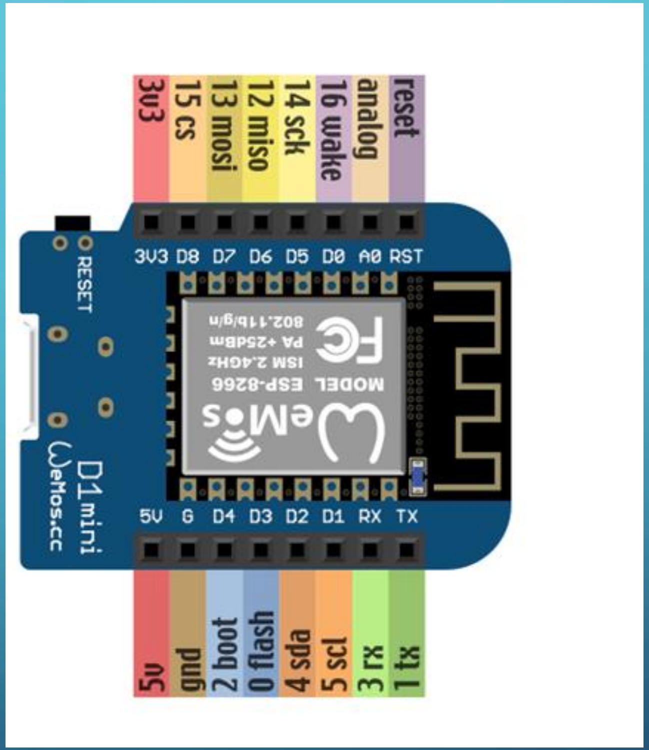
DIGITAL I/O AND TIMERS

BY M. ALLEMANG

THE D1 MINI ESP8266



DIGITAL I/O

- Uses the Pin object from the machine library

from machine import Pin

- Then create an input or output Pin object:

DIGITAL INPUTS

```
myInput = Pin( pin_number, Pin.IN, Pin.PULL_UP)
```

(Pin.PULL_UP is optional)

- Then to use the input:

```
if myInput():
  print(‘the input is high’)
else:
  print(‘the input is low’)
```

DIGITAL OUTPUTS

```
myOutput = Pin(pin_number, Pin.OUT)
```

- To control the output:

```
myOutput(1)
myOutput(0)
```

- The constants 1 and 0 can obviously be variables or list elements
- How to work with lists and input/output objects is described in the Lab – Intro to the GPIO

TIMERS

- All implementations of python have a time function allowing your program to sleep for a number of seconds or fractions of seconds..then wake up and continue

```
import time
while(1):
  print(‘hello’)
  time.sleep(.5)
  print(‘goodbye’)
  time.sleep(.5)
```

TIMER CALLBACKS

- But no work can be done while the program is sleeping
- There needs to be a way to provide a periodic or one-shot timer separate from the main loop, such as for the Real Timed Interrupt or the output timer/counter subsystem in other microcontrollers
- In high level language implementations, this is done via call-back functions.
- The call-back function is essentially the interrupt service routine
- The call-back function will be called periodically by the timer

TIMER ON THE ESP8266

- There is a virtual hardware timer implemented on this microcontroller (others have actual hardware timers).

```
from machine import Timer
myTim = Timer(-1)     (the -1 would be replaced by the hardware timer number)
```

- Now that the timer object is created, you configure it as follows:

```
myTim.init(period=2000, mode=Timer.PERIODIC, callback=my_func)    or
myTim.init(period=2000, mode=Timer.ONE_SHOT, callback=my_func)
```

The function my_func(myTim) will be called when the timer fires.

```
# testing callback

from machine import Pin
from machine import Timer
import time

#a global counter
intcnt=0
#create a virtual timer
tim = Timer(-1)

#this is the call-back function
def my_func(tim):
    global intcnt
    intcnt+=1

#configure the timer
tim.init(period=500, mode=Timer.PERIODIC, callback=my_func)

while(1):
    print('intcnt is now ', intcnt)
    time.sleep(1)
```

EXTRAS

- You can modify the period at any time by simply re-executing the init method with the same arguments with a new delay

EXAMPLE MODIFYING THE PERIOD OF A FLASHING LED

timer_example.py