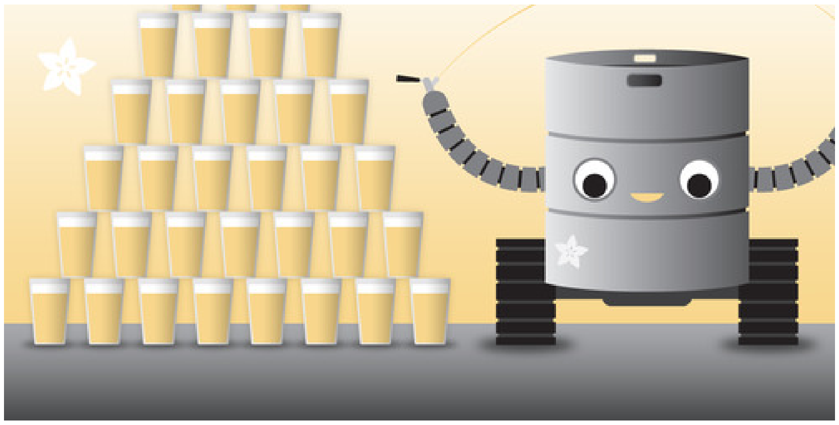
adafruit learning system

Adafruit Kegomatic

Created by Becky Stern



Last updated on 2015-01-01 04:00:12 PM EST

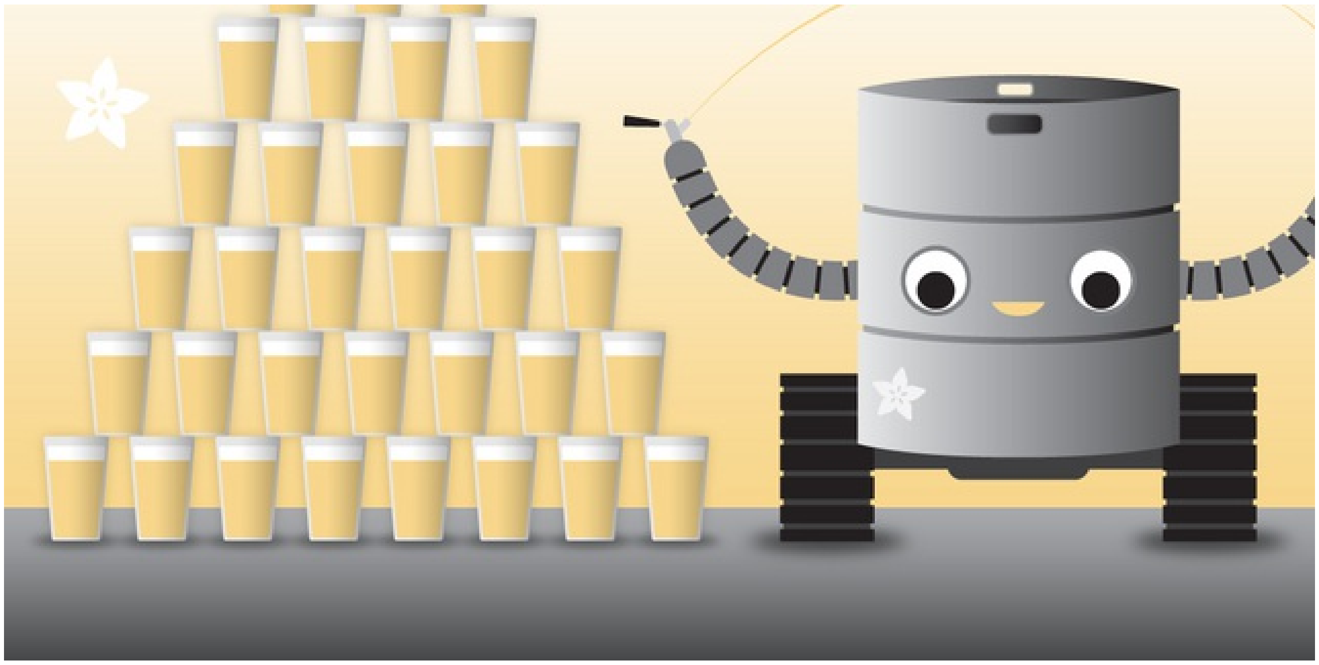
Guide Contents

Overview

The Adafruit Kegomatic is a team project by everyone at Adafruit! We wanted to see how we could augment a kegerator with cool hardware from the Adafruit store and make it active online.

