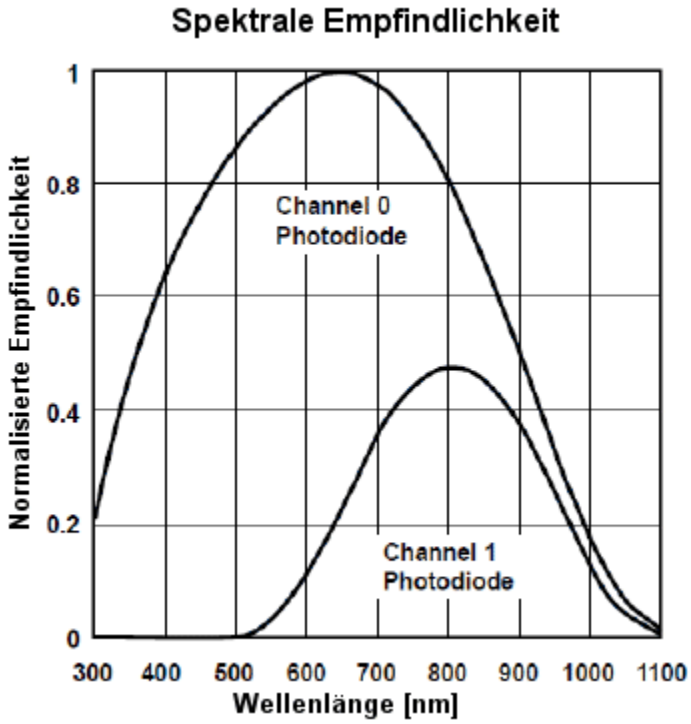
Raspberry-Pi-Projekte: Helligkeitssensoren TSL2561, TSL25911, TSL45315

Prof. Jürgen Plate

Raspberry Pi: Helligkeitssensoren TSL2561, TSL45315

Helligkeitssensor TSL2561

Der Helligkeitssensor TSL2561 ist ein digitaler Lichtsensor für den Einsatz in einem weiten Bereich von Lichtsituationen. Im Vergleich zu kostengünstigen Photozellen ist dieser Sensor präziser. Es lässt sich aus den gewonnen Daten die Helligkeit in Lux berechnen. Der Sensor kann per Programmierung auf verschiedene Helligkeitsbereiche konfiguriert werden und erlaubt so einen weiten Messbereich. Der Sensor besitzt zwei Photodetektoren, einer für Infrarot und einer für das volle Spektrum. Sie können also das Infrarotspektrum, das Vollspektrum oder das sichtbare Licht berechnen. Durch Subtrahieren des Infrarot-Anteils ergibt sich die Stärke des sichtbaren Lichts.



Spektrale Empfindlichkeit des TSL2561

Der Sensor verfügt über eine digitale I^2C-Schnittstelle. Der eingebaute Analog/Digital-Wandler erlaubt den Einsatz mit jedem Mikrocontroller, auch wenn er nicht über analoge Eingänge verfügt. Er eignet sich also bestens für den Raspberry Pi. Der Strombedarf ist extrem niedrig: ca. 0.5 mA im Betrieb und weniger als 15 µA im Powerdown-Modus. Der Sensor wird mit 3.3 V Spannung betrieben, passt also auch von dieser Seite her zum RasPi.

Es können maximal drei Sensoren an einem Bus angeschlossen werden. Die Adressierung erfolgt hierbei über den Anschluss "Addr." des TSL2561. Die folgende Beschaltung zur Adressierung ist möglich:

```
GND   = 0x29
offen = 0x39
VDD   = 0x49
```

Für die nun beschreibenen Experimente bleibt der Anschluss für die Adressauswahl unbeschaltet (Baustein-Adresse 0x39).

Zusätzlich bietet der Sensor die Möglichkeit, über einen Interrupt-Ausgang zu melden, dass ein eingestellter Schwellwert überschritten wurde. Dies ist interessant, wenn der exakte Helligkeitswert nicht so wichtig ist und man ständiges Pollen des Sensors vermeiden will. Auf die dazu notwendigen Registerwerte (Threshold, Interrupt) wird hier nicht eingegangen.

Register des TSL2561

Primär interessieren die Register mit den Adressen 0xC bis 0xF, in denen die Helligkeitswerte stehen.

