

Electronics for IoT

Smart Sensors

Bernhard E. Boser

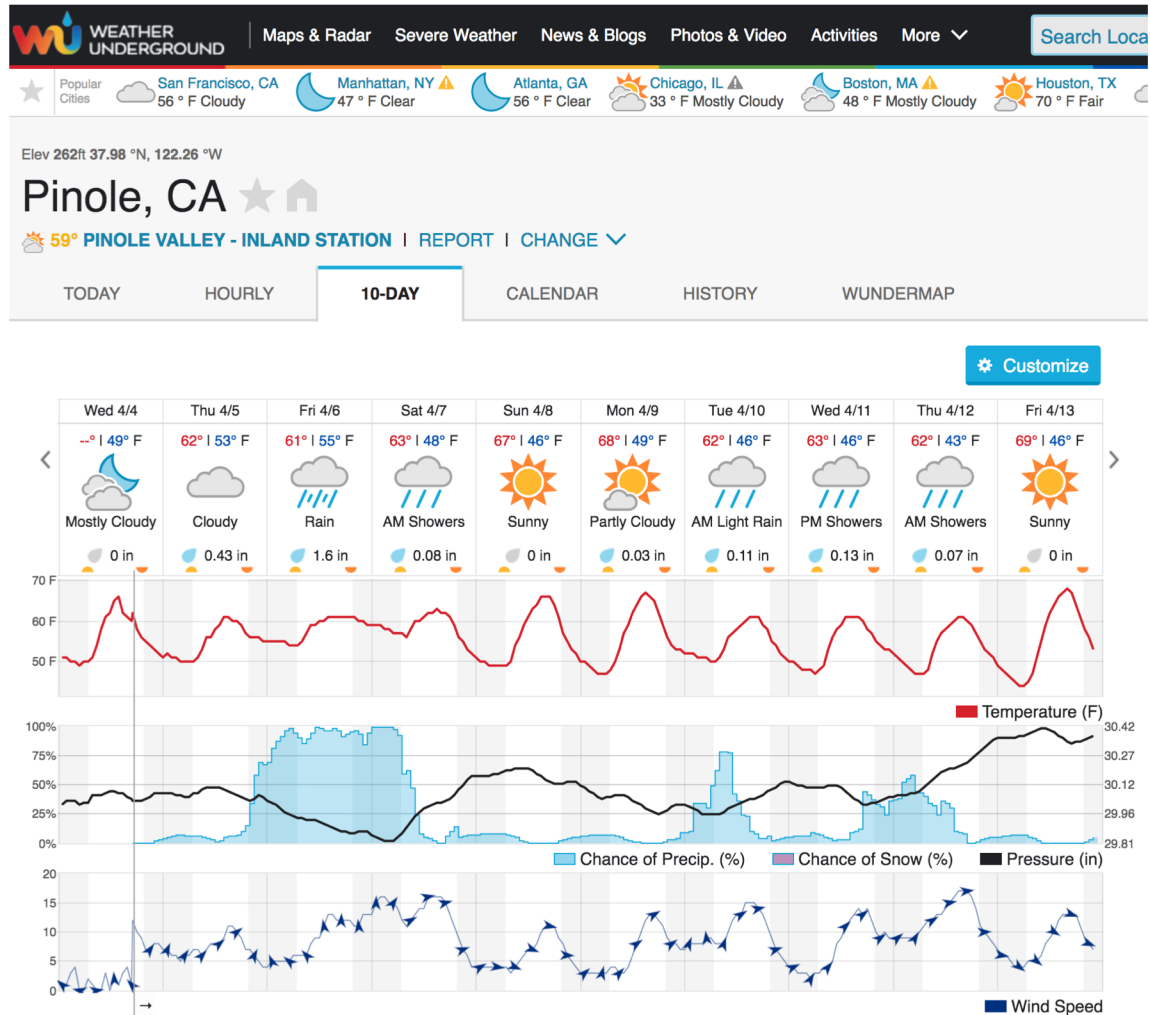
University of California, Berkeley

boser@eecs.berkeley.edu

IoT Services

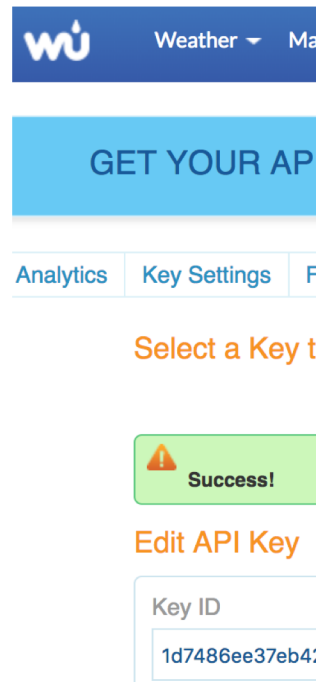
- Current Weather
- Send email
- Send SMS
- ...