Our first experiment was to hook up a liquid flow meter and have the keg tweet every time someone drinks from it! Follow @AdafruitKegBot (http://adafru.it/cJQ)for updates.

For now the keg has birch beer, but we'll be getting a beer beer kegerator soon and will mod that one up as well!



Bill of Materials

Adafruit hardware:

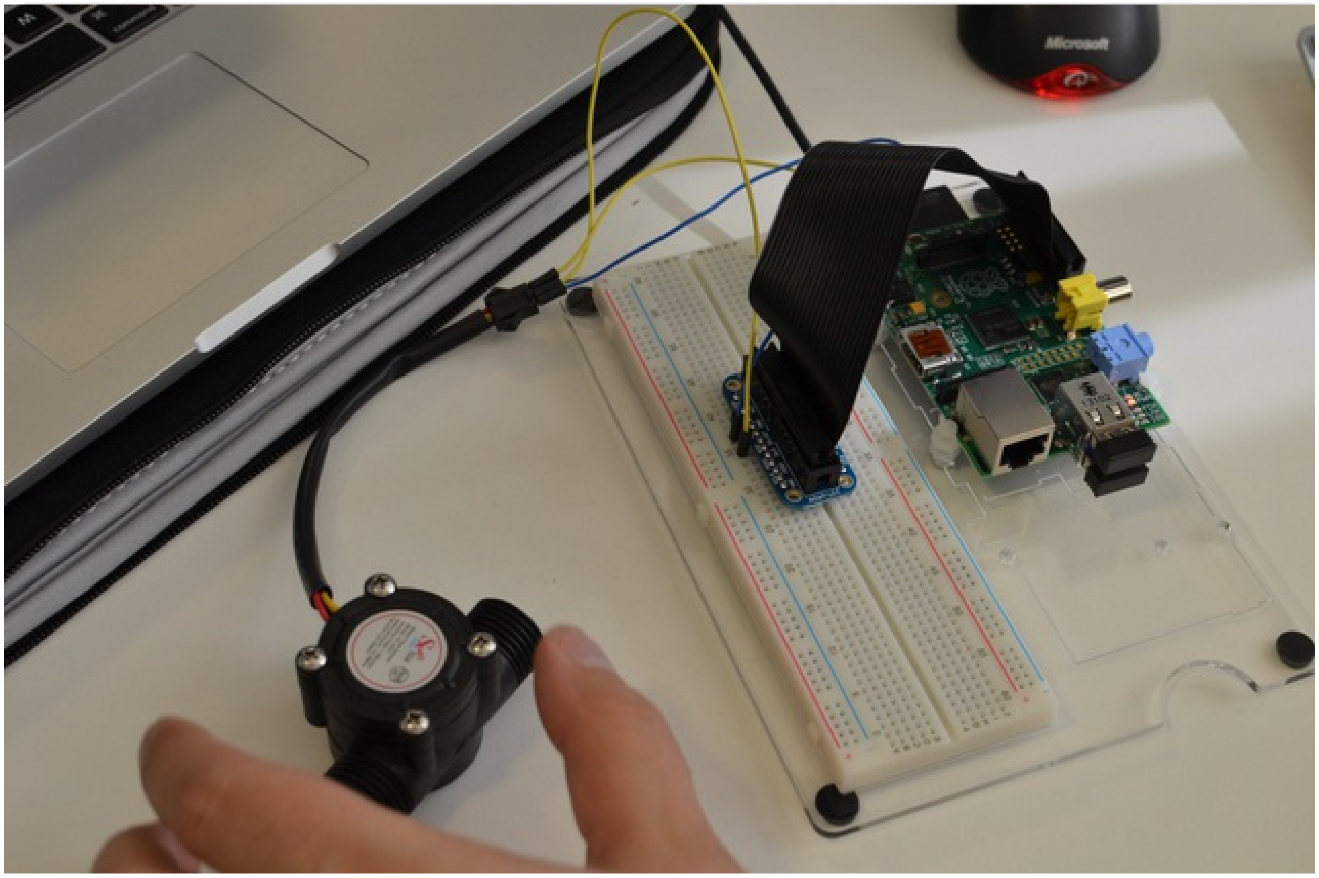
- Raspberry Pi (http://adafru.it/998)
- Liquid flow meter (http://adafru.it/828) (plastic not brass, which could make the tasty beverage taste funny)
- HDMI monitor (http://adafru.it/1033)
- Mini wifi module (http://adafru.it/814)
- Mini wireless keyboard (http://adafru.it/922)

