TRENDING Using the BME280 I2C Temperature and Pressure Sensor in Python

Search...

HOME CATEGORIES BERRYCLIP BUY TOOLS TUTORIALS & HELP CONTACT US SITE MAP

YOU ARE AT: Home » Hardware » Reading Analogue Sensors With One GPIO Pin

Reading Analogue Sensors With One GPIO Pin

30

BY MATT ON AUGUST 13, 2012 HARDWARE, PYTHON, TUTORIALS & HELP

Unlike some other devices the Raspberry Pi does not have any analogue inputs. All 17 of its GPIO pins are digital. They can output high and low levels or read high and low levels. This is great for sensors that provide a digital input to the Pi but not so great if you want to use a sensor that doesn't.

For sensors that act as a variable resistor such as LDRs (Light Dependent Resistors) or thermistors (temperature sensors) there is a simple solution. It allows you to measure a number of levels using a single GPIO pin. In the case of a light sensor this allows you to measure different light levels.

It uses a basic "RC" charging circuit (Wikipedia Article) which is often used as an introduction to electronics. In this circuit you place a Resistor in series with a Capacitor. When a voltage is applied across these components the voltage across the capacitor rises. The time it takes for the voltage to reach 63% of the maximum is equal to the resistance multiplied by the capacitance. When using a Light Dependent resistor this time will be proportional to the light level. This time is called the time constant :

```
t = RC

where t is time,
R is resistance (ohms)
and C is capacitance (farads)
```

So the trick is to time how long it takes a point in the circuit the reach a voltage that is great enough to register as a "High" on a GPIO pin. This voltage is approximatey 2 volts, which is close enough to 63% of 3.3V for my liking. So the time it takes the circuit to change a GPIO input from Low to High is equal to 't'.

With a 10Kohm resistor and a 1uF capacitor t is equal to 10 milliseconds. In the dark our LDR may have a resistance of 1Mohm which would give a time of 1 second. You can calculate other values using an online time constant calculator.

In order to guarantee there is always some resistance between 3.3V and the GPIO pin I inserted a 2.2Kohm resistor in series with the LDR.

Here is the circuit :

RECENT POPULAR NEWS

NOVEMBER 30, 2015
Raspberry Pi Family Photo by RasPi.TV

NOVEMBER 26, 2015
Introducing the Raspberry Pi Ze

NOVEMBER 23, 2015
The Official Raspberry Pi Projec Book

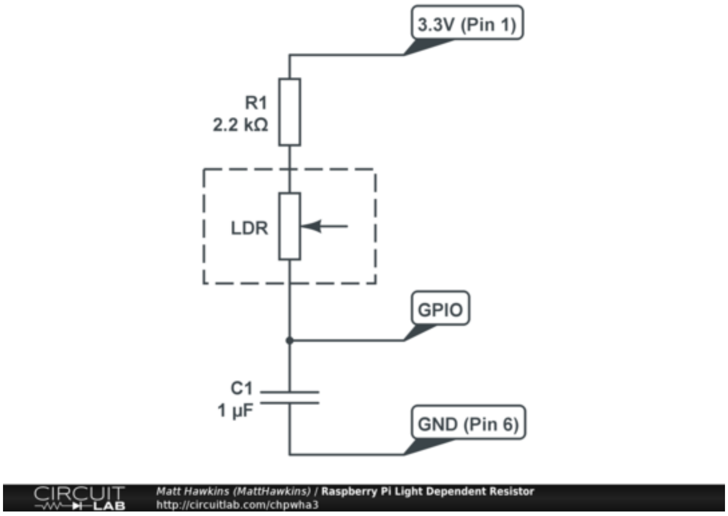
NOVEMBER 22, 2015
Raspberry Pi microSD Card Sho out

