Timer Interrupts

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #17

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

The previous lecture covered edge interrupts

- Subroutine called by hardware
- Triggered by a rising or falling edge
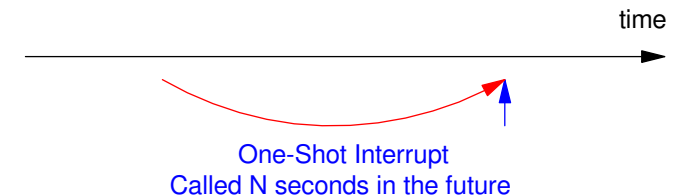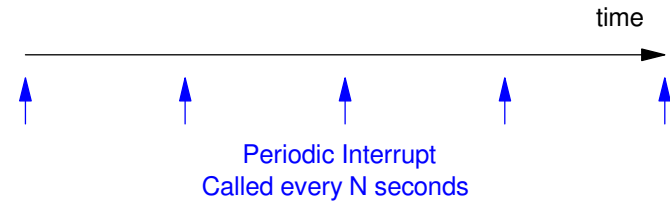
This lecture looks at timer interrupts

- Interrupts triggered every N seconds
- Interrupts triggeded N seconds from now

Timer interrupts are useful:

- More efficient & accurate version of sleep()
- More precise timing of the main loop
- Processor isn't shut down like sleep()

With timer interrupts, you can

- Build an automated stoplight
- Sample a voltage every N ms
- Implement a digital filter

time

Periodic Interrupt
Called every N seconds

time

One-Shot Interrupt
Called N seconds in the future
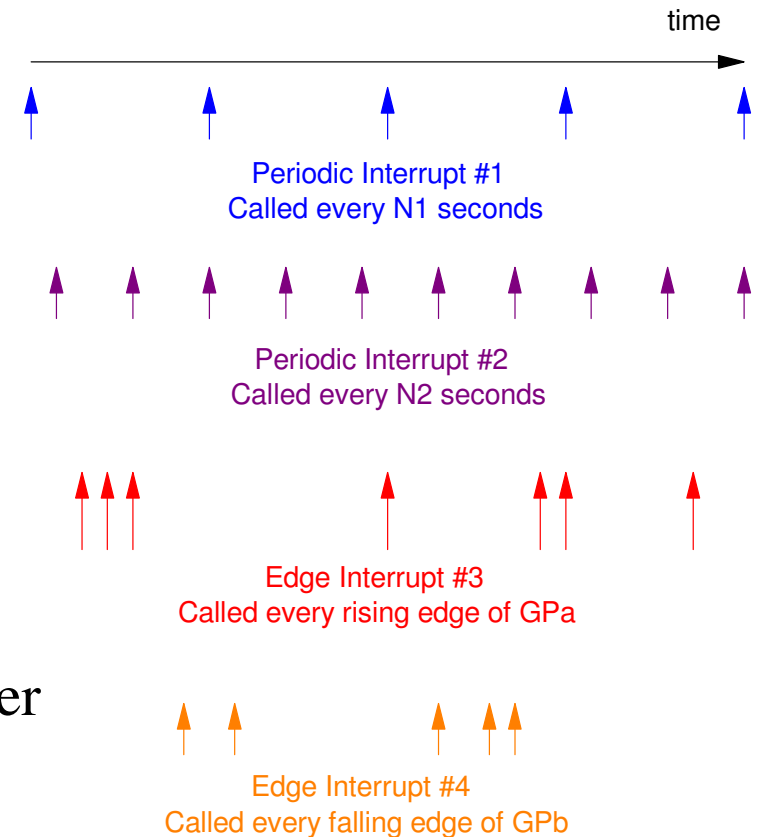
In this lecture, we'll go over

- What Timer interrupts are,
- How they're set up,

and some things you can do with them

- Implement a Ton() function:
    - A button has to be held down for T seconds
- Implement a Toff() function:
    - An output remains on for T seconds
- Set the timing for a stoplight
- Set the timing of a game
    - play *hungry hungry hippo* for 10.00 seconds

Note: Interrupts don't interfere with each other

- You can turn on multiple timer interrupts
- You can turn on multiple edge interrupts
- All at the same time

time

Periodic Interrupt #1
Called every N1 seconds

Periodic Interrupt #2
Called every N2 seconds

Edge Interrupt #3
Called every rising edge of GPa

Edge Interrupt #4
Called every falling edge of GPb

Turning On Periodic Timer Interrupts

Starting out, let's just turn on a timer interrupt

- Count once per second

Timer interrupts are similar to edge interrupts:

- Called by hardware
  - one second elapses
- Global variables are needed
  - pass data to the main routine
- Only called when needed
  - The main loop is free to do whatever between interrupts
  - Different from sleep(1)

```python
from machine import Pin, Timer
from time import sleep_ms

led = Pin(17, Pin.OUT)
tim = Timer()
N = 0

def tic(timer):
    global N
    N += 1

tim.init(freq=1, mode=Timer.PERIODIC,
callback=tic)

while(1):
    print(N)
    sleep_ms(100)
```

Timer Interrupts

Program Desciption:

