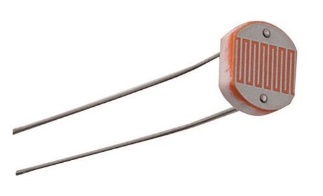
**** A photoresistor or photocell is a light-controlled variable resistor.

The resistance of a photoresistor decreases with increasing light.

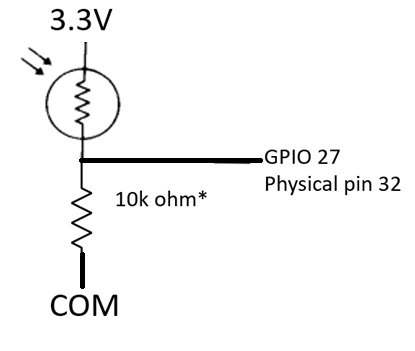
These photoresistors are built to have the same spectral response as human eyes.

The difference between component’s varies a lot, in bright light 100 ohms to 400 ohms.

In the dark, measures anywhere from 200k to 2 Mohm.

Because of the low resistance in bright light, a series resistor is needed to limit the photo resistor current. That series resistor also creates a nice voltage divider to wire into GPIO for brightness detection.

The photoresistor datasheet details about light levels and range the photoresistor can be applied to.



Wired this way, bright light has a higher voltage & numeric value of analog input, darkness takes the GPIO analog input value to zero.

Vcc is 3.3V

Vgpio/ Vcc = 10 k ohm / (Rphotoresistor + 10kohm)

\*this is where we become engineers. 10k was selected as a swag – educated guess of a starting point.

A simulator program could produce tables of different volts VS ohms VS light conditions. We could do the same thing with a spreadsheet to optimize and also experiment with different values.

**CASE 1 (300 ohm)**At extreme bright the photoresistor resistance is near zero. 300 ohms current limit resistor would limit the ckt to 10mA, easy on component & Pico power supply.

Vgpio would be 3.3V \*300 ohms /500 ohms, 2 V. Average light to dark 3.3V\*200,000 ohms/200,300 ohms 3.3V, the analog range is only using 30% of available counts.

**CASE 2 ( use 1kohm resistor)** - bright light Vgpio would be 3.3V \* 1000 ohms /1300 ohms, 2.5 V

Much improved range, average light to dark 3.3V\*1000 ohms/201,000 ohms = .016V

**CASE 3 (10 kohm resistor)** – bright light Vgpio would be 3.3V \* 10000 ohms /10300 ohms, 3.2V. The average light to dark 3.3V\*10,000 ohms/210,000 ohms = .157 V has more range and tolerance in case the photo resistor has a higher real value. If 2Mohm, then 3.3V\*10,000 ohms/2,010,000 ohms= .016V

Even tho python provides a resolution of 1/65535, the real analog resolution is still 1/4095 = .0002 V

So could even try 100kohm as an experiment.

For this example, we stick with the swag at 10kohm.

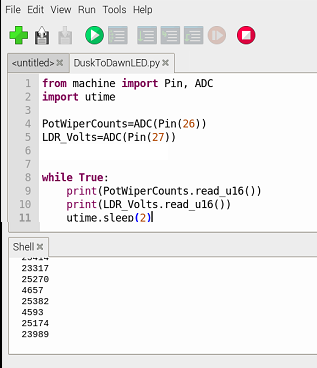
**Open program ReadPot\_GP26\_A0.py, and save as DuskToDawnLED.py**

**So lets first write a program to switch on the LED when the it gets dark outside**.

**Your house may have outside lights with this same circuit, or car may have this exact same control circuit for the headlights! Headlights would have some time delay in case you drive under a shade tree.**

**First add in the code line to read the LDR\_Volts at GPIO 27 (A1), run the code to see the value.**

**Put your finger over the LDR and observe the value change, it should get smaller.**

****

**After proving the analogs are reading, then complete the coding**

**DuskToDawnLED.py**

from machine import Pin, ADC

import utime

PotWiperCounts=ADC(Pin(26))

LDR\_Counts=ADC(Pin(27))

u16\_2\_volts = 3.3/(65535) #create a conversion factor

DuskToDawnLED= Pin(25, machine.Pin.OUT)#GP25 is the Pico mounted LED

DuskToDawnLED.value(0)

LightOnAdjust=1.1 ## in volts at GPIO

while True:

#print(PotWiperCounts.read\_u16())

#print(LDR\_Counts.read\_u16())

LDR\_Volts=(LDR\_Counts.read\_u16())\*u16\_2\_volts

print(LDR\_Volts)

if ((LDR\_Volts) < (LightOnAdjust)):

DuskToDawnLED.value(1)

