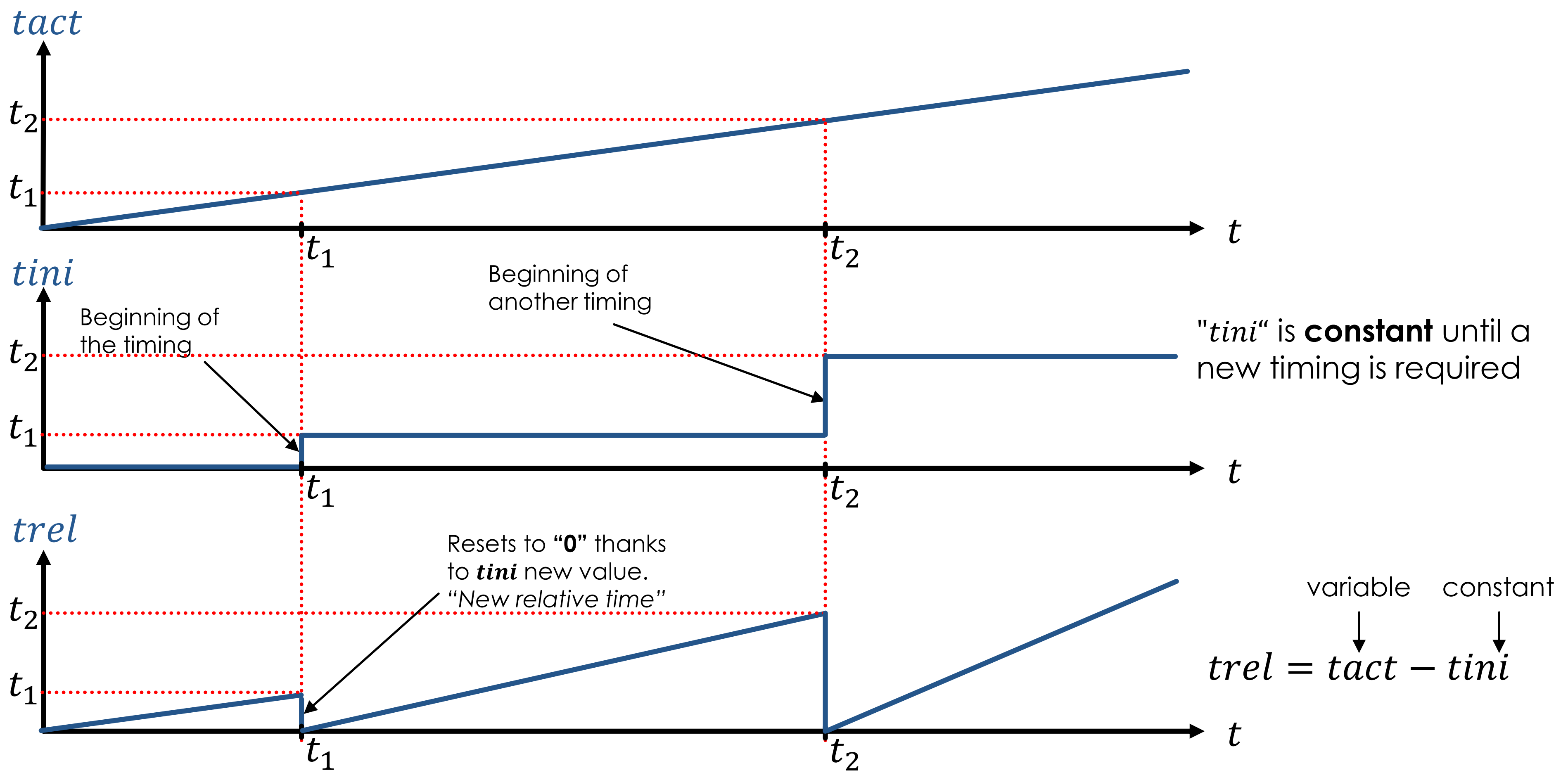
Workshop Basic Raspberry Class 2 – Relative Timing and FSM with Raspberry

MSc. David Velásquez Rendón

Contents

Relative timing in Python using time.time()