- Timer
  - timer interrupt function
  - low-level routine from machine
- N
  - Global variable
  - Passes the count to the main routine
- tick(time):
  - Interrupt service routine
- freq=1
  - interrupt is called every 1Hz
- mode=Timer.PERIODIC
  - called over an over again
- mode=Timer.ONE_SHOT
  - called just once

```python
from machine import Pin, Timer

led = Pin(17, Pin.OUT)
tim = Timer()
N = flag = 0

def tick(timer):
    global N
    global flag
    N += 1
    flag = 1

tim.init(freq=1, mode=Timer.PERIODIC,
callback=tick)

while(1):
    if(flag):
        flag = 0
        print(N)
```
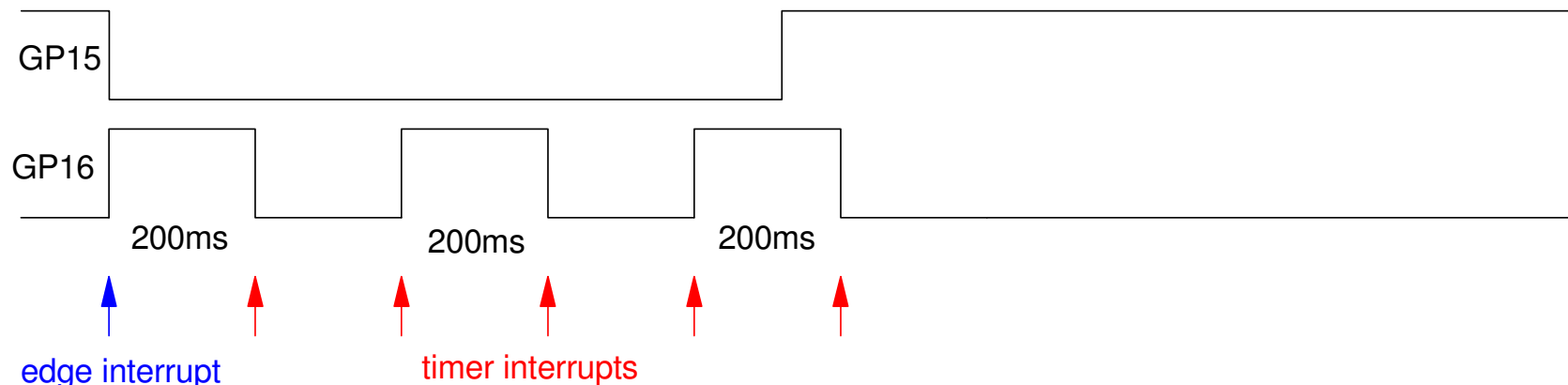
Program 2: Fire Cheat:

Both edge and timer interrupts can be used

- Interrupts do not conflict

Example: Fire-Button Cheat

- When GP15 is pressed, (falling edge interrupt)
- Three pulses are output of GP16 (three shots using a timer interrupt)
- Each shot should be on for 200ms and off for 200ms

Fire Cheat

Fire

- Falling Edge interrupt
- Initializes N = 5
  - five toggles coming up
- Initializes timer interrupts
  - 5Hz (200ms)
  - periodic

Tic

- Timer interrupt
- Decrements N down to zero
- If on last toggle (N==1)
  - Makes next interrupt a one-shot

Main Loop

- Doesn't do anything
- Interrupts do all the work
- The flag isn't needed
  - just prints on toggles

```python
from machine import Pin, Timer
from time import sleep_ms

tim = Timer()
flag = N = 0
pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
LED  = Pin(16,Pin.OUT)

def Fire(pin1):
    global N, flag
    LED.value(1)
    if(N == 0):
        N = 5
        flag = 1
        tim.init(freq=5, mode=Timer.PERIODIC,
callback=Tic)

def Tic(timer):
    global N, flag
    if(N):
        flag = 1
        N -= 1
        LED.toggle()
        if(N == 1):
            tim.init(freq=5,
mode=Timer.ONE_SHOT, callback=Tic)
    else:
        LED.value(0)

pin1.irq(trigger=Pin.IRQ_FALLING, handler=Fire)

while(1):
    if(flag):
        flag = 0
        print(LED.value(), N)
```
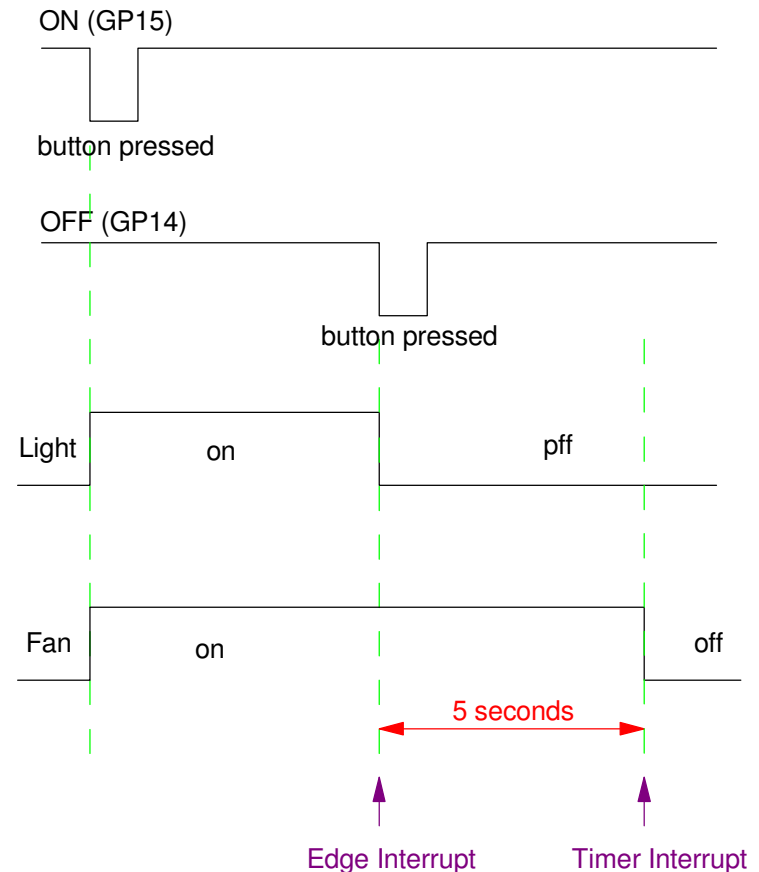
Toff: Bathroom Light & Fan:

The ONE_SHOT feature allows you to set up events N seconds ahead

Example: Bathroom light and fan

- When GP15 is pressed (on)
    - The bathroom light turns on, and
    - The bathroom fan turns on
- When GP14 is pressed (off)
    - The light turns off immediately
    - The fan turns off five seconds later

With ladder logic, this is called a Toff function

ON (GP15)

button pressed

OFF (GP14)

button pressed

Light   on   pff

Fan   on   off

5 seconds

Edge Interrupt   Timer Interrupt

Python Code

Three interrupts are used

- On
    - Falling edge interrupt
    - Turn on the light and fan
- Off
    - Falling edge interrupt
    - Turn off the light
    - Set up a timer interrupt in 5 seconds
- Tick
    - Timer interrupt
    - Turn off the fan

Note

- This is a fairly simple program using interrupts

```python
from machine import Pin, Timer
from time import sleep

Time = Timer()
N = 0
pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
pin2 = Pin(14,Pin.IN,Pin.PULL_UP)
LED  = Pin(16,Pin.OUT)
Fan  = Pin(17,Pin.OUT)

def On(pin1):
    LED.value(1)
    Fan.value(1)

def Off(pin2):
    LED.value(0)
    Time.init(freq=1/5, mode=Time.ONE_SHOT,
callback=Tick)

def Tick(timer):
    Fan.value(0)

pin1.irq(trigger=Pin.IRQ_FALLING, handler=On)
pin2.irq(trigger=Pin.IRQ_FALLING, handler=Off)


while(1):
    print(LED.value(), Fan.value())
    sleep(1)
```

Ton Function

- Turn on a light if a button is held down for X seconds

Sometimes you want to require a button is held down for X seconds

- Inadvertent button presses are ignored
- In ladder logic, this is called a Ton function

For example

- Turn on a light (GP16) if a button is held down for 5.00 seconds
  - Ignore button presses shorter than 5.00 seconds
- Turn off a light as soon as the button is released

GP15   pressed

        5.0 seconds

GP16

    < 5 seconds ignored
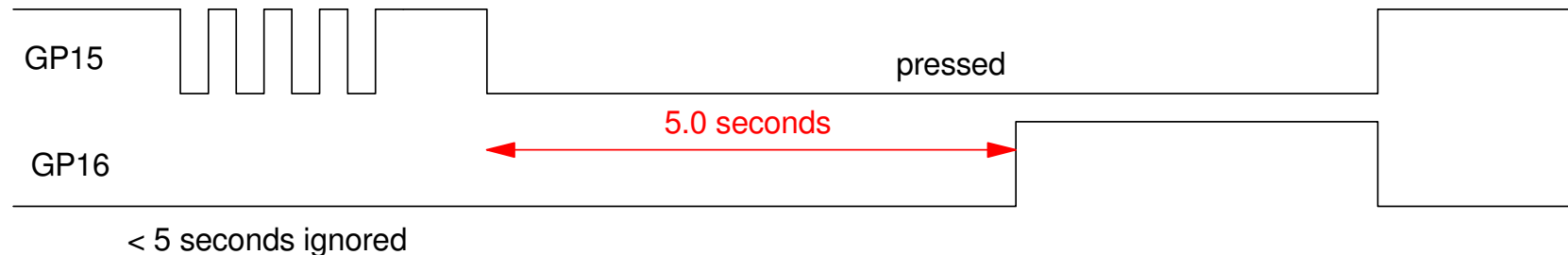
Ton Code (take 1)

There are several ways to do this.

Option #1:

- Set a timer for 10ms
- Set up a counter (TimeOn)
  - Clear the counter when button is released
  - Increment the counter when buttton is pressed
- If the counter exceeds 500 (5sec)
  - Turn on the light

- Note:
  - This code works, but it is inefficient (interupts are called every 10ms)
  - Timing is a little off (changes every 10ms)

```
from machine import Pin, Timer
import time

light = Pin(16, Pin.OUT)

tim = Timer()
TimeOn = 0

pin1 = Pin(15,Pin.IN,Pin.PULL_UP)

def Tick(timer):
    global TimeOn
    if(pin1.value() == 0):
        TimeOn += 1
    else:
        TimeOn = 0
        light.value(0)

    if(TimeOn >= 500):
        TimeOn = 500
        light.value(1)

tim.init(freq=100,
mode=Timer.PERIODIC, callback=Tick)

while (1):
    print(light.value(), TimeOn)
    time.sleep(0.1)
```

Ton Code (take 2)

- A better solution:
  - A little trickier
  - More precise

Use an edge interrupt (Light):