Weather



Wunderground

- Weather service
- Create account at <https://www.wunderground.com/weather/api/>
- Copy api key



Documentation

wu Weather ▾ Maps & Radar ▾ Severe Weather ▾ Photos & Video ▾ Community ▾ News ▾ Climate ▾ ★ 🔍 Sign Out

DOCUMENTATION

Analytics | Key Settings | Featured Applications | **Documentation** | Forums

API Table of Contents

- Weather API**
 - WunderMap Layers
 - Radar
 - Satellite
 - Radar + Satellite
 - Data Features
 - alerts
 - almanac
 - astronomy
 - conditions
 - currenthurricane
 - forecast
 - forecast10day
 - geolookup
 - history
 - hourly
 - hourly10day
 - planner
 - rawtide
 - satellite
 - tide

Weather API: Introduction

Getting Started

Before you start using the Weather API, it is important to know that

- Most of the API features require an API key. [Sign up for a key.](#)
- API requests** are made over HTTP. Data features return JSON or XML. WunderMap layer features return image files. This documentation is full of examples of how to use API features. See [code samples for several languages](#), and [user-generated code and libraries](#).
- Multiple API features can be combined into a single HTTP request. This is an easy way to economize your requests. The [Data Features](#) page documents how features can be combined into a single request.
- Per the [Terms of Service](#) the Weather Underground logo must be included with a credit line where Weather Underground data is displayed. Please see individual Logo variations for all acceptable combinations of layout in the [Logo Usage Guide](#).

Example

Current Conditions in US City

```
http://api.wunderground.com/api/1d7...?76/conditions/q/CA/San_Franci  
sco.json
```

Show Response

Page Contents

- Weather API: Introduction
- Getting Started
- Example
- Current Conditions in US City
- Resources
- Community
- Code + Tools
- Images

Request

```
import urequests as requests

res = requests.get("http://api.wunderground.com/api/1d748saf321a4276/conditions/q/CA/San_Francisco.json")

print(res.json()['current_observation'], end='\n\n')

weather = res.json()['current_observation']
print("Temperature", weather['temp_f'], "F")
print("Temperature", weather['temp_c'], "C")
print("Weather    ", weather['weather'])
```

Response

```
[L:/Users/boser/Dropbox/Files/Class/49/esp32> run wunderground.py
{'visibility_mi': '10.0', 'feelslike_string': '56.4 F (13.6 C)', 'precip_1hr_metric': ' 0', 'observation_epoch': '1522888
set': '-0700', 'dewpoint_f': 48, 'wind_string': 'From the South at 1.0 MPH Gusting to 6.0 MPH', 'wind_gust_mph': '6.0', '
ncisco.html', 'history_url': 'http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=KCASANFR131', 'ob_url': 'h
t?query=37.778488,-122.408005', 'observation_time': 'Last Updated on April 4, 5:33 PM PDT', 'pressure_trend': '0', 'wind_
precip_today_string': '0.00 in (0 mm)', 'display_location': {'wmo': '99999', 'elevation': '60.0', 'city': 'San Francisco'
o3166': 'US', 'state': 'CA', 'zip': '94102', 'latitude': '37.77999878', 'country': 'US', 'state_name': 'California', 'ful
me_rfc822': 'Wed, 04 Apr 2018 17:33:56 -0700', 'image': {'title': 'Weather Underground', 'url': 'http://icons.wxug.com/gr
ound.com'}, 'icon_url': 'http://icons.wxug.com/i/c/k/cloudy.gif', 'estimated': {}, 'heat_index_f': 'NA', 'local_tz_short'
, 'visibility_km': '16.1', 'relative_humidity': '72%', 'precip_1hr_string': '0.00 in ( 0 mm)', 'windchill_f': 'NA', 'sola
: '56.4 F (13.6 C)', 'precip_today_metric': '0', 'windchill_c': 'NA', 'local_tz_long': 'America/Los_Angeles', 'station_id
822': 'Wed, 04 Apr 2018 17:33:48 -0700', 'nowcast': '', 'pressure_in': '30.07', 'UV': '1.2', 'dewpoint_string': '48 F (9
{'country': 'US', 'longitude': '-122.41', 'state': 'California', 'city': 'San Francisco', 'elevation': '23 ft', 'country
titude': '37.78'}, 'temp_f': 56.40000000000001, 'temp_c': 13.6, 'precip_today_in': '0.00', 'feelslike_c': '13.6', 'precip
: '56.4', 'weather': 'Overcast'}


Temperature 56.40000000000001 F
Temperature 13.6 C
Weather      Overcast
L:/Users/boser/Dropbox/Files/Class/49/esp32> █
```