Other hardware:

- Kegco kegerator (http://adafru.it/cJS) or modded fridge
- keg of birch beer (better for testing than alcoholic beer!)
- 2x liquid lines (one to mod and a backup)
- small CO2 tank with regulator and air hose
- teflon tape
- 2x barb -> threaded connector that matches interior diameter of the liquid line and threads on the flow meter
- 2x hose clamps

Thanks to Snoegoer Co. in Minnesota for making and donating this sweet tap, we'll be putting it on our next kegerator!

Prototype Circuit



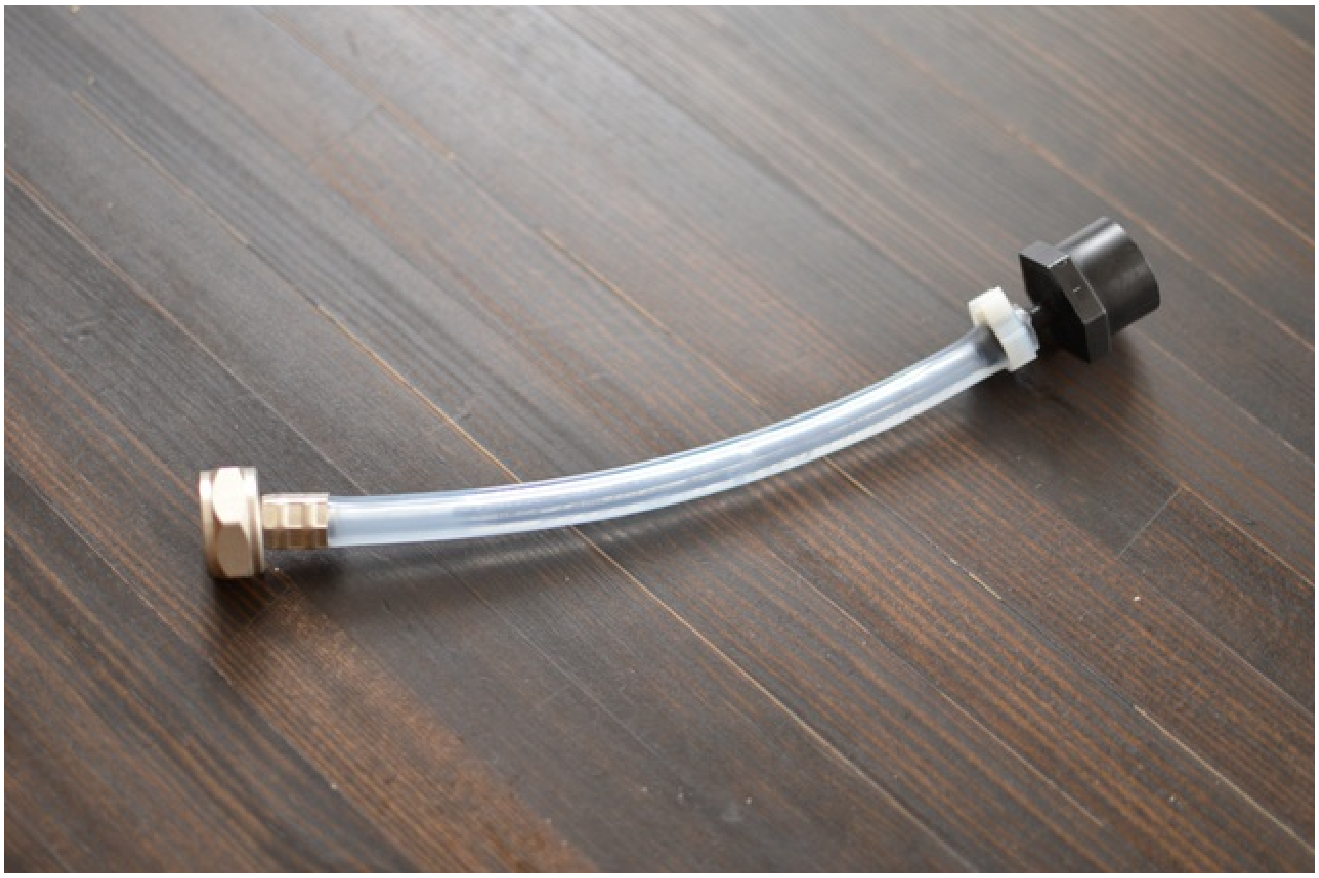
You can hook the sensor up to an Arduino using our sample code (http://adafru.it/cJV), or read on for the Raspberry Pi setup.