OCTOBER 12, 2015
How To Autorun A Python Scrip Boot Using systemd

Search ... Search

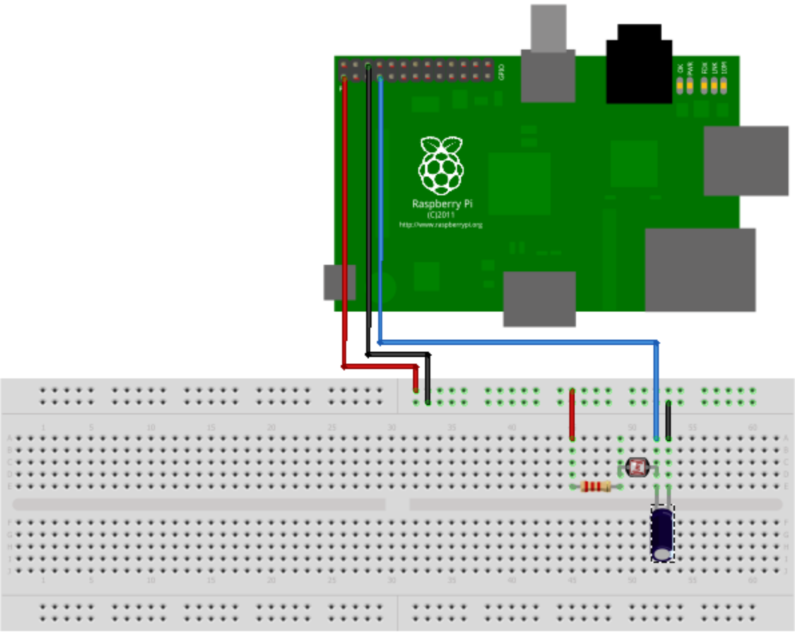
CATEGORIES

- 3D Printing
- Add-ons
- BBC Micro:bit
- BerryClip
- Books
- Camera Module
- Cases
- Events



Matt Hawkins (MattHawkins) / Raspberry Pi Light Dependent Resistor
http://circuitlab.com/chpwha3

Here is the circuit implemented on a breadboard :



Theory

Here is the sequence of events :

- Set the GPIO pin as an output and set it Low. This discharges any charge in the capacitor and ensures that both sides of the capacitor are 0V.
- Set the GPIO pin as an input. This starts a flow of current through the resistors and through the capacitor to ground. The voltage across the capacitor starts to rise. The time it takes is proportional to the resistance of the LDR.
- Monitor the GPIO pin and read its value. Increment a counter while we wait.
- At some point the capacitor voltage will increase enough to be considered as a High by the GPIO pin (approx 2v). The time taken is proportional to the light level seen by the LDR.
- Set the GPIO pin as an output and repeat the process as required.

Python Code

Here is some code that will print out the number of loops it takes for the capacitor to charge.

```
#!/usr/local/bin/python

# Reading an analogue sensor with
# a single GPIO pin
```

General
Hardware
I2C
Infographics
Minecraft
Model A+
Model B+
News
Pi 2 Model B
Pi Models
Pi Zero
Power
Programming
Python
Raspbian
Robotics
Sensors
Software
SPI
Tutorials & Help

TAGS

3D Printing 44780 audio avconv battery berryclip Birthday bluetooth cambridge camera CamJam cpu DigiMakers display games GPIO i2c Kickstarter lcd LED Linux Minecraft Mod Model B mp4 pcb photography photos Pi-Lite portable power python Raspberry Pi Bootcamp raspbian ravpower revision sd card security sensor soldering SPI temperature ultrasonic usb video

```
# Author : Matt Hawkins
# Distribution : Raspbian
# Python : 2.7
# GPIO   : RPi.GPIO v3.1.0a

import RPi.GPIO as GPIO, time

# Tell the GPIO library to use
# Broadcom GPIO references
GPIO.setmode(GPIO.BCM)

# Define function to measure charge time
def RCtime (PiPin):
  measurement = 0
  # Discharge capacitor
  GPIO.setup(PiPin, GPIO.OUT)
  GPIO.output(PiPin, GPIO.LOW)
  time.sleep(0.1)

  GPIO.setup(PiPin, GPIO.IN)
  # Count loops until voltage across
  # capacitor reads high on GPIO
  while (GPIO.input(PiPin) == GPIO.LOW):
    measurement += 1

  return measurement

# Main program loop
while True:
  print RCtime(4) # Measure timing using GPIO4
```

Accuracy

Given we only want to spot different light levels we don't really need to know the resistance of the LDR or the exact time it takes to charge the capacitor. You can do the maths if you want to but I just needed to get a measurement and compare it to some known values. Seconds or Python loop counts, it doesn't matter.

Python is an interpreted language which means the timing of loops is always going to be affected by the operating system performing other background tasks. This will affect the count loop in our example.