Send SMS

- [twilio.com](https://www.twilio.com)

Send SMS from MicroPython

TWILIO FLEX: A CONTACT CENTER PLATFORM WITH FULL-STACK PROGRAMMABILITY (844) 814-4627 [HELP](#) [LOG IN](#)

 [Products](#) [Solutions](#) [Docs & Tools](#) [Customer Stories](#) [Pricing](#) [Talk to Sales](#) [Console](#)

BUILD THE FUTURE OF COMMUNICATIONS

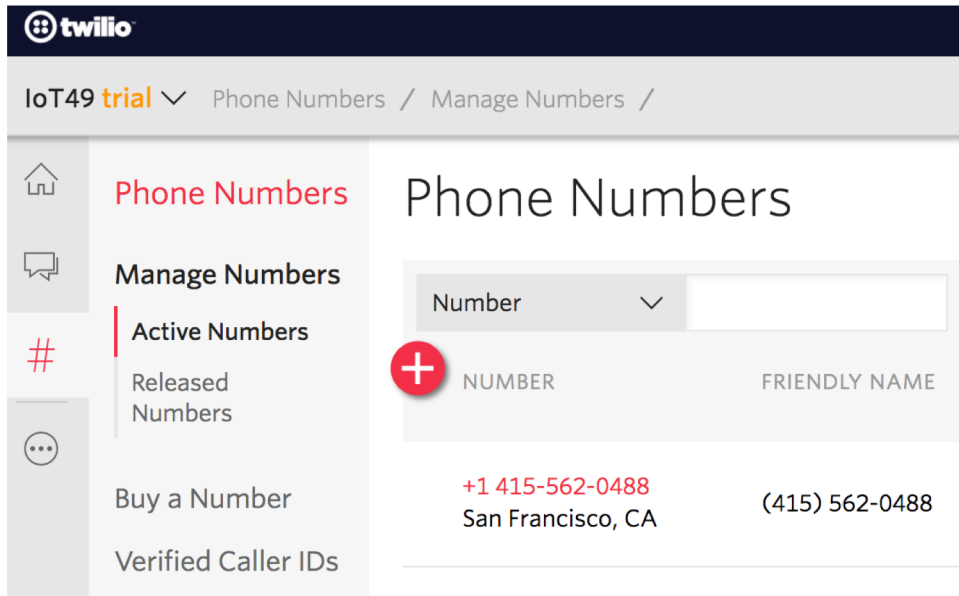
Connect the world with the leading platform for Voice, SMS, and Video.

[Get a free API key >](#)

