens for you.

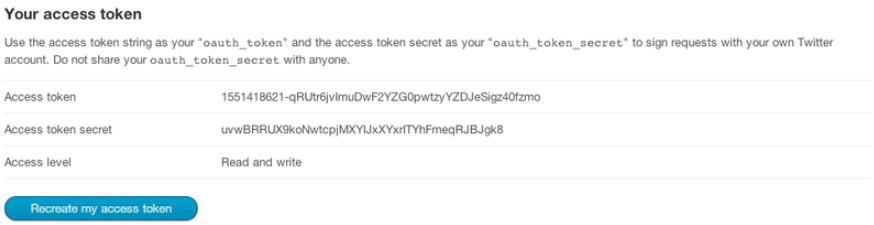
I have found that I need to click the buttons to create Access Tokens, and to change the Application Type twice before it updates properly. Ensure that you have what you need before moving on.



The token created is read-only, and we need to give it write access. Click on the Settings tab at the top. Scroll down to Application Type, and check "Read and Write". Then click "Update this Twitter application's settings" at the bottom. Make sure it updates to "Read and Write".



Now go back to the Details tab. We need to update our Access Token. Click "Recreate my access token" at the bottom. Make sure it says "Read and Write" after you update it.



We now have all the information we need to tweet. From the Details tab in your Twitter application page, carefully copy the appropriate information into the variables near the bottom of your Agent code in the Electric Imp Web IDE. You'll need to fill out these variables:

_CONSUMER_KEY <- "YOUR KEY"
_CONSUMER_SECRET <- "YOUR SECRET"
_ACCESS_TOKEN <- "YOUR TOKEN"
_ACCESS_SECRET <- "YOUR SECRET"

Paste your information inside the quotes.

Finally, uncomment this line near the end of the Agent code:

//twitter.update_status("The temperature is: " + v + "F.");

If you let this code run, your Imp will tweet every 10 seconds! To avoid spamming your Twitter followers and only execute once, comment out this line in the device code:

imp.wakeup(10, readChip189); //Wakeup every 10 second and read data.

Click "Build and Run" in the web IDE. Check the log, if you don't see any errors you have tweeted successfully. Check your Twitter account for the message.

If you do see error messages, check all of your keys and tokens for accuracy, and try again.

Congrats! Your Electric Imp can tweet! Don't forget to send a tweet to @adafruit and @electricimp!

Stay tuned for an upcoming tutorial on building a dual probe thermocouple temperature monitor with LCD readout using the Adafruit Arduino enclosure.