Test the flow meter by blowing into it (there's an arrow on the meter to show you the correct flow direction).

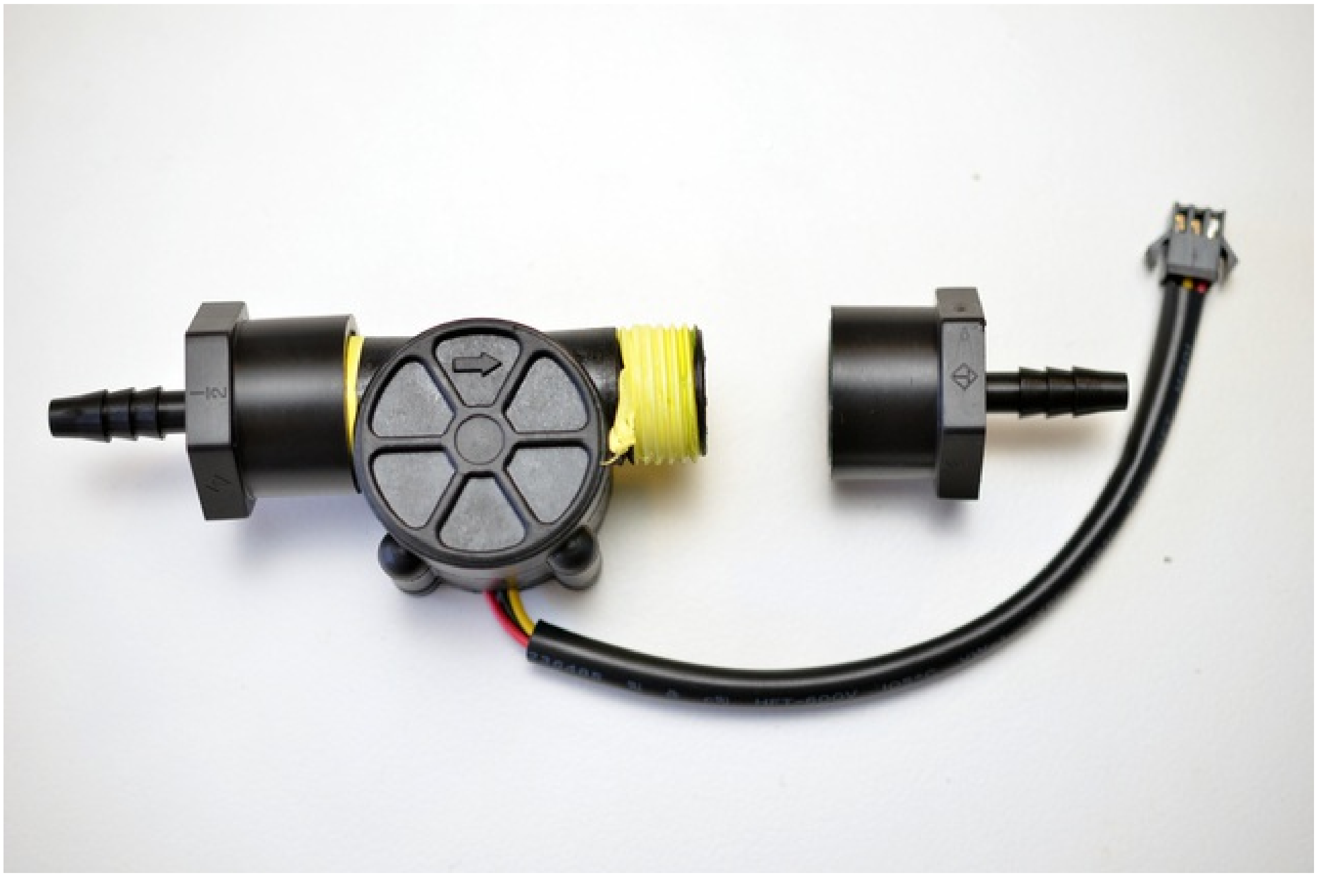
When it's working to your satisfaction, flush it with soapy water to clean it before installing in your kegbot.