- If a rising edge (off)
  - turn off the light
  - Change the timer interrupt to call Light Off (light stays off)
- If a falling edge (on)
  - Turn on a timer interrupt in 5 seconds
  - Change the timer interupt to call LightOn
- Timer Interrupt
  - When timer interrupt kicks in, turn on or off the light
  - Only turns on if falling edge was detected 5.00 seconds ago

```python
# Ton Function (ver 2)

from machine import Pin, Timer
import time

tim = Timer()
pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
LED = Pin(16, Pin.OUT)

def Light(pin1):
    if(pin1.value() == 1):
        LED.value(0)
        tim.init(freq=5,
mode=Timer.ONE_SHOT, callback=LightOff)
    else:
        tim.init(freq=1/5,
mode=Timer.ONE_SHOT, callback=LightOn)

def LightOn(timer):
    LED.value(1)

def LightOff(timer):
    LED.value(0)

pin1.irq(trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING,
handler=Light)

while (1):
    print(light.value(), TimeOn)
    time.sleep(0.1)
```
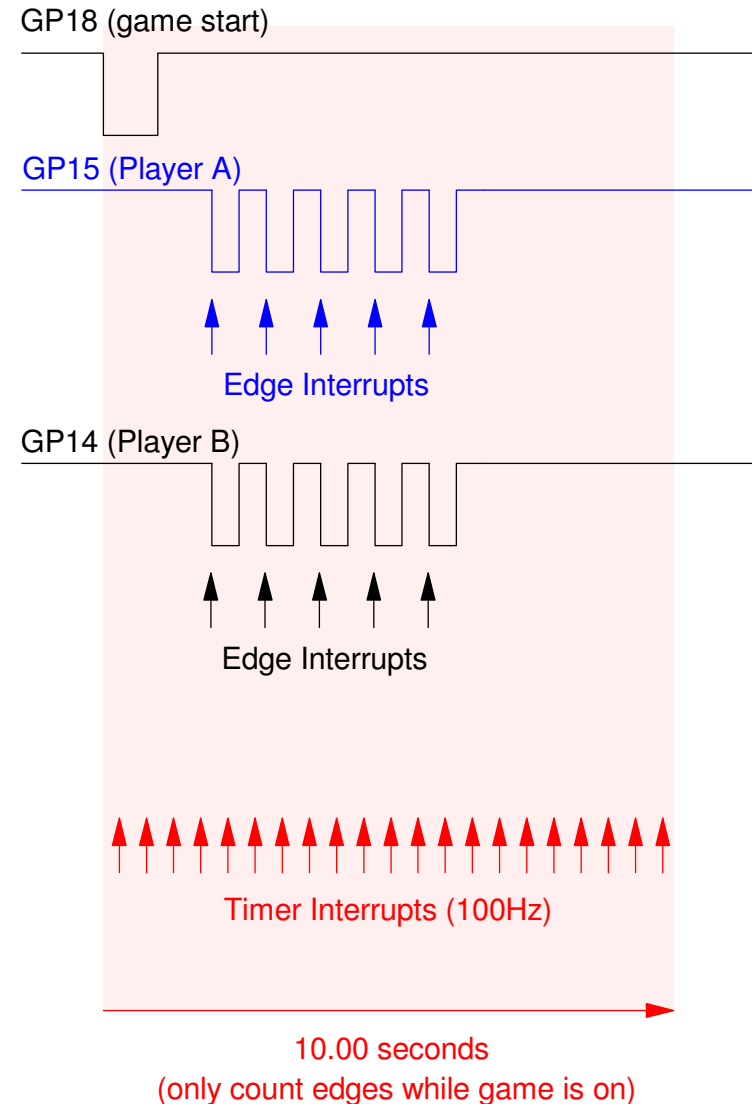
Hungry-Hungry Hippo (ver 3)

Use edge interrupts

- Start the game on GP18
  - Set the time to 10.00 seconds
- Count button presses for each player
  - Falling edges on GP15 and GP14

Use timer interrupts for game duration

- Timer interrupts every 10ms
  - Allows you to display the time remaining
  - Decrement time to 0.00, stopping at zero
- Score points while the game is ongoing
  - Stop counting edges once the game ends

GP18 (game start)

GP15 (Player A)

Edge Interrupts

GP14 (Player B)

Edge Interrupts

Timer Interrupts (100Hz)

10.00 seconds
(only count edges while game is on)

Hungry-Hungry-Hippo v3

- Edge Interrupts

Start the game on reset

- Reset counters to zero
- Reset game time to 10.00 seconds

Player1

- Falling edge interrupt
  - GP15 is pressed
- If the game is ongoing
  - Count button presses for player 1

Player2

- Falling edge interrupt
  - GP14 is pressed
- If the game is ongoing
  - Count button presses for player 2

```python
from machine import Pin, Timer
import time

pin0 = Pin(18,Pin.IN,Pin.PULL_UP)
pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
pin2 = Pin(14,Pin.IN,Pin.PULL_UP)

N1 = N2 = 0
Game Time = 1000

def player1(pin1):
    global N1
    global GameTime
    if(GameTime > 0):
        N1 = N1 + 1

def player2(pin2):
    global N2
    global GameTime
    if(GameTime > 0):
        N2 = N2 + 1

pin0.irq(trigger=Pin.IRQ_FALLING, handler=GameStart)
pin1.irq(trigger=Pin.IRQ_FALLING, handler=player1)
pin2.irq(trigger=Pin.IRQ_FALLING, handler=player2)
```

Hungry-Hungry-Hippo v3

- Timer Interrupts

Tic

- If the game is ongoing
  - Decrement time to zero
  - Toggle the LED
- Once the game is over
- Turn off the LED