Practical Uses

In a more useful application you can call "RCtime" when you need it to get a count value. Your code can then perform other tasks based on the value of the count, perhaps comparing to values you have measured previously.

When I built my test circuit it was sat in front on my TV. I could see the count number change as the lighting level changed on the TV. It's so simple you really need to just give it a try!

Acknowlegments

This article was inspired by the excellent article on Adafruit.com.

SHARE.

‹ PREVIOUS ARTICLE
20×4 LCD Module Control Using Python

NEXT ARTICLE ›
Sending SMS Text Messages Using Python

RELATED POSTS

RASPBERRY PI RELATED

- Averageman Vs Pi
- MattsBits – Pi Resources
- Official RaspBerry Pi Site
- Raspberry Pi Pod
- RasPi.tv
- Raspihub.com
- RaspTut
- Rastrack.co.uk
- RPi School

TECH RESOURCES

- How2Code
- Technology Spy

ARCHIVES

Archives Select Month ▾

OCTOBER 23, 2016 🗨 2

Waterproof Ultrasonic Distance Measuring Module

JULY 21, 2016 🗨 9

Using the BME280 I2C Temperature and Pressure Sensor in Python

JULY 21, 2016 🗨 2

12 inch GPIO Zero Ruler from RasPi.TV

MAY 1, 2016 🗨 1

Using the BBC Microbit with the Raspberry Pi

APRIL 28, 2016 🗨 0

Kano Computer Kit First Impressions

APRIL 11, 2016 🗨 3

Write SD Card Images Using Etcher On Windows, Linux & Mac

MARCH 24, 2016 🗨 2

How to Change Your Character Skin In Minecraft Pi Edition

MARCH 10, 2016 🗨 7

7-segment Display Modules and the Raspberry Pi

MARCH 4, 2016 🗨 0

Ultimate Heatsink for the Raspberry Pi 3

30 COMMENTS

ADAFRUIT on AUGUST 13, 2012 9:58 PM

nice 🙂 thanks for the link too! you might want to check out fritzing parts, we just added a bunch for raspberry pi!

https://github.com/adafruit/Fritzing-Library

REPLY ›

MATT on AUGUST 13, 2012 10:45 PM

Excellent. Just imported the library into my Fritzing installation. I like the parts for the PIR and membrane keypad. Will definitely be able to make use of the PIR part in future projects. I like the Raspberry Pi part but wish the pin labels were GPIO numbers rather than the alternative function labels.

REPLY ›

GRAHAM T. on SEPTEMBER 1, 2012 11:27 PM

Great article Matt, very useful for a project I'm considering. Keep up the good work!
G.

REPLY ›

RICHARD on OCTOBER 4, 2012 9:55 PM

Cool stuff, good idea!
But I get wildly fluctuating readings that don't seem to change too much when I hold my hand over the sensor.

REPLY ›

ALEXK on OCTOBER 7, 2012 12:35 AM

Thanks for the nice article.

I suppose the same method will work with an analog temp sensor.

Thanks,
alexk

REPLY ›

RICK SEIDEN on OCTOBER 11, 2012 10:21 PM

I made some small changes to the code:

```
from datetime import datetime


GPIO.setup(PiPin, GPIO.IN)
# Count loops until voltage across
# capacitor reads high on GPIO
startTime=dateTime.now()
while (GPIO.input(PiPin) == GPIO.LOW):
measurement += 1
endTime=dateTime.now()
diffTime=endTime-startTime
return str(diffTime.total_seconds()/0.001)
```

I figured that would give me the time in diffTime, and I know the capacitance is 1uF, so I should be returning the resistance. I put a static 22ohm resister in the circuit as a test, and I get nothing but zeros. If I crank it up to

```
 return str(diffTime.total_seconds()/0.000001)
```

I start getting values, but in the range of 120, not 220 like I'd expect. I know timing isn't going to be perfect, but I'd expect it to be more accurate than to be always about 100 ohms off. Am I doing something wrong?

Thanks
Rick

REPLY ›

MATT on OCTOBER 15, 2012 12:06 AM

In your example, with a 1uF capacitor, and multiplying the time by 1000000 I would expect the result to be 22. As it is 120 I think this just shows a delay in measuring the small time interval.