ADDRESS	RESISTER NAME	REGISTER FUNCTION
—	COMMAND	Specifies register address
0h	CONTROL	Control of basic functions
1h	TIMING	Integration time/gain control
2h	THRESHLOWLOW	Low byte of low interrupt threshold
3h	THRESHLOWHIGH	High byte of low interrupt threshold
4h	THRESHHIGHLOW	Low byte of high interrupt threshold
5h	THRESHHIGHHIGH	High byte of high interrupt threshold
6h	INTERRUPT	Interrupt control
7h	—	Reserved
8h	CRC	Factory test — not a user register
9h	—	Reserved
Ah	ID	Part number/ Rev ID
Bh	—	Reserved
Ch	DATA0LOW	Low byte of ADC channel 0
Dh	DATA0HIGH	High byte of ADC channel 0
Eh	DATA1LOW	Low byte of ADC channel 1
Fh	DATA1HIGH	High byte of ADC channel 1

Quelle: Datenblatt TSL2561

Bei den Registern des Sensors ist zu beachten, dass die höherwertigen vier Bits (7 - 4) das Kommando bestimmen und die niederwertigen vier Bits (3 - 0) die Adresse des Datenregisters für die folgenden Leseoperationen festlegen. Für jede lesende oder schreibende Operation ist mindestens das Bit 7 (CMD - Select Command Register) auf "1" zu setzen. Möchte man beispielsweise das Register mit der Adresse 0xA (Part number/Rev ID) abfragen, so lautet der entsprechende Registerbefehl hierfür 0x83.

Breakout-Board mit TSL2561 (Bild: Adafruit)

Der Sensor kann über die folgenden Befehle ein- bzw. ausgeschaltet werden. Dabei wird das Control-Register (Adresse 0) angesprochen, bei dem nur die Bits 0 und 1 eine Rolle spielen. "00" bedeutet "aus, "11" bedeutet "ein":

```
# ausschalten
i2cset -y 1 0x39 0x80 0x00
# einschalten
i2cset -y 1 0x39 0x80 0x03
```

Die Abtastrate des Sensors wird im Register "Timing" (0x1) eingestellt. Voreinstellung ist hierbei die geringste Rate mit der höchsten Genauigkeit. Ist die Umgebung sehr dunkel, muss gegebenenfalls die Rate ("Gain", Bit 4) von 1x auf 16x umgestellt werden. Die Auflösung ("Integration Time") wird über die Bits 0 und 1 eingestellt. Die folgenden Befehle erlauben die Einstellung der unterschiedlichen Abtastraten auf der Kommandozeile. Bei der Berechnung der Lux-Werte muss dann auch der Skalierungsfaktor beachtet werden:

```
# 13.7 ms, geringste Auflösung, Skalierungsfaktor 0.034
i2cset -y 1 0x39 0x81 0x00
# 101 ms, mittlere Auflösung, Skalierungsfaktor 0.252
i2cset -y 1 0x39 0x81 0x01
# 402 ms, beste Auflösung, Skalierungsfaktor 1.0
i2cset -y 1 0x39 0x81 0x02
```

Die Skalierungswerte ergeben sich aus der Frequenz des Oszillators und der Integrationszeit:

- Integrationszeit(0): (11 × 918)/fosc = 13.7 ms → Skalierungsfaktor: 11/322 = 0.034
- Integrationszeit(1): (81 × 918)/fosc = 101 ms → Skalierungsfaktor: 81/322 = 0.252
- Integrationszeit(2): (322 × 918)/fosc = 402 ms → Skalierungsfaktor: 322/322 = 1

Durch die Wahl der Integrationszeit wird auch der MAximalwert des A/D-Wandlers beschränkt:

- Integrationszeit(0): Maximalwert = (11 × 918)/2 - 2 = 5047
- Integrationszeit(1): Maximalwert = (81 × 918)/2 - 2 = 37177
- Integrationszeit(2): Maximalwert = 65535 (größter 16-Bit-Wert)

Das Timing-Register wird beim Einschalten mit 0x02 vorbelegt (Gain = 1x, Integration Time = 402 ms, Skalierungsfaktor 1.0). Zusätzlich besteht noch die Möglichkeit, für besondere Fälle auf manuelle Abtastung umzuschalten - beispielsweise für eine noch längere Abtastrate:

```
i2cset -y 1 0x39 0x81 0x03 # manuelle Abtastung einschalten
i2cset -y 1 0x39 0x81 0x07 # Start der Messung
i2cset -y 1 0x39 0x81 0x03 # Ende der Messung
```