The main routine

- Displays the score every 100ms

```python
:
LED = Pin(18,Pin.OUT)

def tic(timer):
    global led
    global GameTime
    if(GameTime > 0):
        GameTime -= 1
        led.toggle()
    else:
        led.off()

tim.init(freq=100, mode=Timer.PERIODIC, callback=tic)

while(1):
    while (GameTime > 0):
        print(GameTime*0.01, N1, N2)
        time.sleep(0.1)
    time.sleep(0.1)
```
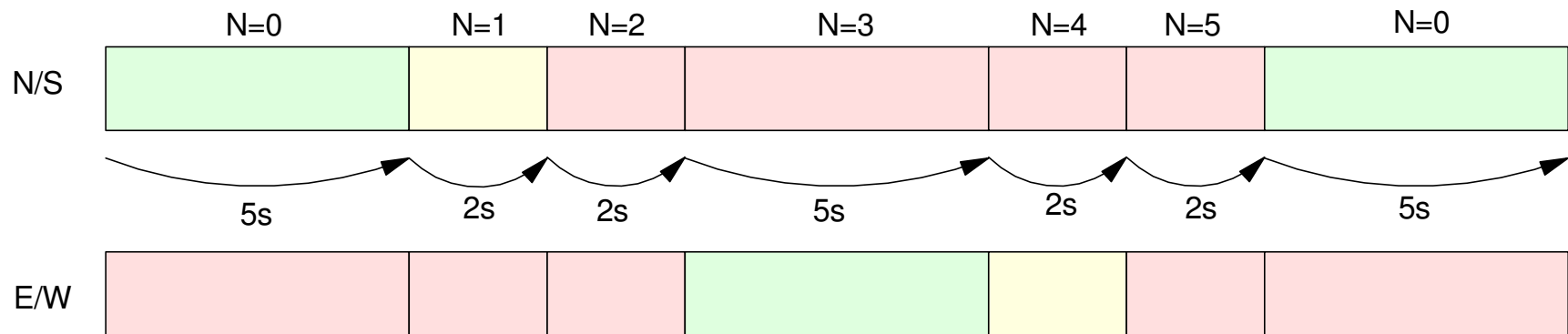
StopLight (ver 3)

With timer interrupts, you can set up the time for each color. Building upon the previous stoplight program, write a program that counts from N=0 to 6 with timing being:

Interrupts can be used to set these times several ways:

- A timer interrupt can be called every 100ms and keep track of time.
    - At certain times, the lights change (N increases), or
- Each time the light changes,
    - Set up the next interrupt X seconds in the future

Using the latter method:

```
# Stoplight (ver 3)

from machine import Pin, Timer
from time import sleep_ms

tim = Timer()
N = 0

def StopLight(pin1):
    global N
    N = (N + 1) % 6
    if(N == 0):
        tim.init(freq=1/5,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 1):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 2):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 3):
        tim.init(freq=1/5,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 4):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 5):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)

time.init(freq=1/5,mode=Timer.ONE_SHOT,callback=StopLight)

Time = 0
while(1):
    print(Time, N)
    Time += 1
    speep_ms(1000)
```
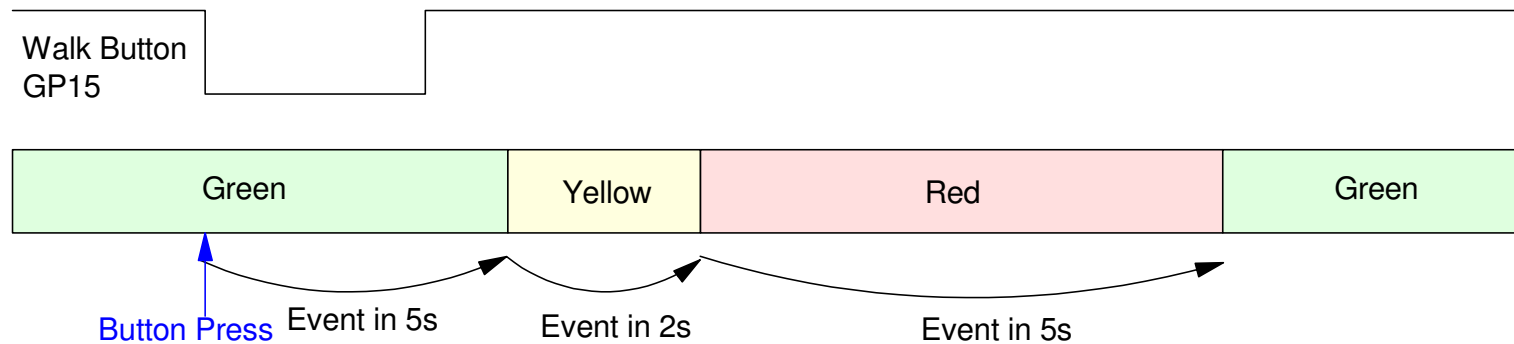
StopLight (ver 4): Adding a Walk button

A simple variation is to have the light always green E/W unless the walk button is pressed. Once that happens, go through the sequence of $N=0..5$ then stop again at 0 until the walk button is pressed again.

Walk Button
GP15

Green	Yellow	Red	Green

Button Press Event in 5s   Event in 2s   Event in 5s

Walk Button: Trigger a light change using one-time events (one-shot interrupts)

```python
from machine import Pin, Timer
from time import sleep_ms

tim = Timer()
N = 0

pin1 = Pin(15,Pin.IN,Pin.PULL_UP)

def Walk(pin1):
    tim.init(freq=1/5,mode=Timer.ONE_SHOT,callback=StopLight)

def StopLight(pin1):
    global N
    N = (N + 1) % 6
    if(N == 1):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 2):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 3):
        tim.init(freq=1/5,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 4):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)
    if(N == 5):
        tim.init(freq=1/2,mode=Timer.ONE_SHOT,callback=StopLight)

pin.irq(trigger=Pin.IRQ_FALLING,handler=Walk)

Time = 0
while(1):
    print(Time, N)
    Time += 1
    sleep_ms(1000)
```
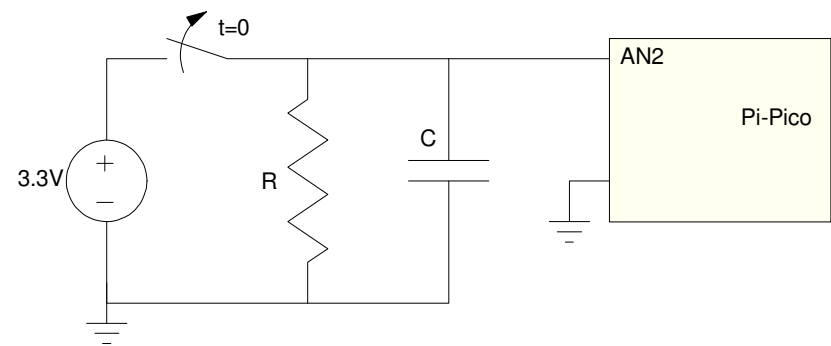
Setting a Fixed Sampling Rate with Timer Interrupts

Timer interrupts are a more precise way of setting the sampling rate.