Because there is so much going on within the Linux operating system you are probably just seeing the inaccuracy of the technique. I just count loops because it gives me a number to compare without worrying about the exact meaning of the number. With the same light level I see at least a 10%-20% jumping about in the measurement. If anything loads the system at the same time this can be even worse.

Stick in a 20K resistor and your results will probably be a bit closer to the predicted values.

REPLY ›

Pingback: Playing with GPIO and sensors on OpenWrt | #labOS

FMBOY on APRIL 28, 2014 10:50 AM

Hi,

Interesting article. I wonder if anyone used this approach to measure distance with sharp IR sensor on Pi?
I get number of counts and from datasheet I can see that it takes ~38 msec to get one sample. So time is known, but I can't get distance with that input.

Any help or suggestion is appreciated.
thanks

REPLY ›

PAUL on JUNE 5, 2014 12:40 PM

Seems like a "dangerous" solution to me. Can the GPIO pin handle the inrush current created when discharging the capacitor which would have to rather large in order to get any precision in time measurement? I wouldn't try this myself without some interfacing electronics between the Pi and the capacitor.

REPLY ›

PAUL (WHAT A COINCIDENCE) on FEBRUARY 5, 2016 5:34 PM

I was wondering about this too. And I don't know the exact answer. But if I was about to implement this, I would probably try to figure out, how much inrush current the PI can handle. By the way, I think you can spent a PWM output in order to reduce the inrush current peak. Beginning with a very high duty cycle and reducing it continuously to a steady ground. I also suggest pigpio for things like PWM output, if it has to be accurate. With pigpio you can also measure the time between two edges much more precisely than with RPi.GPIO. For all those, who complain about this method being inaccurate. But that would certainly be a bit more time consuming.

REPLY ›

Pingback: Why i chose Raspberry instead of Arduino Yun and Spark-Core in the end | Making connected stuff

RPA on OCTOBER 5, 2014 1:47 PM

Thanks for your tutorial! Is a 3.3 uF capacitor ok to use for the circuit? I don't have a 1 uF capacitor available yet. I hope it does not cause something like short circuit…

REPLY ›

MATT on OCTOBER 6, 2014 12:11 PM

That would be OK. It will just change the timing calculations as the time constant will be based on 3.3uF rather than 1uF.

REPLY ›

ANGGA on NOVEMBER 22, 2014 3:52 AM

Can't using ldr more than 1, for example 8 ldr 8 GPIO pins?

REPLY ›

Pingback: Experiment: use an arduino as a slave to your raspberry pi | Project Pi

ST on JANUARY 28, 2015 1:12 AM

Great tutorial. I used this technique to monitor two different LDRs to notify me if I had accidentally left lights on in two outbuildings. Just counted number of loops; low number meant light was on, high number – lights off. If either light was on, I put power out on one of the GPIO pins to turn on an LED in the main house. Slick!

REPLY ›

PI2015 on MAY 5, 2015 12:00 AM

Great Article!

I have been working off of the same Adafruit article for a Light Detection system of my own.

I have one question id like to ask. How would one go about scaling this up? How do you go about adding more light sensors in to allow for say, a 3×3 grid? And beyond?

Apologies for the entry level questions!

REPLY ›

MATT on MAY 6, 2015 11:55 AM

To be honest this technique probably isn't the way to go if you've got lots of sensors. You would have to poll each one and wait for the result and this would start to add a larger delay to your script. I would add a MCP3008 ADC and read the analogue outputs of the light sensor. It would be quicker and more accurate.

REPLY ›

AYA on MAY 30, 2015 8:21 PM

“With a 10Kohm resistor and a 1uF capacitor t is equal to 1 millisecond. In the dark the LDR may have a resistance of 10Mohm which would give a time of 1 second.”

Don’t you mean :
“With a (1Kohm) resistor and a 1uF capacitor t is equal to 1 millisecond. In the dark the LDR may have a resistance of (1Mohm) which would give a time of 1 second.”
There is like 1 zero added in your statement 😀
Excellent tutorial Mate (Y)

REPLY ›

MATT on JUNE 2, 2015 8:31 PM

Well spotted. I’ve updated the numbers and added a link to a time constant calculator.

REPLY ›

BRUCE on JULY 25, 2015 10:04 PM

Hi, great post!

Any idea why it doesn’t work on a rasperry pi 2?