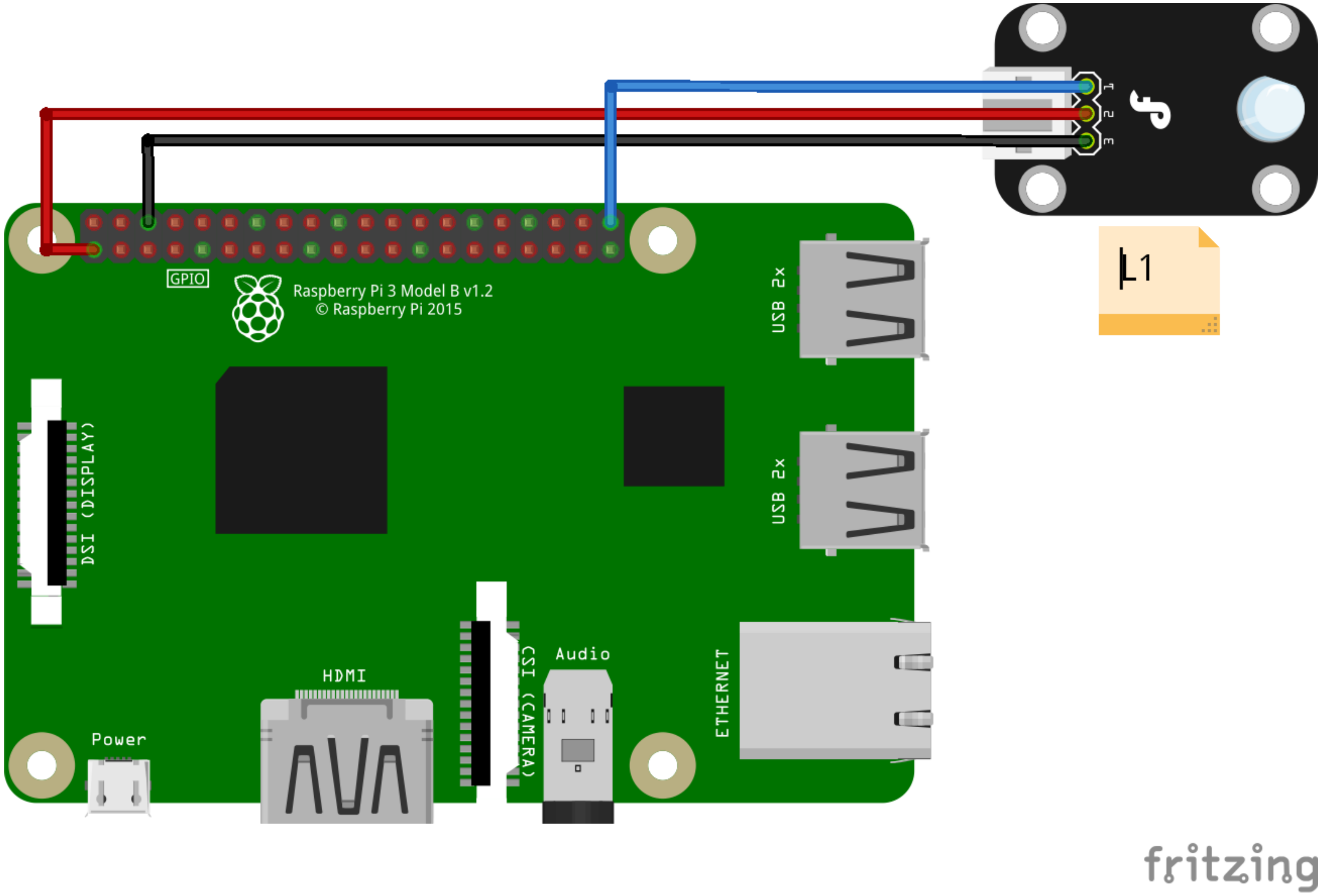
- The function `var = time.time();` returns the total execution time in seconds since the Python program was executed.
- Allows to do relative timing calculations without using delays with the same behavior as millis() function in Arduino:



Example 2.1 – Relative timing in Raspberry with time.time()

UNIVERSIDAD EAFIT

- Do a Python program that allows to blink a LED (L1) connected on PIN 40 (GPIO21) ½ sec ON and ½ sec OFF using the time.time() function.