- Example: measure the discharge of a capacitor
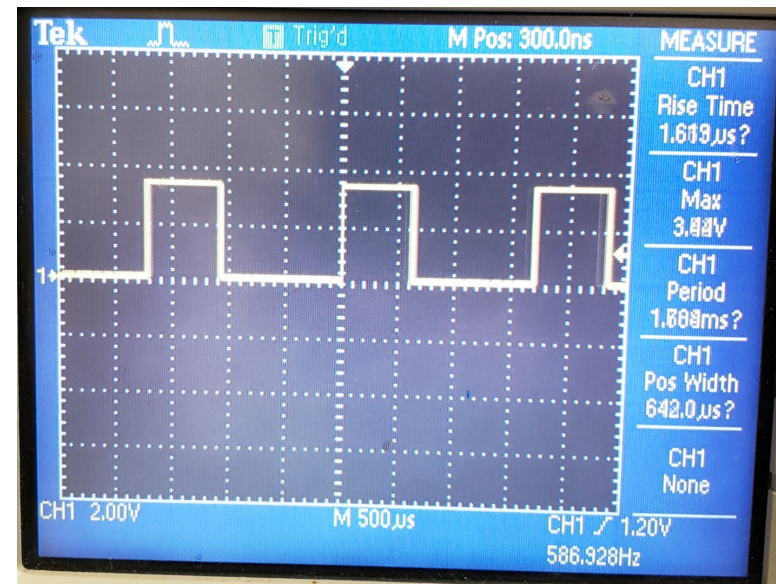- Sample the voltage every 1.00ms

time.sleep( ) works, but

- The time isn't exact
    - the total loop includes additional code
- sleep() wasts a lot of processor time

Example (following page)

- The sampling rate is 586.928Hz
- The sampling time is 1.703ms
    - should be 1.000ms

Better Solution:

- Set up a timer interrupt for 1.00ms
- Set a flag every time you interrupt

The main routine waits for the flat to be set

- When detected, run through the main loop
- Clear the flag, then
- Wait for it to be set again
  - every 1.000 ms

Note: The output square wave is 1.00018kHz

- The sampling rate is 1.00ms

Code:

- A flag is used
- Indicates 1ms has elapsed

```
from machine import ADC, Pin, Timer
from time import slee

a2d2 = machine.ADC(2
tim = Timer()

flag = 0
k = 3.3 / 65520
dT = 0.001

def tick(timer):
    global flag
    flag = 1

tim.init(freq=1000,

LED = Pin(16,Pin.OUT)

while(1):
    LED.value(0)
    while(flag == 0):
        pass
    LED.value(1)
    flag = 0

    a2 = a2d2.read_u16()
    V2 = k*a2

    print('{: 7.4f}'.format(V2))
```
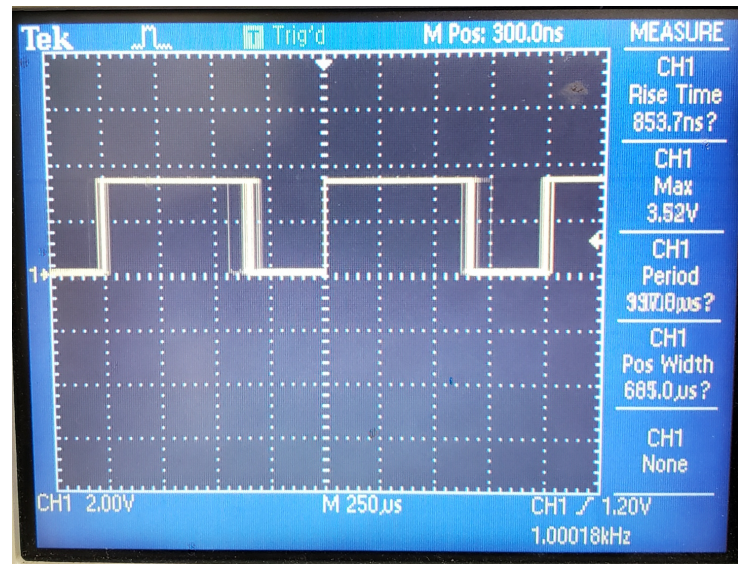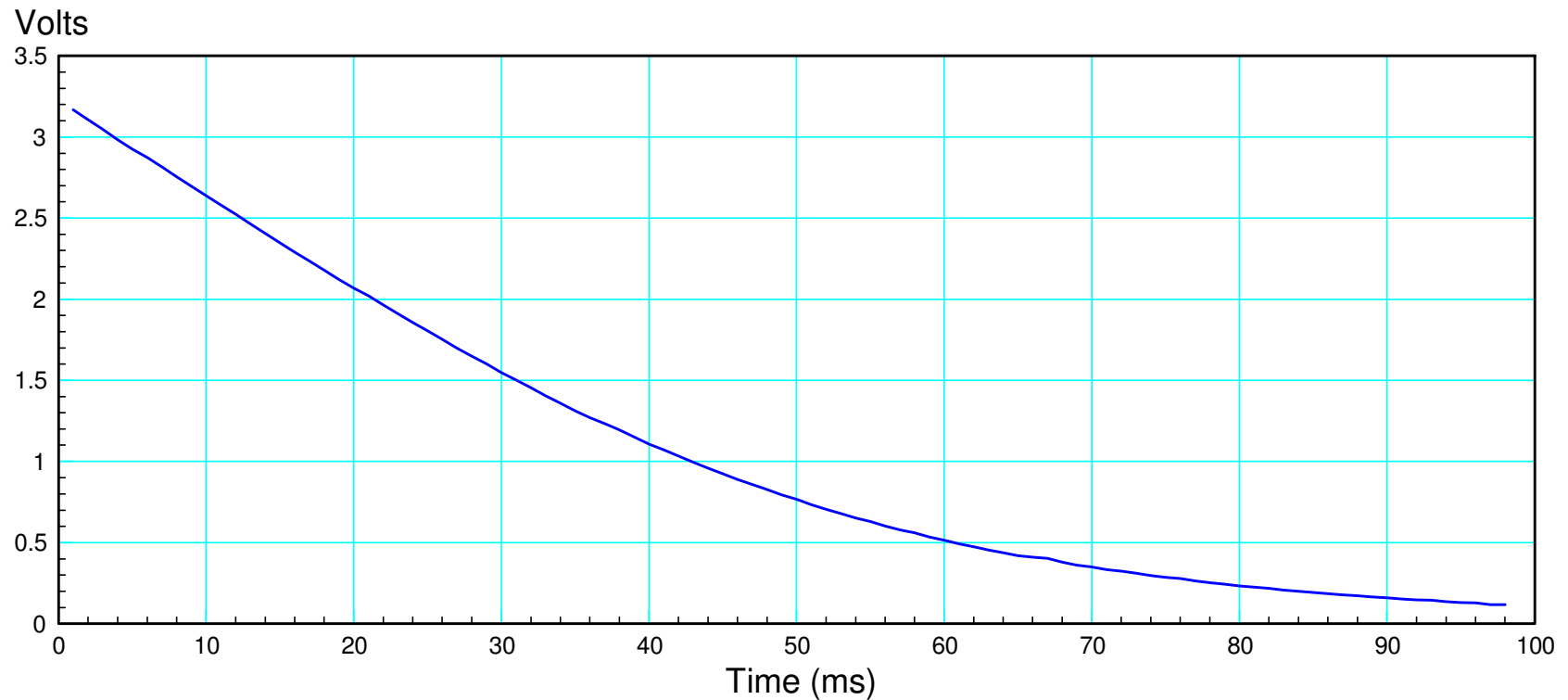
Fixed Sampling Rate: Example