Programmierung des TSL2561

Mit dem folgenden Python-Programm kann der Lichtsensor getestet werden. Es gibt die Ausgelesenen Werte für Infrarot- und Gesamthelligkeit aus. Anschließend werden der Wert für das sichtbare Licht, das Verhältnis Gesamt/Infrarot und der Lux-Wert berechnet und ausgegeben.

```
#!/usr/bin/env python

import smbus
from time import sleep

# ---------- Konfigurationsbereich:
# die Adressse des TSL2561 kann
# 0x29, 0x39 oder 0x49 sein
BUS = 1
TSL2561_ADDR = 0x39
# --------------------------------

#Instanzieren eines I²C Objektes
i2cBus = smbus.SMBus(BUS)

# Starten des Messvorganges mit 402 ms
# (Skalierungsfaktor 1)
i2cBus.write_byte_data(TSL2561_ADDR, 0x80, 0x03)

while True:
  # Gesamthelligkeit auslesen
  # niederwertiges Byte lesen
  LSB = i2cBus.read_byte_data(TSL2561_ADDR, 0x8C)
  # hoeherwertiges Byte lesen
  MSB = i2cBus.read_byte_data(TSL2561_ADDR, 0x8D)
  Ambient = (MSB << 8) + LSB
  print "Ambient: %d"  % Ambient

  # Infrarothelligkeit auslesen
  # niederwertiges Byte lesen
  LSB = i2cBus.read_byte_data(TSL2561_ADDR, 0x8E)
  # hoeherwertiges Byte lesen
  MSB = i2cBus.read_byte_data(TSL2561_ADDR, 0x8F)
  Infrared = (MSB << 8) + LSB
  print "Infrared: %d" % Infrared

  # Berechnen des sichtbaren Spektrums
  Visible = Ambient - Infrared
  print "Visible: %d" % Visible

  # Berechnen des Faktors Infrared/Ambient
  Ratio = 0
  Lux = 0
  if Ambient != 0:
    Ratio = float(Infrared)/float(Ambient)
    print "Ratio: %f" % Ratio

    # Berechnung Lux-Wert gemaess Datenblatt TSL2561T
    # T, FN, and CL Package
    if 0 < Ratio <= 0.50:
      Lux = 0.0304*float(Ambient) - 0.062*float(Ambient)*(Ratio**1.4)
    elif 0.50 < Ratio <= 0.61:
      Lux = 0.0224*float(Ambient) - 0.031*float(Infrared)
    elif 0.61 < Ratio <= 0.80:
      Lux = 0.0128*float(Ambient) - 0.0153*float(Infrared)
    elif 0.80 < Ratio <= 1.3:
      Lux = 0.00146*float(Ambient) - 0.00112*float(Infrared)
```

```
    else:
      Lux = 0
    print "Lux = %f\n" % Lux
  sleep (0.5)
```

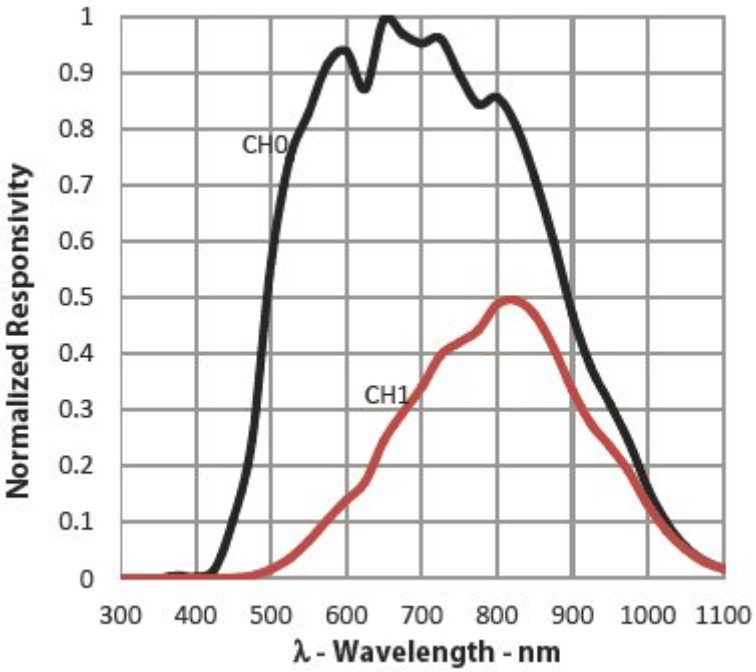
Startet man das Programm, ergibt beispielsweise folgender Output:

```
Revision 0x50

Ambient  522
Infrared 183
Visible  339
Ratio    0.35
Lux      8.41

Ambient  409
Infrared 165
Visible  244
Ratio    0.40
Lux      5.32

Ambient  405
Infrared 164
Visible  241
Ratio    0.40
Lux      5.23

Ambient  537
Infrared 179
Visible  358
Ratio    0.33
Lux      9.17

Ambient  1038
Infrared 201
Visible  837
Ratio    0.19
Lux     25.09
^CBye
```