Thanks!

REPLY ›

MATT on JULY 28, 2015 4:34 PM

There’s no reason it shouldn’t work on a Pi 2. It’s just measuring the time taken for a GPIO pin to go high so I would expect it to work on all models.

REPLY ›

BRUCE on JULY 29, 2015 9:41 AM

Thanks Matt for the answer. The thing is that the pin never goes high on my PI 2 🙁

REPLY ›

GAB on JULY 31, 2015 9:30 AM

Hi,
I gpio is detected high around 1.3V, not 2V (I got an Rpi B+)
I found that with a 10uF capacitor and 10K resistor. I got a high on GPIO after 48ms (according to t=RC it should be 100ms). After 48ms, equation Vc=V-V*e^(-t/(RC)) says it is 1.3V
Am doing something wrong ? Or gpio gets high lower on Rpi B+ ?

REPLY ›

GAB on AUGUST 4, 2015 8:26 AM

Hi (again),
According to different things I found on the web, threshold voltage for GPIO pin is between 0.8V and 2.0V, and … as is on rising edge.
IMHA, one should calibrate rpi’s gpio high treshold. On my rpi b it’s 1.3V, not 2.0V, and it changes the value read

REPLY ›

WUILMER on MAY 7, 2016 2:46 AM

Hi , I have a 100k thermistor ntc 3950 b Podre transform analog signal to digital with this method ?

REPLY ›

KENC on AUGUST 27, 2016 6:40 PM

Very clever. However, I must reiterate a WARNING (mentioned by PAUL on JUNE 5, 2014 12:40 PM), and a simple solution:

Discharging the capacitor directly with a GPIO pin will provide a large inrush current, which risks damage to the GPIO pin. A larger the cap would be even more risky (some people wanted to go with a larger cap, trading off speed for better resolution) . From specs I've read, the default setting on GPIO outputs is 8mA (can be set to 2ma to 16mA), and the recommendation is not to exceed that setting (it isn't actually a current limit). The short time to discharge the cap might let you get away with not over-dissipating the device, but IMO it is not worth the risk when there is a simple solution.

The SOLUTION: Simply add a 470 OHM R from the GPIO pin to the junction of the LDR and cap. This limits the current to ~ 7mA maximum. It will take ~ 2 mSec to discharge the cap (5 time constants).

One (probably minor) effect is that if the LDR is a low resistance (say 1K, with the 2.2K in series), this forms a voltage divider, and the cap will discharge only to ~ 0.4V instead of near zero. That might upset the timing a bit, and a very low resistance (if you used something less that 2.2K in series with the LDR) might keep it above the threshold (which I've read is typically 1.8V but can be between about 0.8V and 2.0V . But I think you will be fine with the 2.2K series and typical LDR values. It just might take a little tweaking of the loop?

REPLY ›

FOLLY on OCTOBER 8, 2016 7:57 PM

In the reply's i did not find how we can measure accurate resistor values and or even calculate the accuracy.
Seems to me this is important to get more out of this technique.

In my setup i used a 3.3uF condensator and a RPI-2.
The timing measured are not directly equal to the resistance.
It seemed that the timing measurements are indeed linear to the resistance.
I used 3 resistors which i choose to measure (1K , 2.2K ,15K)
Then you can draw a graph and calculate an equation y=ax+b or x=(y-b)/a.
(search the internet for more info)

This is the edited program to test the accuracy of the measured resistor.
(i'm not a perfect programmer but this works)
It asks for the resistor you want to measure.
Then it reads 4 measurements (named it "timing-counts") and makes an average to make it more accurate then uses the equation to calculate the resistance.
Last but not least is calculates the accuracy in %.