Measure the voltage across a capacitor as it discharges

- Sample the voltage every 1.00ms
- Result is a clean exponential decay

Digital Filters

Yet another application of timer interrupts is implementing digital filters. With a digital filter, you *have* to know the sampling rate, T: the conversion from the s-plane to the z-plane depends upon knowing T. If T changes, the filter is wrong.

For example, design a digital filter to implement

$$Y = \left(\frac{50}{(s+5+j5)(s+5-j5)} \right) X = \left(\frac{50}{s^2+10s+50} \right) X$$

One way to do this is to convert to the z-plane

$$z = e^{sT}$$

Assuming a sampling rate of 10ms (T = 0.01), the poles in the s-plane convert to the z-plane as

$$s = -5 + j5 \qquad\qquad z = e^{sT} = 0.9500 + j0.0475$$

$$s = -5 - j5 \qquad\qquad z = e^{sT} = 0.9500 - j0.0475$$

so a discrete-version of G(s) would be

$$Y = \left(\frac{k(z+1)^2}{(z-0.9500-j0.0475)(z-0.9500+j0.0475)} \right) X$$

where k is chosen to set the DC gain to 1.00 (same as G(s)). Multiplying out

$$Y = \left(\frac{0.001209\left(z^2+2z+1\right)}{z^2-1.900z+0.904837} \right) X$$