Install Flow Meter

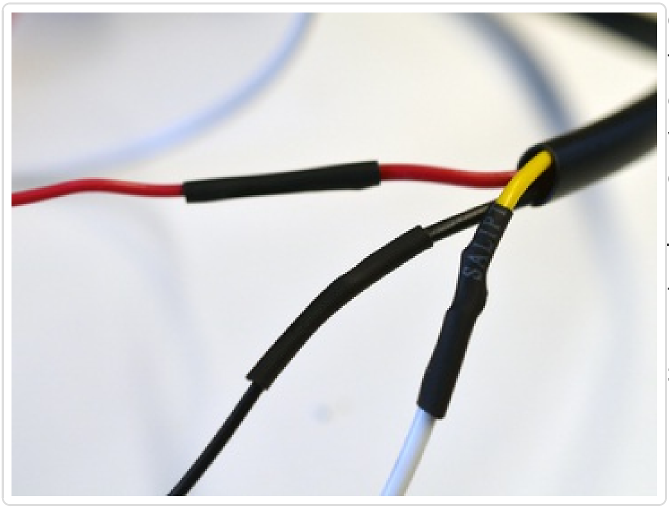
It helps to have two liquid tubes, in case you break one but still want to access your beer. Cut the liquid line with a pair of sharp scissors.



Insert the barbed connectors into the cut ends of the tubes and secure with hose clamps.