My finding was that a time.sleep of 0.4 was better for accuracy.
The accuracy is in average about 1 to 2 %

```
#!/usr/local/bin/python

# Reading an analogue sensor with
# a single GPIO pin

# gpio 4
# |
# +3.3V o–====—–====—–||–o gnd
# 2.2k choose 3.3uF

# Author : Matt Hawkins
# Distribution : Raspbian
# Python : 2.7
# GPIO : RPi.GPIO v3.1.0a

import math
import RPi.GPIO as GPIO, time
from datetime import datetime

# Tell the GPIO library to use
# Broadcom GPIO references
GPIO.setmode(GPIO.BCM)

# Define time.sleep variable (edited to 0.4 to get otimal results with 3.3uF)
tmsl = 0.4

# input the measurable Resistance value
realR = input("Please enter the measurable Resistance value in Ohm:")
```

```
# Define function to measure charge time
def RCtimea (PiPin):
measurement = 0

# Discharge capacitor
GPIO.setup(PiPin, GPIO.OUT)
GPIO.output(PiPin, GPIO.LOW)
time.sleep(tmsl)
# Count loops until voltage across
# capacitor reads high on GPIO
GPIO.setup(PiPin, GPIO.IN)
while (GPIO.input(PiPin) == GPIO.LOW):
measurement += 1

return measurement

def RCtimeb (PiPin):
measurement = 0

# Discharge capacitor
GPIO.setup(PiPin, GPIO.OUT)
GPIO.output(PiPin, GPIO.LOW)
time.sleep(tmsl)
# Count loops until voltage across
# capacitor reads high on GPIO
GPIO.setup(PiPin, GPIO.IN)
while (GPIO.input(PiPin) == GPIO.LOW):
measurement += 1

return measurement

def RCtimec (PiPin):
measurement = 0

# Discharge capacitor
GPIO.setup(PiPin, GPIO.OUT)
GPIO.output(PiPin, GPIO.LOW)
time.sleep(tmsl)
# Count loops until voltage across
# capacitor reads high on GPIO
GPIO.setup(PiPin, GPIO.IN)
while (GPIO.input(PiPin) == GPIO.LOW):
measurement += 1

return measurement

def RCtimed (PiPin):
measurement = 0

# Discharge capacitor
GPIO.setup(PiPin, GPIO.OUT)
GPIO.output(PiPin, GPIO.LOW)
time.sleep(tmsl)
# Count loops until voltage across
# capacitor reads high on GPIO
GPIO.setup(PiPin, GPIO.IN)
while (GPIO.input(PiPin) == GPIO.LOW):
measurement += 1

return measurement

# Main program loop
while True:
R1 = RCtimea(4) # Measure timing using GPIO4
R2 = RCtimeb(4) # Measure timing using GPIO4
R3 = RCtimec(4) # Measure timing using GPIO4
R4 = RCtimed(4) # Measure timing using GPIO4
print R1
print R2
print R3
print R4
meanR = ((R1+R2+R3+R4)/4)
```

```
print “Average of 4 measurements “,meanR , ” timing-counts”# Measure timing using GPIO4
# The equation beneath is a lineair equation formed from time-count measures and 3 different resistances so a lineair graph and an equation can be made, the equation is y=ax+b or x=(y-b)/a
# This equation is only usefull with a 2.2k resistor and a 3.3uF Condensator
R = round(((meanR-1500)/0.72693),0) # Measure timing and calculate resistance and round with no decimals using GPIO4
print R,” Ohm”
Difference = (realR-R)
print round(((Difference/realR)*100),3), ” % Accuracy”# won’t work if earlier calculation were done with int()
```

REPLY ›

F on OCTOBER 10, 2016 5:10 PM

Great post.

Used it to get temperature readings with a thermistor.

REPLY ›

LEAVE A REPLY

Your Comment

Your Name

Your Email

Your Website

POST COMMENT

ABOUT

Unofficial site devoted to the Raspberry Pi credit card sized computer offering tutorials, guides, resources,scripts and downloads. We hope to help everyone get the most out of their Pi by providing clear, simple articles on configuring, programming and operating it.

POPULAR POSTS

JULY 27, 2012 90
16×2 LCD Module Control Using Python

OCTOBER 20, 2013 87
Analogue Sensors On The Raspberry Pi Using An MCP3008

SEPTEMBER 19, 2014 70
Top 5 Reasons The Raspberry Pi Sucks

RECENT POSTS

SEPTEMBER 30, 2015
Raspbian Configuration Utility ar New Boot Options

AUGUST 23, 2015
Bristol Mini Maker Faire 2015

JULY 14, 2015
RasPiO GPIO Quick Reference Ru for Raspberry Pi

Entries RSS | Comments RSS

This site is not associated with the official Raspberrypi.org site or the Raspberry Pi Foundation. Raspberry Pi is a trademark of the Raspberry Pi Foundation.