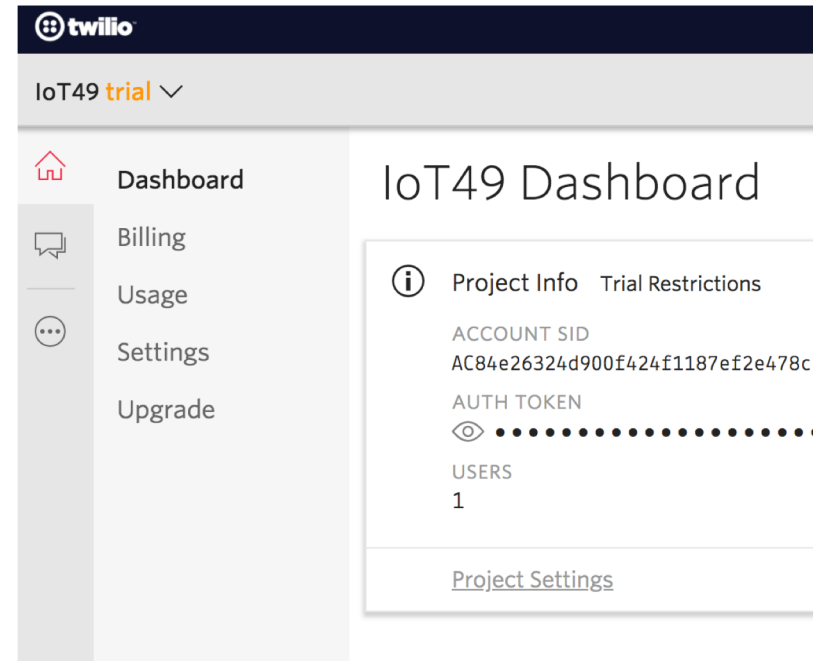
Questions? [Talk to Sales.](#)

Create Account

- Setup free trial
- Get phone number, account SID, auth token



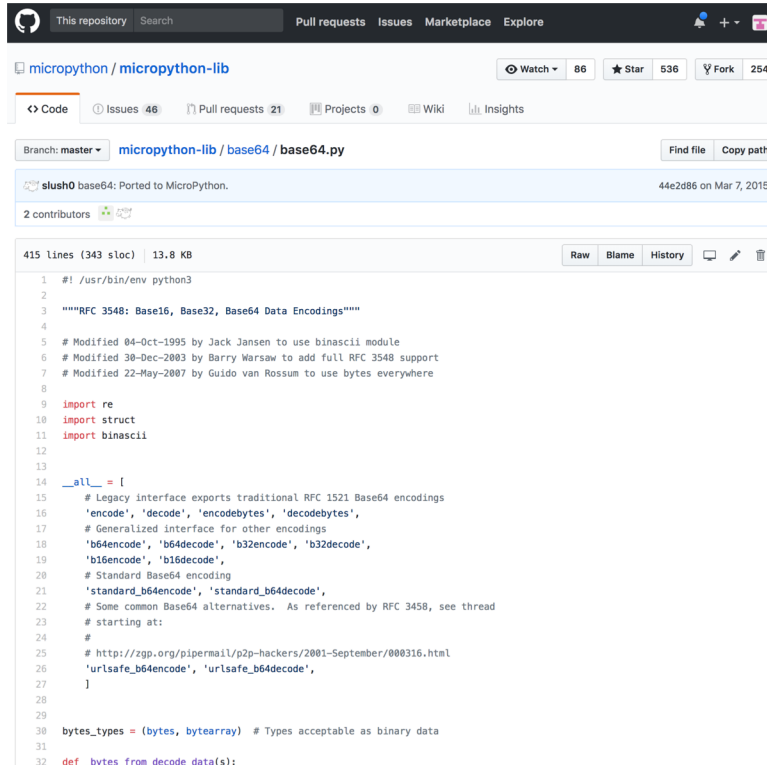
The screenshot shows the Twilio 'Phone Numbers' management interface. The breadcrumb trail is 'IoT49 trial > Phone Numbers / Manage Numbers /'. The left sidebar contains navigation options: 'Phone Numbers' (selected), 'Manage Numbers', 'Active Numbers', 'Released Numbers', 'Buy a Number', and 'Verified Caller IDs'. The main content area is titled 'Phone Numbers' and features a search bar with a dropdown menu labeled 'Number'. Below the search bar is a table with a red plus icon in the top-left corner. The table has two columns: 'NUMBER' and 'FRIENDLY NAME'. One row is visible with the number '+1 415-562-0488' and the friendly name 'San Francisco, CA (415) 562-0488'.



The screenshot shows the Twilio 'IoT49 Dashboard'. The breadcrumb trail is 'IoT49 trial >'. The left sidebar contains navigation options: 'Dashboard' (selected), 'Billing', 'Usage', 'Settings', and 'Upgrade'. The main content area is titled 'IoT49 Dashboard' and contains a section for 'Project Info' with a sub-section for 'Trial Restrictions'. The 'Project Info' section displays the following details: 'ACCOUNT SID' (AC84e26324d900f424f1187ef2e478c), 'AUTH TOKEN' (represented by a series of dots), and 'USERS' (1). There is a 'Project Settings' link at the bottom of the section.