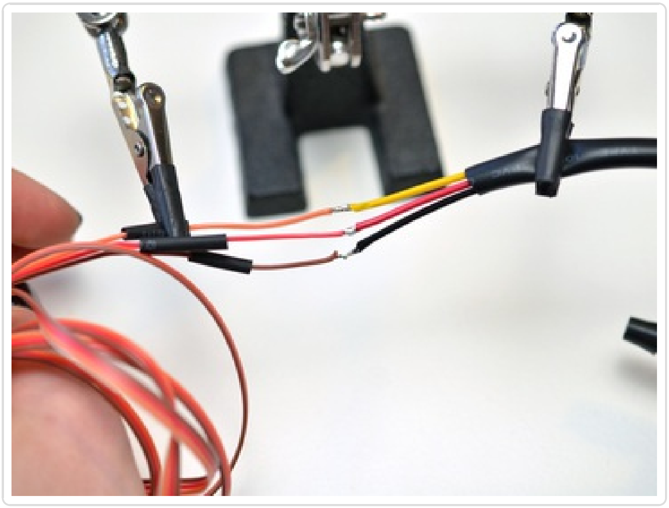


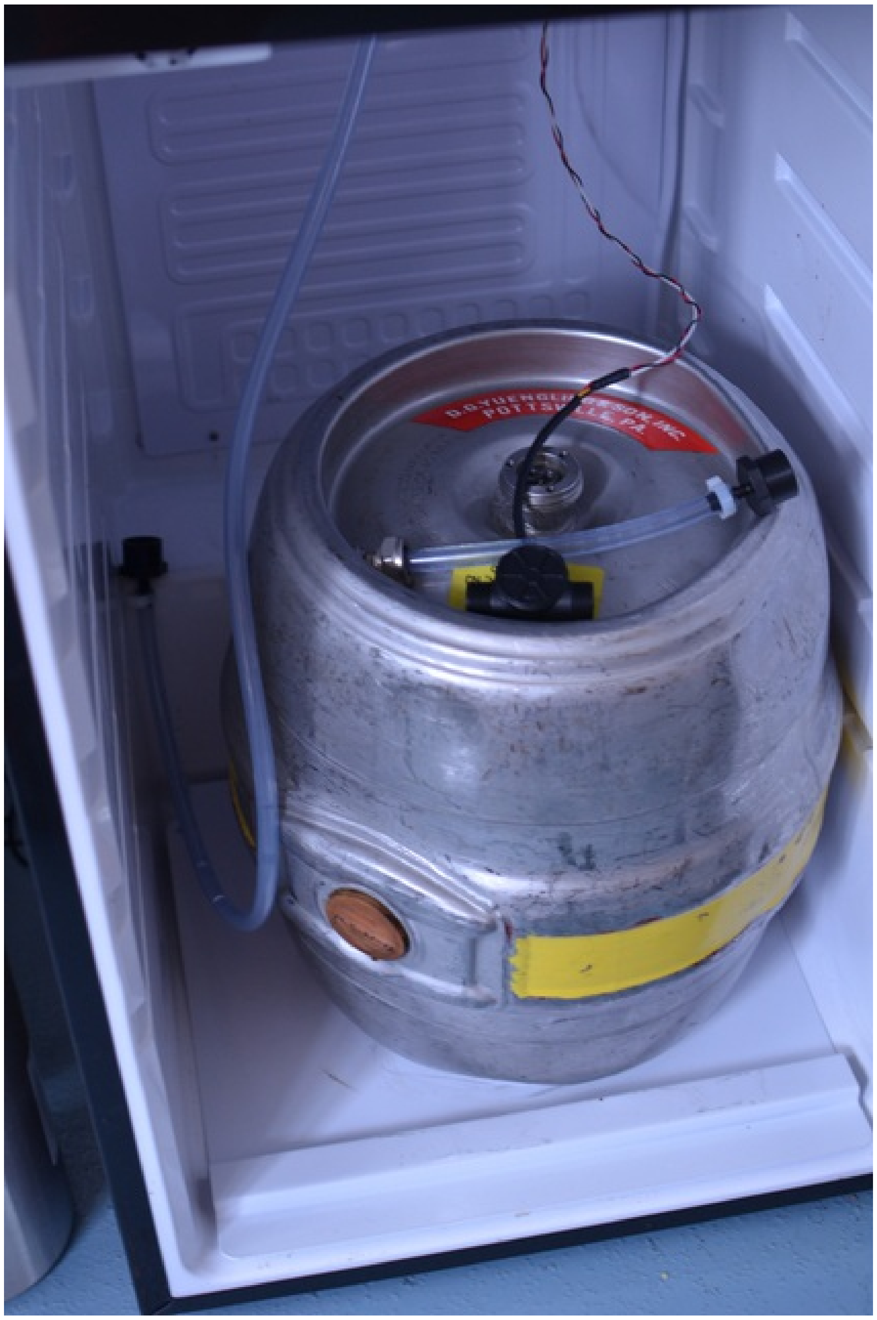
Thread the flow meter into the connectors (don't forget the teflon tape)-- now the liquid will go through the flow meter!



Cut off the flow meter's connector, strip the three wires inside, and solder on a long extension for each wire, enough to go from your keg tube, outside the keg, to the output display. We made ours about four feet long.

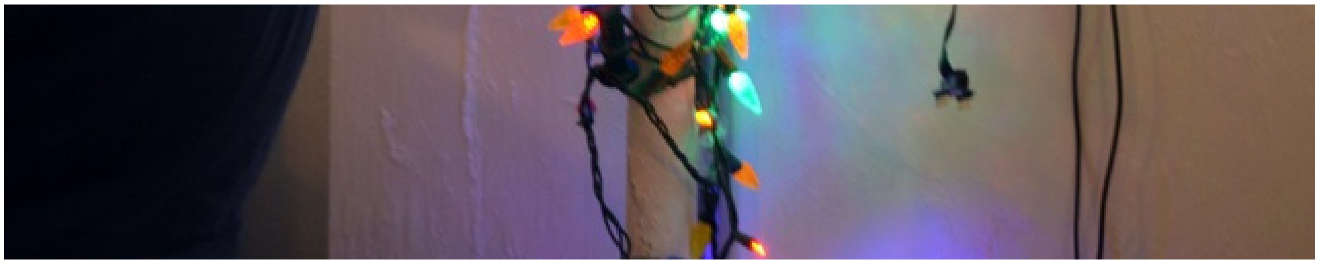
To connect each wire, tin the stripped ends, then position the two wire ends together and remelt the solder. Slide on pieces of heat shrink tubing to insulate the solder joints.