Es hat sich gezeigt, dass der Sensor bei großer Helligkeit (Sonnenschein, aussen) in die Begrenzung geht und Maximalwerte für sichtbares wie infrarotes Licht liefert. Das hat zur Folge, dass die Berechnung der Lux-Werte daneben geht. Das Extrembeispiel `65535 65535 0 1.00 22.28` verdeutlicht das. Deshalb wurden weitere Sensoren untersucht.

Helligkeitssensor TSL25911

Der Helligkeitssensor TSL25911 ist ein digitaler Lichtsensor, der dem vorhergehenden Modell ziemlich ähnlich ist. Auch dieser Sensor kann per Programmierung auf verschiedene Helligkeitsbereiche konfiguriert werden und erlaubt so einen weiteren Messbereich als der TSL2561. Der Sensor besitzt zwei Photodetektoren, einer für Infrarot und einer für das volle Spektrum. Sie können also das Infrarotspektrum, das Vollspektrum oder das sichtbare Licht berechnen. Durch Subtrahieren des Infrarot-Anteils ergibt sich die Stärke des sichtbaren Lichts.



Spektrale Empfindlichkeit des TSL25911

Der TSL25911 hat ebenfalls eine digitale I^2C-Schnittstelle und eignet sich daher ebenfalls bestens für den Raspberry Pi. Der Strombedarf ist extrem niedrig: ca. 0.3 mA im Betrieb. Der Sensor wird mit 3.3 V Spannung betrieben, passt also auch von dieser Seite her zum RasPi. Der Sensor liegt fest auf der I^2C-Adresse 0x29.

Technische Daten:

- Empfindlichkeit: 0.2 bis 40'000 Lux
- Dynamikbereich: 600M:1
- Programmierbare Interruptfunktion
- Abmessungen: 2,0 mm x 2,4 mm Dual Flat no-lead (FN) package
- Betrieb bei Infrarotlicht möglich
- Versorgungsspannung: 2,7 V - 3,6 V
- Programmierbar: Gain, Integrationszeit, Interrupt

Zusätzlich bietet der Sensor die Möglichkeit, über einen Interrupt-Ausgang zu melden, dass ein eingestellter Schwellwert überschritten wurde. Dies ist interessant, wenn der exakte Helligkeitswert nicht so wichtig ist und man ständiges Pollen des Sensors vermeiden will. Auf die dazu notwendigen Registerwerte (Threshold, Interrupt) wird hier nicht eingegangen.

Das Breakout-Board mit dem TSL25911 enthält einen Spannungsregler, das Board kann mit Spannungen zwischen 3,3 V und 5 V betrieben werden. Die Spannungsanpassung an den I^2C-Bus ist ebenfalls auf den Board realisiert.

Breakout-Board mit TSL25911 (Bild: Watterott)

Register des TSL25911

Bei den Registern des Sensors ist zu beachten, dass die höherwertigen drei Bits (7 - 5) das Kommando bestimmen und die niederwertigen vier Bits (4 - 0) die Adresse des Datenregisters für die folgenden Leseoperationen festlegen. Für jede lesende oder schreibende Operation ist mindestens das Bit 7 (CMD - Select Command Register) auf "1" zu setzen. Die nächsten beiden Bits erlauben die Wahl zwischen normalem Betrieb und speziellen Transaktionen. Für den Normalbetrieb gilt hier die Kmbination "01", so dass das höherwertige Nibble des Kommandowortes - abhängig von der Registeradresse den Wert 0xA oder 0xB besitzt.