Install base64 library

- <https://github.com/micropython/micropython-lib/blob/master/base64/base64.py>
- Copy to `/flash/lib`
- Add `sys.path.append('/flash/lib')` to `boot.py`



The screenshot shows the GitHub repository page for `micropython-lib`. The file `base64/base64.py` is selected, showing its content. The file is 415 lines long, 13.8 KB, and was last modified on Mar 7, 2015. The code includes a shebang line, a docstring, and several comments about the file's history. It also includes imports for `re`, `struct`, and `binascii`, and defines a `__all__` list of functions and a `bytes_types` tuple.

```
1 #!/usr/bin/env python3
2
3 """RFC 3548: Base16, Base32, Base64 Data Encodings"""
4
5 # Modified 04-Oct-1995 by Jack Jansen to use binascii module
6 # Modified 30-Dec-2003 by Barry Warsaw to add full RFC 3548 support
7 # Modified 22-May-2007 by Guido van Rossum to use bytes everywhere
8
9 import re
10 import struct
11 import binascii
12
13
14 __all__ = [
15     # Legacy interface exports traditional RFC 1521 Base64 encodings
16     'encode', 'decode', 'encodebytes', 'decodebytes',
17     # Generalized interface for other encodings
18     'b64encode', 'b64decode', 'b32encode', 'b32decode',
19     'b16encode', 'b16decode',
20     # Standard Base64 encoding
21     'standard_b64encode', 'standard_b64decode',
22     # Some common Base64 alternatives. As referenced by RFC 3548, see thread
23     # starting at:
24     #
25     # http://zgp.org/pipermail/p2p-hackers/2001-September/008316.html
26     'urlsafe_b64encode', 'urlsafe_b64decode',
27 ]
28
29
30 bytes_types = (bytes, bytearray) # Types acceptable as binary data
31
32 def _bytes_from_decode_data(s):
```