Example 2.1 – Relative timing in Raspberry with time.time()


```
#Library declaration
import RPi.GPIO as GPIO
import time

#I/O pin labeling
L1 = 40 #Label LED connected in pin 40 as “L1”

#Constant declaration
TBLINK = 0.5 #Blink constant TBLINK initialized on 0.5s

#Variable declaration
tact = 0 #Actual time (tact)
tini = 0 #Initial time (tini)
trel = 0 #Relative time (trel)

#SETUP
#I/O Pin Configuration
GPIO.setmode(GPIO.BOARD) #Configures all pins reference using pin #
GPIO.setup(L1, GPIO.OUT) #Set pin L1 as Output
#Output cleaning
GPIO.output(L1,0) #Turn OFF L1 (also posible GPIO.output(L1,False))
#Reset first time
tini = time.time() #Reset tini to current time

#EXECUTION
while True:
	tact = time.time()
	trel = tact - tini  #Calculate the relative time
	if trel < TBLINK: #If relative time (trel) is less than the blinking time constant (TBLINK)
		GPIO.output(L1,1) #Turn ON L1
	elif trel < TBLINK: #If trel is greater than blinking time constant but less than blinking time x 2 (1/2 sec ON and ½ sec OFF)
		GPIO.output(L1,0) #Turn OFF L1
	else: #In other case (if trel is greater than 2 times the blinking time constant)
		tini = time.time()  #Take a new initial time in order to begin again the blinking cycle (reset rel time to 0 in next iteration)
```

Python Program Structure for HW

UNIVERSIDAD EAFIT

Library declaration(e.g: `import RPi.GPIO as GPIO`)

FSM States Labeling (e.g: `SINI = 0`)

I/O Pin Labeling (e.g: `LEDPIN = 36`)

Constant declaration (e.g: `CONTMAX = 10`)

Variable declaration (e.g: `temperature = 0.0`)

- CURRENT STATE Variable Declaration (`state = SLEDOFF`)
- TIMING VARIABLES Declaration (e.g `tini = 0.0`)

Subroutines or functions declaration:

The FSM needs to be described as dictionary definitions as follows:

```
def FSLEDOFF():                    #Initial State LED OFF
       #Physical Outputs State
       GPIO.output(L1,0) #Turn OFF L1
       #Internal Variable Computations
       #->Relative time calculation
       trel = tact - tini
       #Transition Questions
       if trel >= BLINK:
              state = SLEDON #Change state
              tini = time.time() #Reset tini to current time
```

```
def FSLEDON():                                #State LED ON
       #Physical Outputs State
       GPIO.output(L1,1) #Turn ON L1
       #Internal Variable Computations
       #->Relative time calculation
       trel = tact - tini
       #Transition Questions
       if trel >= BLINK:
              state = SLEDOFF #Change state
              tini = time.time() #Reset tini to current time

FSM = {0: FSLEDOFF,
       1: FSLEDON,
}
```

Pin configuration and cleaning:

```
#SETUP
#CONFIGURATION: Indicate which pins are inputs and which are outputs
#->setmode and setup functions must be used for this part
#CLEANING: For safety, it is important to clean used outputs with the purpose that they are turned off at the beginning of the program. Use the function GPIO.output(PIN,False).
#COMMUNICATIONS: For example, for communications with Arduino, import Serial library at Library declaration and use the function ser = serial.Serial("/dev/ttyACM0", 9600) to begin this communications.
```