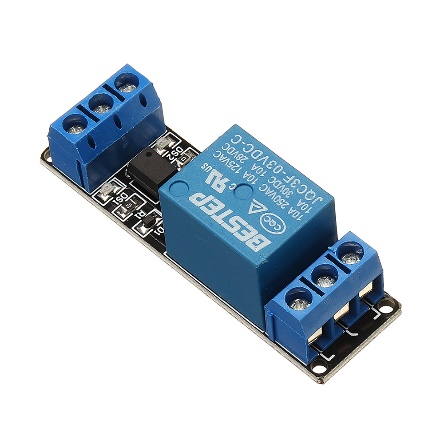
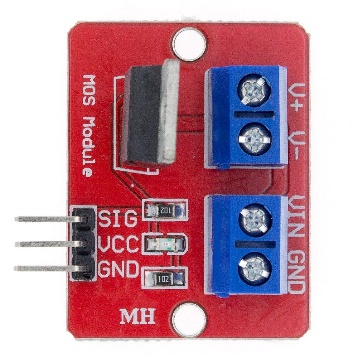
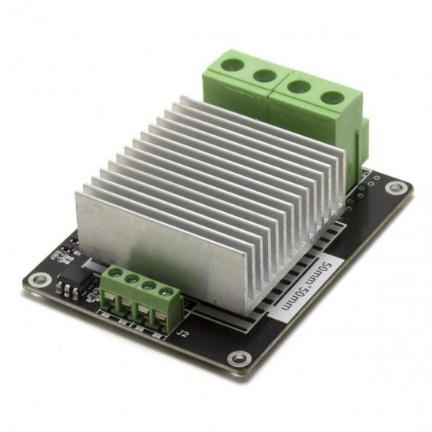
else:

DuskToDawnLED.value(0)

utime.sleep(2)

**hardware:**

If you’d like to turn on a bank of high power LEDS or a motor, you can use this GPIO output to drive a bipolar transistor base or a MOSFET gate to switch on a lot bigger loads, or use a relay module.

**NEXT DuskToDawnLED.py save as DuskToDawnLEDwDimmer.py**

**\*\*\* sorry, GP25 – Rpi Pico mounted LED cannot be PWM ☹ please use GP2 (pin 4 ) for an LED**

**write a program to dim or brighten the LED as the ambient light changes.**

**HINT: Your car may have this exact same control circuit for the dashboard lights !**

**Make the LED its brightest when it turns on, and dims as the LDR\_volts heads towards zero.**

**DuskToDawnLEDwDimmer.py**

**from machine import Pin, ADC, PWM**

**import utime**

**PotWiperCounts=ADC(Pin(26))**

**LDR\_Counts=ADC(Pin(27))**

**u16\_to\_volts = 3.3/(65535)**

**#volts\_2\_U16 = (65535)/3.3**

**# DuskToDawnLED= Pin(25, machine.Pin.OUT)#GP25 is the Pico mounted LED**

**# DuskToDawnLED.value(0)**

**DuskToDawnLED= PWM(Pin(2))# physical pin 4**

**DuskToDawnLED.freq(100)**

**LightOnAdjust=1.5 ## in volts at GPIO**

**# volts\_to\_U16 = (65535)/LightOnAdjust**

**volts\_to\_U16 = (65535)/LightOnAdjust**

**while True:**

**#print(PotWiperCounts.read\_u16())**

**#print(LDR\_Counts.read\_u16())**

**LDR\_Volts=(LDR\_Counts.read\_u16())\*u16\_to\_volts**

**print(LDR\_Volts)**

**if ((LDR\_Volts) < (LightOnAdjust)): ## floating point value of volts**

**DimmerDutyCounts=65535**

**if ((LDR\_Volts) < ((LightOnAdjust)\*.8)): ## floating point value of volts**

**DimmerDutyCounts=52400**

**if ((LDR\_Volts) < ((LightOnAdjust)\*.6)): ## floating point value of volts**

**DimmerDutyCounts=39000**

**if ((LDR\_Volts) < ((LightOnAdjust)\*.4)): ## floating point value of volts**

**DimmerDutyCounts=26200**

**if ((LDR\_Volts) < ((LightOnAdjust)\*.2)): ## floating point value of volts**

**DimmerDutyCounts=13000**

**if ((LDR\_Volts) > ((LightOnAdjust))): ## floating point value of volts**

**DimmerDutyCounts=0**

**DuskToDawnLED.duty\_u16(DimmerDutyCounts)**

**utime.sleep(2)**

**Sidetracked: Persistence of vision** traditionally refers to the optical illusion that occurs when visual perception of an object does not cease for some time after the rays of light proceeding from it have ceased to enter the eye.[[1]](https://en.wikipedia.org/wiki/Persistence_of_vision#cite_note-Cyclop%C3%A6dia-1) The illusion has also been described as "retinal persistence",[[2]](https://en.wikipedia.org/wiki/Persistence_of_vision" \l "cite_note-2) "persistence of impressions",[[3]](https://en.wikipedia.org/wiki/Persistence_of_vision#cite_note-3) simply "persistence" and other variations.

PLAYTIME = change DuskToDawnLED.freq(100) to DuskToDawnLED.freq(10), what do you see ?

PLAYTIME = change DuskToDawnLED.duty\_u16(32767) to change DuskToDawnLED.duty\_u16(xxxxx). How close to 0 can xxxxx before you cant see the LED glow ? what do you see ? Does it matter how bright the room is in seeing the LED glow ?

PLAYTIME change **LightOnAdjust= xx** to different values. What happens ?

Can you change the if statements to a u\_16 variable for dimmer duty cycle ?