Das Enable-Register auf Adresse 0 erlaubt das Ein- und Ausschalten deS Energiesparmodus sowie die Freigabe von Messung und Interrupts. Für die Freigabe der Messung und Power-On muss 0x03 in das Register (KOmmandowort 0xA0) geschrieben werden.

Über das Control-Register werden (neben einem möglichen Soft-Reset) Integrationszeit (Bits 2 - 0) und Gain (Bits 5 und 4) festgelegt. Die Bitpostion kann in die Konstantendefinition des Programms übernommen werden. Auf diese Weise kann dann die Einstellung des Chips durch eine bitweise Oder-Verknüpfung der beiden Konstanten erfolgen:

```
# Konstantendefinition fuer Integrationszeit (Control-Reg.)
ITIME_100 = 0x00  # Integrationszeit [ms]
ITIME_200 = 0x01
ITIME_300 = 0x02
ITIME_400 = 0x03
ITIME_500 = 0x04
ITIME_600 = 0x05

# Konstantendefinition fuer Gain (Control-Reg.)
GAIN_LOW  = 0x00  # low gain (1x)
GAIN_MED  = 0x10  # medium gain (25x)
GAIN_HIGH = 0x20  # high gain (428x)
GAIN_MAX  = 0x30  # maximum gain (9876x)
```

Das Controlregister (Adresse 1) wird über das Kommandowort 0xA1 ausgewählt. Abhängig von Integrationszeit und Gain müssen auch noch zwei Rechenkonstanten für die Lux-Berechnung als Gleitpunktzahlen festgelegt werden. Bei der Integrationszeit entspricht der Wert der Zeit im Millisekunden, die Faktoren für Gain sind oben im Klammern angegeben. Im Programm werden dafür zwei assoziative Arrays angelegt.

Es gibt auch wieder eine Device-Id (Register 0x12) und ein Statusregister (0x13) sowie die diversen Interruptregister. Diesbezüglich sei auf das Datenblatt verwiesen. Die Daten (Gesamtlicht und Infrarot-Anteil) werden als zwei 16-Bit-Werte gespeichert. Wenn das niederwertige Byte eines sogenannten Channels gelesen wird, speichert der Chip gleichzeitig das höherwertige Byte des gleichen Channels wird in einem Schattenregister. Dieses Schattenregister sorgt dafür, dass

beide Bytes das Ergebnis des gleichen Integrationszyklus sind. Die vier Register haben die Adressen:

Register	Address	Description
C0DATAL	0x14	ALS CH0 data low byte
C0DATAH	0x15	ALS CH0 data high byte
C1DATAL	0x16	ALS CH1 data low byte
C1DATAH	0x17	ALS CH1 data high byte

Quelle: Datenblatt TSL25911

Programmierung des TSL25911

Mit dem folgenden Python-Programm kann der Lichtsensor getestet werden. Es gibt die ausgelesenen Werte für Infrarot- und Gesamthelligkeit aus. Anschließend werden der Wert für das sichtbare Licht, das Verhältnis Gesamt/Infrarot und der Lux-Wert berechnet und ausgegeben. Im Programm lassen sich Gain und Integrationszeit durch Angabe der oben gezeigten Konstanten festlegen:

```
   ...
# Auswahl von Gain und Integrationszeit
Gain = GAIN_LOW
Integrationtime = ITIME_400
   ...
```