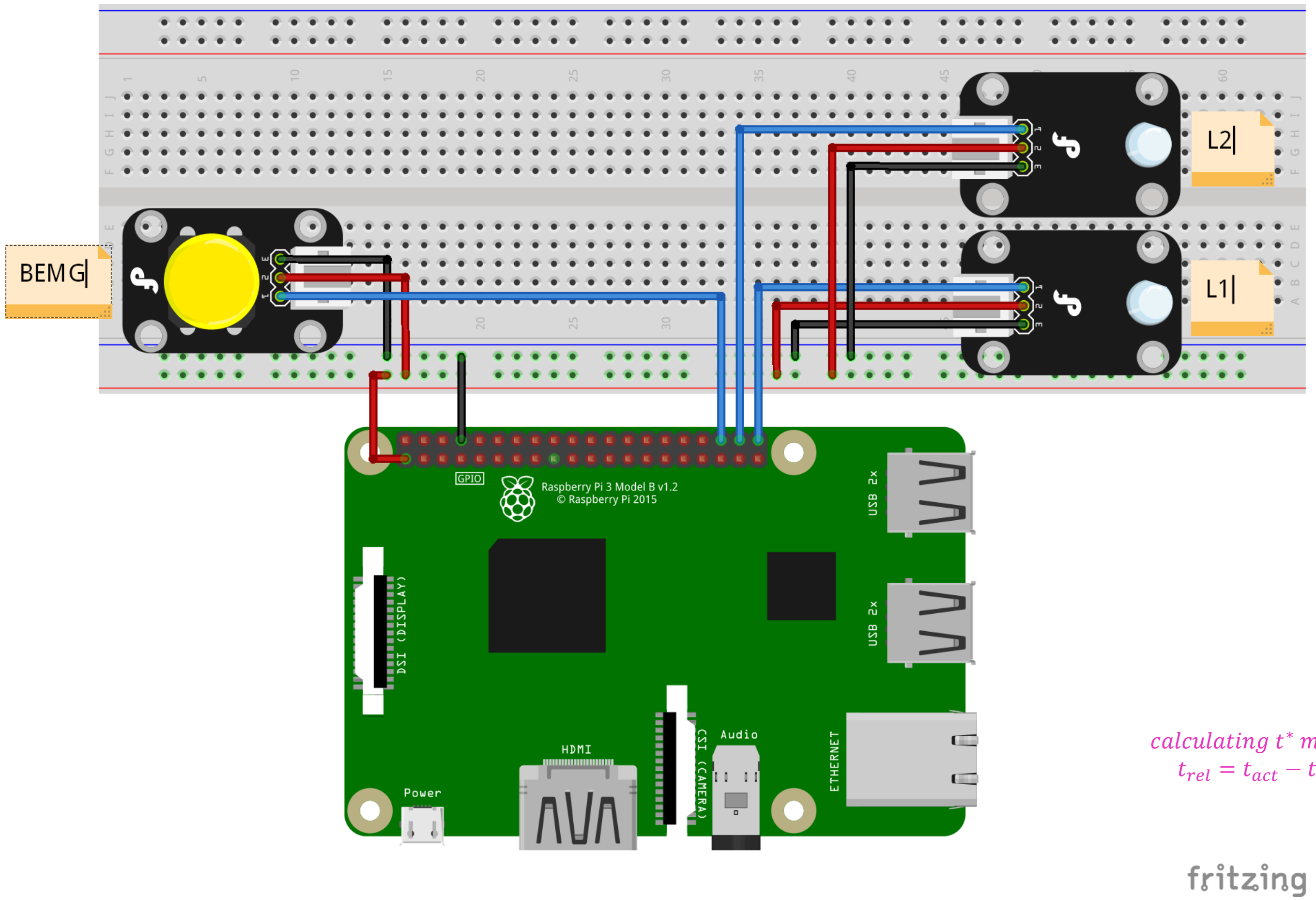
Infinite loop (Main program - Execution):

```
#EXECUTION
while True:
       #Main program
       FSM[state]();
```

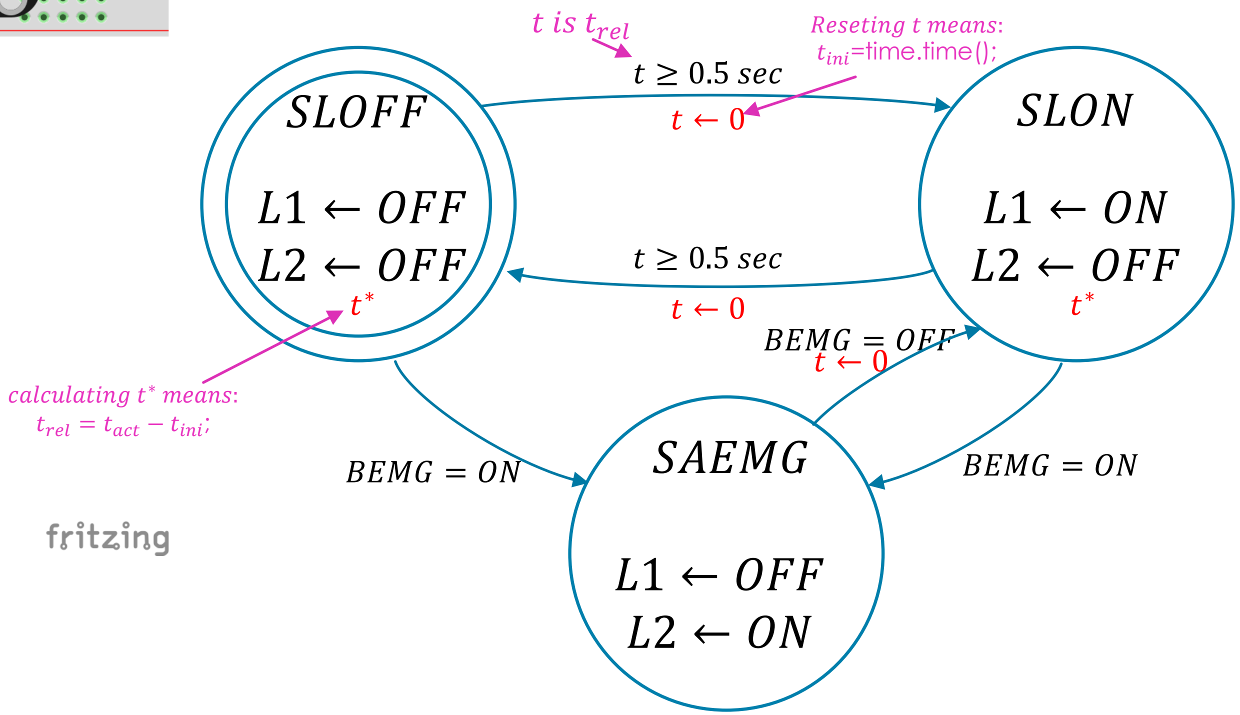
Example 2.2 – FSM with Python and Raspberry