Run the extension wires out the back of the fridge so they can be hooked up to the RasPi/Arduino/system of your choice.

Raspberry Pi Code



The code for this project is on GitHub.

https://github.com/adafruit/Kegomatic (http://adafru.it/e84)

After you've set up your pi, log in or ssh in, and get ready to set up some stuff. First, we'll download the kegbot code:

```
git clone https://github.com/adafruit/Kegomatic.git
```

We'll need to install Python's setup tools in order to make it easier to install the pre-requisites for our Twitter code.

```
sudo -i
wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - | python
```

Next, we will install a whole bunch of pre-requisites. Execute these one line at a time and look for errors. If you don't see any, you're good to go.

```
sudo easy_install twitter
sudo easy_install simplejson
sudo easy_install httplib2
sudo easy_install python-oauth2
```

And that's it for setup... now the code should work... let's take a look at what it does!

```
#!/usr/bin/python
import os
import time
import math
import pygame, sys
from pygame.locals import *
import RPi.GPIO as GPIO
```

```
from twitter import *
```

The above lines are all the imports we need... we need the twitter API for twittering, the GPIO library so we can use the GPIO pins, pygame so we can make this a fun windowed application, math so we can do some math stuff, time so we can do accurate timing, and os so we can run things at the os level.

Next, we'll need to initialize the twitter account:

```
t = Twitter( auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET, CONSUMER_KEY, CONSUMER_SECRET) )
```

First, go set up a Twitter app here (http://adafru.it/ejM). You will need to be logged in to Twitter from whatever account you would like to do the kegomatic tweeting.

Then, come back to the code, and replace OAUTH_TOKEN, OAUTH_SECRET, CONSUMER_KEY, and CONSUMER_SECRET with the actual values from your twitter account app page.

After that, we will need to initialize the GPIO pins...

```
boardRevision = GPIO.RPI_REVISION
GPIO.setmode(GPIO.BCM) # use real GPIO numbering
GPIO.setup(22,GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

The next several lines are there to set up pygame, the windowed context, and various global variables we will need throughout the program...

```
# set up pygame
pygame.init()

# set up the window
VIEW_WIDTH = 0
VIEW_HEIGHT = 0
pygame.display.set_caption('KEGBOT')

# set up the flow meter
pouring = False
lastPinState = False
pinState = 0
lastPinChange = int(time.time() * 1000)
pourStart = 0
pinChange = lastPinChange
pinDelta = 0
```

```
hertz = 0
flow = 0
litersPoured = 0
pintsPoured = 0
tweet = ''

# set up the colors
BLACK = (0,0,0)
WHITE = (255,255,255)

windowSurface = pygame.display.set_mode((VIEW_WIDTH,VIEW_HEIGHT), FULLSCREEN, 32)
FONTSIZE = 48
LINEHEIGHT = 52
basicFont = pygame.font.SysFont(None, FONTSIZE)
```

Next comes the renderThings() function, which draws all of our updated data to the screen...

```
def renderThings(lastPinChange, pinChange, pinDelta, hertz, flow, pintsPoured, pouring, pourStart, twee
  # Clear the screen
  windowSurface.fill(BLACK)

  # Draw LastPinChange
  text = basicFont.render('Last Pin Change: '+time.strftime('%H:%M:%S', time.localtime(lastPinChange/
  textRect = text.get_rect()
  windowSurface.blit(text, (40,1*LINEHEIGHT))

  # Draw PinChange
  text = basicFont.render('Pin Change: '+time.strftime('%H:%M:%S', time.localtime(pinChange/1000)),
  textRect = text.get_rect()
  windowSurface.blit(text, (40,2*LINEHEIGHT))