Die Berechnung des Lux-Wertes nach einer empirischen Formel ist wesentlich einfacher als beim TSL2561. Nach Auswahl der Rechenkonstanten für Gain und Integrationszeit genügen vier Berechnungen. Das Programm läuft in einer Endlosschleife, die mit [Strg]-[C] abgebrochen werden kann.

```
#!/usr/bin/env python

import smbus
from time import sleep

# Busnummer und Deviceadresse
BUS = 1
TSL2561_Adr = 0x29

# Konstantendefinition fuer Integrationszeit (Control-Reg.)
ITIME_100 = 0x00  # Integrationszeit [ms]
ITIME_200 = 0x01
ITIME_300 = 0x02
ITIME_400 = 0x03
ITIME_500 = 0x04
ITIME_600 = 0x05

# Konstantendefinition fuer Gain (Control-Reg.)
GAIN_LOW  = 0x00  # low gain (1x)
GAIN_MED  = 0x10  # medium gain (25x)
GAIN_HIGH = 0x20  # high gain (428x)
GAIN_MAX  = 0x30  # maximum gain (9876x)

# Rechenfaktoren fuer Lux-Berechnung abhaengig
# von Integrationszeit und Gain
Integ_sel = {
    ITIME_100: 100.0,
```

```
    ITIME_200: 200.0,
    ITIME_300: 300.0,
    ITIME_400: 400.0,
    ITIME_500: 500.0,
    ITIME_600: 600.0,
    }

Gain_sel = {
    GAIN_LOW: 1.0,
    GAIN_MED: 25.0,
    GAIN_HIGH: 428.0,
    GAIN_MAX: 9876.0,
    }

# Auswahl von Gain und Integrationszeit
Gain = GAIN_LOW
Integrationtime = ITIME_400

# Instanzieren eines I2C Objektes
i2cBus = smbus.SMBus(BUS)

# Starten des Messvorganges
i2cBus.write_byte_data(TSL2561_Adr, 0xA0, 0x03)
sleep(0.5)

# Gain und Integrationszeit einstellen
i2cBus.write_byte_data(TSL2561_Adr, 0xA1, Gain | Integrationtime)
sleep (1.0)

# Id auslesen
Id = i2cBus.read_byte_data(TSL2561_Adr, 0xB2)
print "Id: 0x%2X\n" % Id

while True:
  # Auslesen
  LSB = i2cBus.read_byte_data(TSL2561_Adr, 0xB4)
  MSB = i2cBus.read_byte_data(TSL2561_Adr, 0xB5)
  Ambient = float(MSB*256) + float(LSB)
  print "Ambient %d"  % Ambient
  LSB = i2cBus.read_byte_data(TSL2561_Adr, 0xB6)
  MSB = i2cBus.read_byte_data(TSL2561_Adr, 0xB7)
  Infrared = float(MSB*256) + float(LSB)
  print "Infrared %d" % Infrared

  # Berechnen des sichtbaren Spektrums
  Visible = Ambient - Infrared
  print "Visible %d" % Visible

  # Berechnen des Faktors IR/Ambient
  Ratio = Infrared/Ambient
  print "Ratio %.2f" % Ratio

  # Berechnung Lux-Wert
  if Gain in Gain_sel.keys():
    again = Gain_sel[Gain]
  else:
    again = 1.0

  if Integrationtime in Integ_sel.keys():
    atime = Integ_sel[Integrationtime]
  else:
    atime = 100.0
  cpl = (atime*again)/408.0
  if cpl != 0.0:
```

```
    lux1 = (Ambient - (1.64*Infrared))/cpl
    lux2 = ((0.59*Ambient) - (0.86*Infrared))/cpl
    Lux = max([lux1, lux2])
  else:
    Lux = 0

  print "Lux = %.2f\n" % Lux
  sleep(1.0)
```

Die Ausgabe des Programms ergibt beispielsweise:

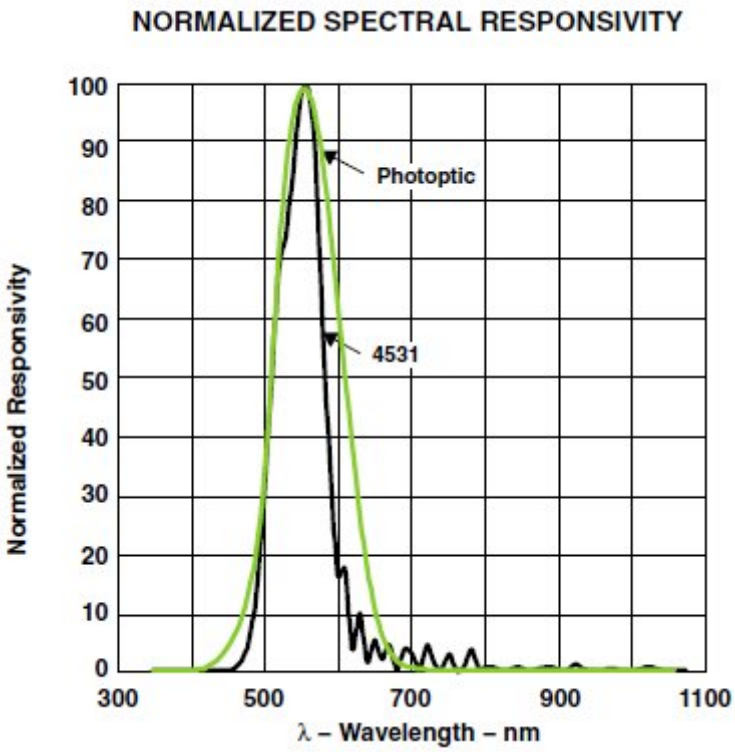
```
Id: 0x50

Ambient 651
Infrared 158
Visible 493
Ratio 0.24
Lux = 399.72

Ambient 12632
Infrared 1785
Visible 10847
Ratio 0.14
Lux = 9898.69

Ambient 18147
Infrared 2539
Visible 15608
Ratio 0.14
Lux = 14262.70

Ambient 25350
Infrared 3522
Visible 21828
Ratio 0.14
Lux = 19965.40

Ambient 46318
Infrared 6361
Visible 39957
Ratio 0.14
Lux = 36603.68

Ambient 65535
Infrared 9668
Visible 55867
Ratio 0.15
Lux = 50673.07
```