Send SMS

```
import urequests as requests
from base64 import b64encode
import time

"""
Send SMS from Micropython:
0) download and install base64.py
   (https://github.com/micropython/micropython-lib/blob/master/base64/base64.py)
1) create account on twilio.com
2) get 'twilio' phone number (Usage tab) & enter below
3) copy auth_sid and auth_token from API LIVE Credentials (Settings tab) and enter below
4) update msg (below) to what you want to send
"""

# update these (see above)
auth_sid = "AC84e26324d90476324d90476324d9047"
auth_token = "ba0ad421bd69da0b5f041a74d"
to_number = "+1510552350"
from_number = "+1415577028"

# sample message
msg = "{}: Hello from ESP32!".format(time.strftime('%c'))

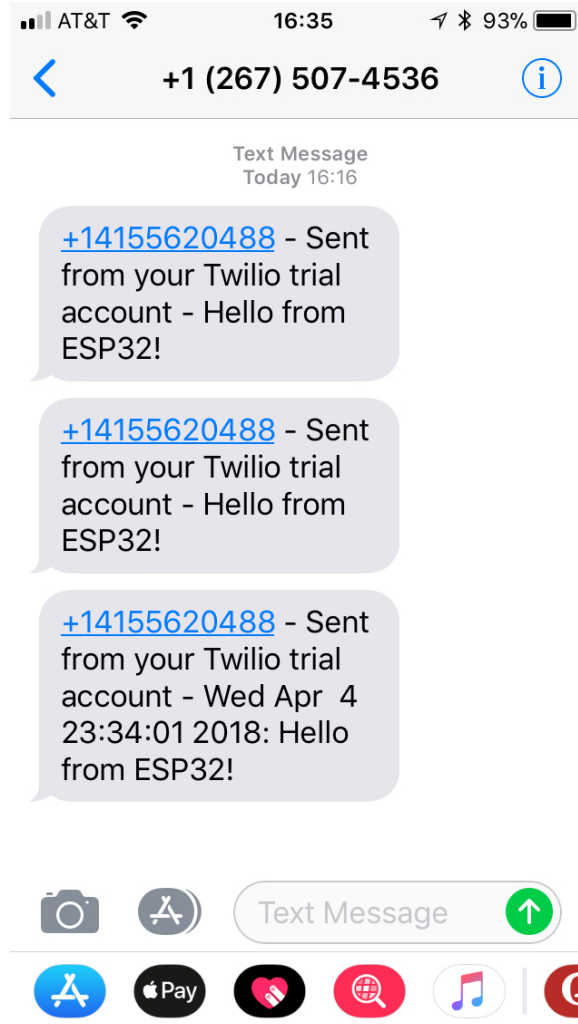
data = "To={}&From={}&Body={}".format(to_number, from_number, msg)

def auth(user, pwd):
    """Twilio uses basic authentication"""
    return (b64encode(
        b':'.join((auth_sid.encode('latin1'),
                  auth_token.encode('latin1'))))
        ).decode('ascii'))

response = requests.post(
    url="https://api.twilio.com/2010-04-01/Accounts/{}/Messages".format(auth_sid),
    data=data,
    headers={
        'Authorization': 'Basic {}'.format(auth(auth_sid, auth_token)),
        'Content-Type': 'application/x-www-form-urlencoded',
    },
)

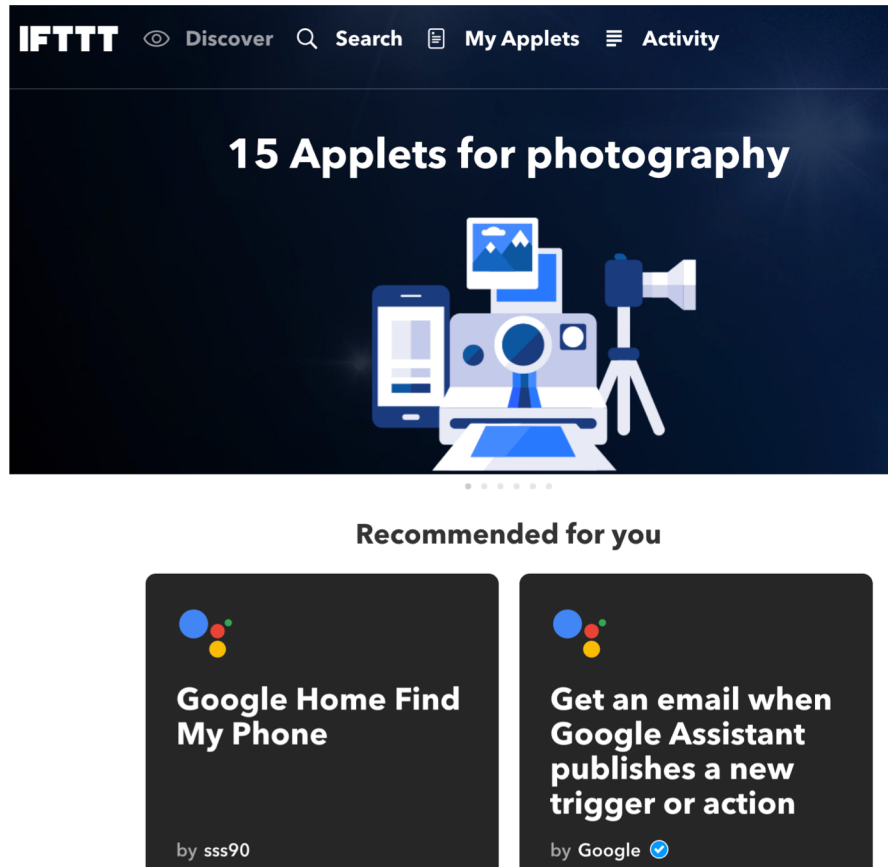
# print success/error messages
print()
print("response.text =", response.text)
```

Result



Other Services

- E.g. IFTTT.com, Echo, Google home, IBM Watson, ...



The screenshot shows the IFTTT website interface. At the top, there is a navigation bar with the IFTTT logo, a 'Discover' button, a search icon, a 'Search' button, a 'My Applets' button, and an 'Activity' button. Below the navigation bar, the main heading reads '15 Applets for photography'. Underneath the heading is an illustration of various photography-related devices: a smartphone, a laptop, a camera on a tripod, and a camera lens. Below the illustration are five small white dots. Underneath the dots is the text 'Recommended for you'. Below this text are two applet cards. The first card is titled 'Google Home Find My Phone' and is attributed to 'by sss90'. The second card is titled 'Get an email when Google Assistant publishes a new trigger or action' and is attributed to 'by Google' with a verified badge.