UNIVERSIDAD EAFIT

- Do an Python program that blinks a LED (***L1***) connected on **pin 40**, **½ sec ON** and **½ sec OFF** while the emergency button is not activated (***BEMG***) connected on **pin 36**. If ***BEMG*** is ON, the LED ***L1*** remains **OFF** and the LED ***L2*** connected on **pin 38** turns **ON**. When there isn't anymore an emergency, the process returns to its normal blinking.



ENTRADAS			SALIDAS			
Nombre	Descripción	Tipo	Nombre		Descripción	Tipo
BEMG	Emergency button	Boolean (Digital)	*L1*		LED	Boolean (Digital)
			L2		Red LED for emergency	Boolean (Digital)
			t	t_{act}	Timer	Internal Variable
				t_{ini}		
				t_{rel}		



Example 5.2 – FSM with Python and Raspberry


```python
#Library declaration
import RPi.GPIO as GPIO
import time

#States
SLEDOFF = 0 #State LED OFF
SLEDON = 1 #State LED ON
SAEMG = 2 #State alarm

#I/O Pin definition
L1 = 40 #LED L1 connected on pin #40
L1 = 38 #LED L2 connected on pin #38
SW = 36 #SW connected on pin #36

#Constants definition
BLINK = 0.5 #Blink time constant 0.5 secs

#Variable definition
tact = 0.0 #Actual time variable
tini = 0.0 #Initial time variable
trel = 0.0 #Relative time variable
state = SLEDOFF #Initial state

#Subroutines and functions
#FSM
def FSLEDOFF():
    global tini
    global state
    #Outputs state
    GPIO.output(L1, 0) #Turn OFF L1
    GPIO.output(L2, 0) #Turn OFF L2
    #Variables Computation
    trel = tact - tini
    #Transition questions
    if trel >= BLINK:
        state = SLEDON #Change state
        print("State: SLEDON")
        tini = time.time()
    elif GPIO.input(SW) == 1:
        state = SAEMG
        print("State: SAEMG")

def FSLEDON():
    global tini
    global state
    #Outputs state
    GPIO.output(L1, 1) #Turn ON L1
    GPIO.output(L2, 0) #Turn OFF L2
    #Variables Computation
    trel = tact - tini
    #Transition questions
    if trel >= BLINK:
        state = SLEDOFF #Change state
        print("State: SLEDOFF")
        tini = time.time()
    elif GPIO.input(SW) == 1:
        state = SAEMG
        print("State: SAEMG")

def FSAEMG():
    global tini
    global state
    #Outputs state
    GPIO.output(L1, 0) #Turn OFF LED
    GPIO.output(L2, 1) #Turn ON L2
    #Transition questions
    if GPIO.input(SW) == 0:
        state = SLEDOFF
        print("State: SLEDOFF")
        tini = time.time()

FSM = {0: FSLEDOFF,
       1: FSLEDON,
       2: FSAEMG,
}

#Configuration
#IO Pin Setup
GPIO.setmode(GPIO.BOARD) #Set pin to board number
GPIO.setup(L1, GPIO.OUT) #LED L1 as output
GPIO.setup(L2, GPIO.OUT) #LED L2 as output
GPIO.setup(SW, GPIO.IN) #SW as input
#Output cleaning
GPIO.output(L1, 0) #Turn off L1
GPIO.output(L2, 0) #Turn off L2
#Reset tini
tini = time.time() #Reset tini time

#Execution
while True:
    tact = time.time() #Acquire actual time
    FSM[state]() #Execute FSM
```

Thanks!