Der Helligkeitssensor TSL45315

Der TSL45315 ist ein digitaler Lichtsensor, der die Messwerte direkt in Lux zurückgibt. Damit erhält man eine physikalische Größe, die direkt die eintreffende Lichtmenge pro Quadratmeter angibt. Der Sensor wird über eine die I^2C-Schnittstelle konfiguriert und abgefragt. Der Umgebungslichtsensor TSL4531 bietet einen einfachen, direkten Lux-Ausgang und eine 16-Bit-Digitalschnittstelle. Ausgefeilte Filter filtern automatisch eventuell vorhandene 100/120-Hz-Störungen, die typischerweise von den Leuchtstoffbeleuchtungssystemen eines Gebäudes produziert werden, so dass der gemessenen Lichtpegel dem Tageslicht entspricht. Eine spezielle Infrarot-Messung gibt es bei diesem Baustein nicht. Einige Eigenschaften des TSL45315 sind:

- Direkte Ausgabe in Lux

- Approximattion des Spektrums des menschlichen Auges
- Drei wählbare Integrationszeiten: 400 ms, 200 ms, and 100 ms
- Dynamikbereich: 3 lux bis 220'000 lux
- Minimaler Strombedarf 110 µA aktiv, 2.2 µA power down
- 16-Bit Digitalausgang über I^2C interface
- 2.5 V Versorgungsspannung, 1.8 V Logikpegel
- Filter für 100/120 Hz



Spektrale Empfindlichkeit des TSL45315

Da der Sensor selbst nur etwa 2x2 mm groß ist, greift man am einfachsten zu einem Breakout-Board, welches z. B. von Watterott angeboten wird. Das bietet auch noch als Features einen Eingangsspannungsbereich von 3,3 V bis 5 V und ein 3,3 V bis 5 V tolerantes I^2C-Interface.

Breakout-Board mit TSL45315 (Bild: Watterott)

Der Sensor liegt auf Adresse 0x29 des I^2C-Bus. Unter der Typbezeichnung TSL45311 ist eine Variante für die Adresse 0x39 verfügbar. Auch existieren Sensoren, die mit 1,8 V Betriebsspannung arbeiten. Da der I^2C-Bus bei Raspberry Pi mit 3,3 V arbeitet, bleibt es beim TSL45315.

Der ADC-Ausgangswert ist eine 16-Bit-Zahl, die direkt proportional zur Luminanz (lux) ist. Die Luminanz kann nach folgender Formel berechnet werden:

```
Luminanz (lux) = FAKTOR × ((DATAHIGH << 8) + DATALOW)
```

Dabei gilt für den Faktor:

- FAKTOR = 1: TCNTRL = 00 (Integrationszeit = 400 ms)
- FAKTOR = 2: TCNTRL = 01 (Integrationszeit = 200 ms)
- FAKTOR = 4: TCNTRL = 10 (Integrationszeit = 100 ms)

Register des TSL45315

Die Zahl der Register des TSL45315 ist überschaubar. Auch hier erfolgt das Ansprechen des jeweiligen Registers über ein Command-Byte mit gesetztem achten Bit und der jeweiligen Registeradresse.