In Code...

```python
from machine import ADC, Pin, Timer

a2d2 = machine.ADC(2)
tim = Timer()

flag = 0
k = 3.3 / 65520
x0 = x1 = x2 = y0 = y1 = y2 = 0

def tick(timer):
    global flag
    flag = 1

tim.init(freq=100, mode=Timer.PERIODIC, callback=tick)
LED = Pin(16,Pin.OUT)
while(1):
    LCD.value(0)
    while(flag == 0):
        pass
    LED1.value(1)
    flag = 0

    a2 = a2d2.read_u16()
    V2 = k*a2

    x2 = x1
    x1 = x0
    x0 = k*a2

    y2 = y1
    y1 = y0
    y0 = 1.9008*y1 − 0.90483*y0 + 0.001209*(x0 + 2*x1 + x2);

    print('{: 7.4f}'.format(x0), '{: 7.4f}'.format(y0))
```
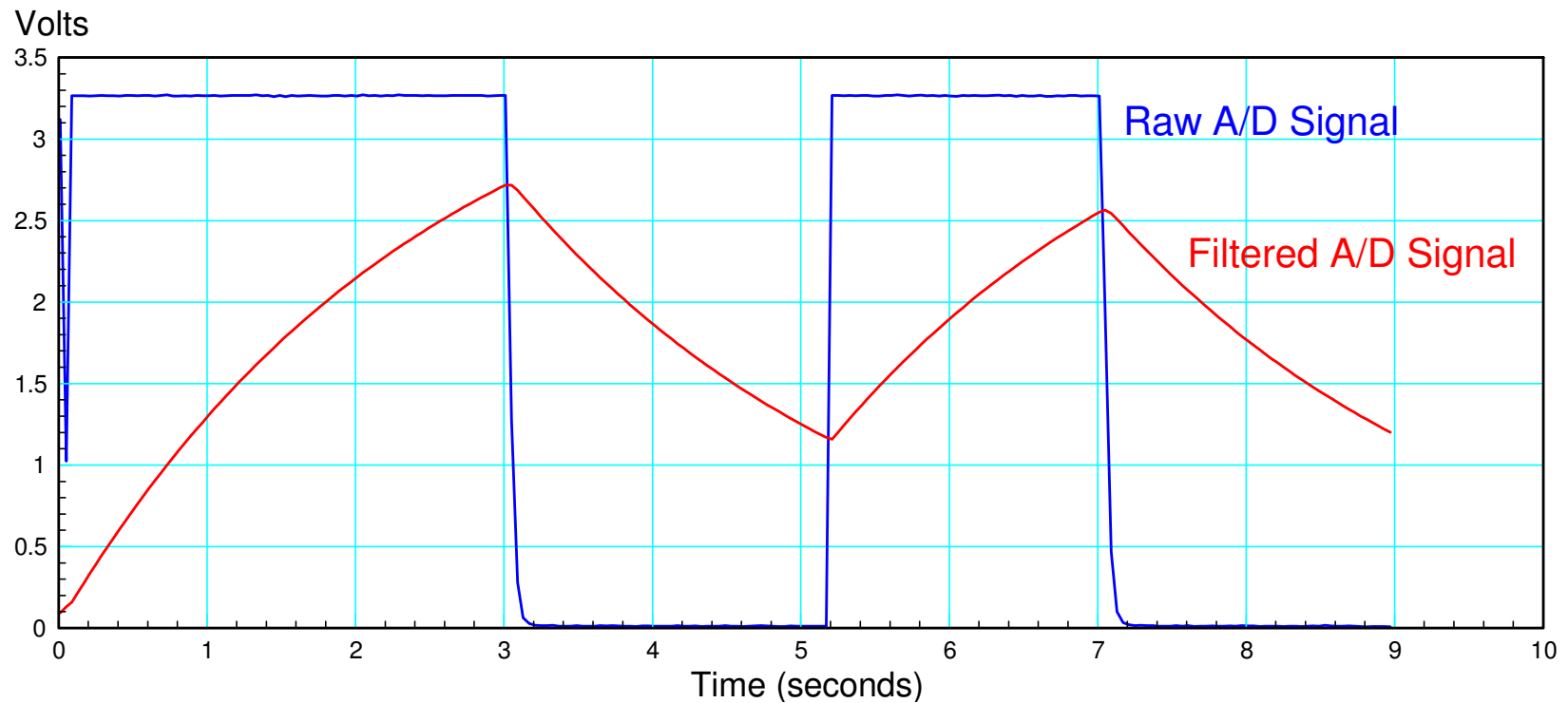
Result when filtering a square wave:

- Filter has poles at s = -5 + /- j5
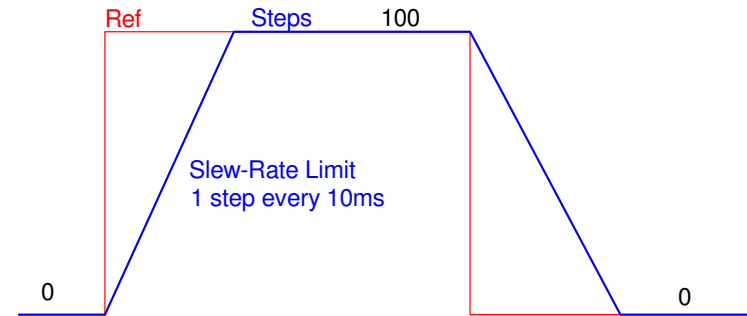- Shows up as exponential rise and fall

Result of filtering the A/D signal

Stepper Motor Controls

Finally, timer interrupts are also useful for stepper motors

Every 10ms (Timer interrupt):

- Step forward one step if Step < Ref
- Step back one step if Step > Ref
- No nothing if Step == Ref

By placing this inside the interrupt

- The main routine is free to do anything
- The timing for the stepper motor won't be affected

```python
from machine import Pin, Timer
import time

tim = Timer()
Step = 0
Ref = 0

pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
pin2 = Pin(14,Pin.IN,Pin.PULL_UP)

Sa = Pin(6,Pin.OUT)
Sb = Pin(7,Pin.OUT)
Sc = Pin(8,Pin.OUT)
Sd = Pin(9,Pin.OUT)

def Stepper(Step):
    X = Step % 4;
    if(X == 0):
        Sa.value(1)
        Sb.value(0)
        Sc.value(0)
        Sd.value(0)
    if(X == 1):
        Sa.value(0)
        Sb.value(1)
        Sc.value(0)
        Sd.value(0)
    if(X == 2):
        Sa.value(0)
        Sb.value(0)
        Sc.value(1)
        Sd.value(0)
    if(X == 3):
        Sa.value(0)
        Sb.value(0)
        Sc.value(0)
        Sd.value(1)

def tick(timer):
    global Step, Ref
    if(Step < Ref):
        Step += 1
    if(Step > Ref):
        Step -= 1
    Stepper(Step)

tim.init(freq=100,
mode=Timer.PERIODIC, callback=tick)

while (1):
    if(pin1.value() == 0):
        Ref = 0
    if(pin2.value() == 0):
        Ref = 100
    print(Ref, Step, )
    time.sleep(0.1)
```

Summary

The Pi-Pico is capable of timer interrupts as well

Timer interrupts

- Allow you to measue time
- Allow you to set a fixed sampling rate
- Allow you to trigger routines N seconds in the future

Again, interrupts are confusing

- If you can figure them out, some programs become much simpler to write

References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- https://learn.pimoroni.com/article/getting-started-with-pico
- https://www.w3schools.com/python/default.asp
- https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
- https://docs.micropython.org/en/latest/library/index.html
- https://www.fredscave.com/02-about.html

Pi-Pico Breadboard Kit

- https://wiki.52pi.com/index.php?title=EP-0172

Other

- https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/
- https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/
- https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/