  # Draw PinDelta
  text = basicFont.render('Pin Delta: '+str(pinDelta) + ' ms', True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,3*LINEHEIGHT))

  # Draw hertz
  text = basicFont.render('Hertz: '+str(hertz) + 'Hz', True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,4*LINEHEIGHT))

  # Draw instantaneous speed
  text = basicFont.render('Flow: '+str(flow) + ' L/sec', True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,5*LINEHEIGHT))
  # Draw Liters Poured
  text = basicFont.render('Pints Poured: '+str(pintsPoured) + ' pints', True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,6*LINEHEIGHT))

  # Draw Pouring
  text = basicFont.render('Pouring: '+str(pouring), True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,7*LINEHEIGHT))

  # Draw Pour Start
  text = basicFont.render('Last Pour Started At: '+time.strftime('%H:%M:%S', time.localtime(pourStart/1
  textRect = text.get_rect()
  windowSurface.blit(text, (40,8*LINEHEIGHT))

  # Draw Tweet
  text = basicFont.render('Tweet: '+str(tweet), True, WHITE, BLACK)
  textRect = text.get_rect()
  windowSurface.blit(text, (40,9*LINEHEIGHT))

  # Display everything
  pygame.display.flip()
```

Now we begin the main loop, which will loop forever (until we quit the program). The first thing we need to do at the beginning of the loop every time is to figure out how much time has passed since the last time we ran the loop. To do that, we get need to get the current time. We also need to know if the pin is set high or low right now, so we can start counting the time between pulses of the flow meter.

```
# main loop
while True:
  currentTime = int(time.time() * 1000)
  if GPIO.input(22):
    pinState = True
  else:
    pinState = False
```

We also have a small amount of keyboard handling code, so that the user can press the escape key to exit the program.

```
# Handle keyboard events
  for event in pygame.event.get():
    if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):
      pygame.quit()
      sys.exit()
```

Now, this next if statement is a doozie, so bear with me...

If the pin state is high and is not the same as it was last time through the loop (so that the pin staying high doesn't result in a steady stream of clicks as the loop runs over and over) then we enter pouring mode. While in pouring mode, we figure out how much time has passed between this click and the last click of the flow meter, and use that to calculate the flow. Once we have the number of milliseconds between the last click and now, we can figure out the hertz of the meter, divide that by the rate expected of the flow meter to get 1 liter per second of flow, and convert the result from liters to pints. This (very small) number represents the amount of beer that has been poured during this time through the loop.

```
# If we have changed pin states low to high...
  if(pinState != lastPinState and pinState == True):
    if(pouring == False):
      pourStart = currentTime
    pouring = True
    # get the current time
    pinChange = currentTime
    pinDelta = pinChange - lastPinChange
    if (pinDelta < 1000):
      # calculate the instantaneous speed
      hertz = 1000.0000 / pinDelta
      flow = hertz / (60 * 7.5) # L/s
      litersPoured += flow * (pinDelta / 1000.0000)
      pintsPoured = litersPoured * 2.11338
```

This next if statement is a little shorter... it says that if we are pouring, and we notice that our sensor has been idle for more than 3 seconds, then we can assume that we are no longer pouring, and it's time to calculate how big the pour was and tweet it. After we do that, we should reset the amount poured so the next guy can begin his pour.

```
if (pouring == True and pinState == lastPinState and (currentTime - lastPinChange) > 3000):
    # set pouring back to false, tweet the current amt poured, and reset everything
    pouring = False
    if (pintsPoured > 0.1):
      pourTime = int((currentTime - pourStart)/1000) - 3
      tweet = 'Someone just poured ' + str(round(pintsPoured,2)) + ' pints of root beer in ' + str(pourTin
      t.statuses.update(status=tweet)
      litersPoured = 0
      pintsPoured = 0
```

Finally, we must draw everything to the screen, and update the time variables so that we can accurately measure time the next time we go through the loop.

```
 renderThings(lastPinChange, pinChange, pinDelta, hertz, flow, pintsPoured, pouring, pourStart, tweet,
 lastPinChange = pinChange
 lastPinState = pinState
```

To run this code, run the following:

```
sudo python kegbot.py
```

That should bring up a window with the bot statistics, and then you can start pouring and tweeting!