ADDRESS	REGISTER NAME	R/W	REGISTER FUNCTION	RESET VALUE
--	COMMAND	W	Specifies register address	0x00
0x00	CONTROL	R/W	Power on/off and single cycle	0x00
0x01	CONFIG	R/W	Powersave Enable / Integration Time	0x00
0x04	DATALOW	R	ALS Data LOW Register	0x00
0x05	DATAHIGH	R	ALS Data HIGH Register	0x00
0x0A	ID	R	Device ID	ID

Quelle: Datenblatt TSL45315

Das Control-Register (Adresse 0) verwendet nur die unteren beiden Bits (1 und 0):

```
00    Power Down
01    Reserved
10    Run a single ADC cycle and return to PowerDown
11    Normal Operation
```

Das Configuration-Register (Adresse 1) verwendet nur Bit 3 und die unteren beiden Bits (1 und 0). Bit 3 (PSAVESKIP) legt den Energiesparmodus fest. Wenn aktiviert, werden die Energiespar-Zustände nach einem Integrationszyklus übersprungen, um eine kürzere Abtastrate zu erreichen. Bei PSAVESKIP = 0 ist die typische Gesamtzykluszeit `Integrationszeit + (60/FAKTOR) ms`. Ist PSAVESKIP = 1, dauert der typische Gesamtzyklus nur so lange wie die Integrationszeit. Die Bits 0 und 1 legen die Integrationszeit und damit den FAKTOR fest:

```
00    Faktor = 1, Integrationszeit = 400 ms
01    Faktor = 2, Integrationszeit = 200 ms
10    Faktor = 4, Integrationszeit = 100 ms
11    reserviert
```

Die beiden Datenregister liefern den Wert der Luminanz (Register 4 LSB, Register 5 MSB), siehe Beispielprogramm. Das letzte Register mit der Adresse 0x0A (!) erlaubt die Indentifizierung des Chips:

```
0x08   TSL45317
0x09   TSL45313
0x0A   TSL45315
0x0B   TSL45311
```

Programmierung des TSL45315

Mit dem folgenden Python-Programm kann der Lichtsensor getestet werden. Es gibt die ausgelesenen Werte für Id und Luminanz aus.

```
#!/usr/bin/env python

import smbus
import time

# ---------- Konfigurationsbereich:
# die Adressse des TSL45315
BUS = 1
TSL45315_ADDR = 0x29
# --------------------------------

#Instanzieren eines I2C Objektes
bus = smbus.SMBus(BUS)

# Control Register, 0x00
# start normal operation
bus.write_byte_data(TSL45315_ADDR, 0x80, 0x03)

# Configuration Register, 0x01
# Multiplikator 1x, Integrationszeit 400 ms
# Je nach gewaehlter Messdauer muss der Messwert laut
# Datenblatt mit einem Faktor multipliziert werden
bus.write_byte_data(TSL45315_ADDR, 0x81, 0x00)

time.sleep(0.5)

# Id lesen
id = bus.read_byte_data(TSL45315_ADDR, 0x8A) >> 4

# Daten lesen von Register 0x04, 2 bytes, LSB first
data = bus.read_i2c_block_data(TSL45315_ADDR, 0x84, 2)

# 16-Bit-Wert (Lux)
#luminance = (data[1] * 256) + data[0]
luminance = (data[1] << 8) + data[0]

# Output data to screen
print "Id: 0x0%X" %id
print "Luminanz: %d lux" %luminance
```

Startet man das Programm, ergibt beispielsweise folgender Output:

```
Id: 0x0A
Luminanz: 489 lux
```

Mit mehr Licht ergibt sich beispielsweise:

```
Id: 0x0A
Luminanz: 65535 lux
```

Links

- Datenblatt TSL2561
- Testprogramm TSL2561_2.py (Verwendung der smbus-Library)
- Testprogramm TSL2561_1.py (nur die Werte einmal in einer Zeile)
- Testprogramm TSL2561.py (ohne smbus-Library)

- TSL2561-Breakoutboard bei Watterott
- Datenblatt TSL25911
- Testprogramm TSL25911.py (Verwendung der smbus-Library)
- TSL25911-Breakoutboard bei Watterott
- Datenblatt TSL45315
- Testprogramm TSL45315.py (Verwendung der smbus-Library)
- TSL45315-Breakoutboard bei Watterott

Letzte Aktualisierung: 06/10/2017 17:19:03