Workshop Basic Arduino Class 5 – Relative Timing and FSM with Arduino

MSc. David Velásquez Rendón

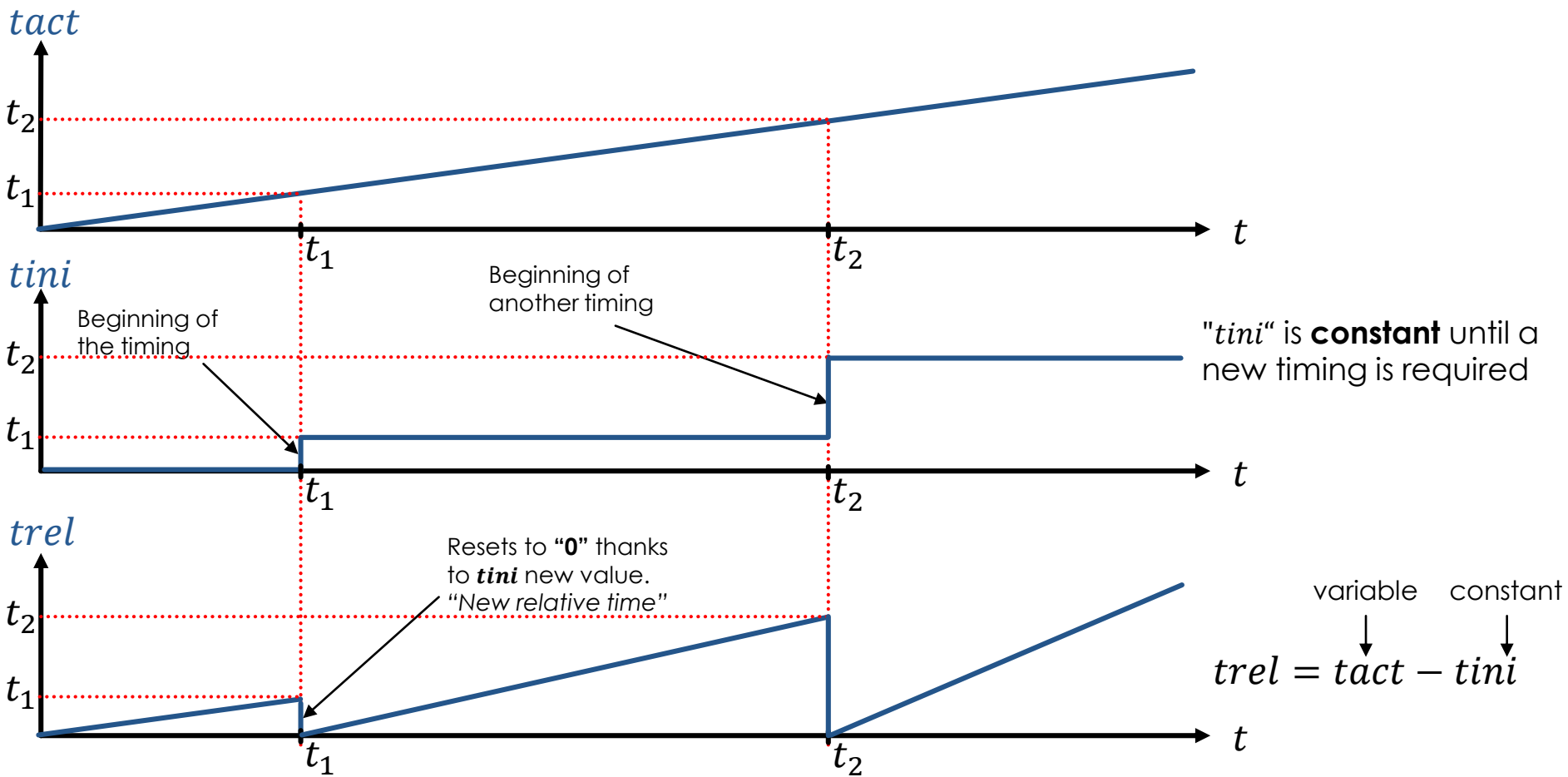
UNIVERSIDAD EAFIT®

Contents

Relative timing in Arduino using millis()

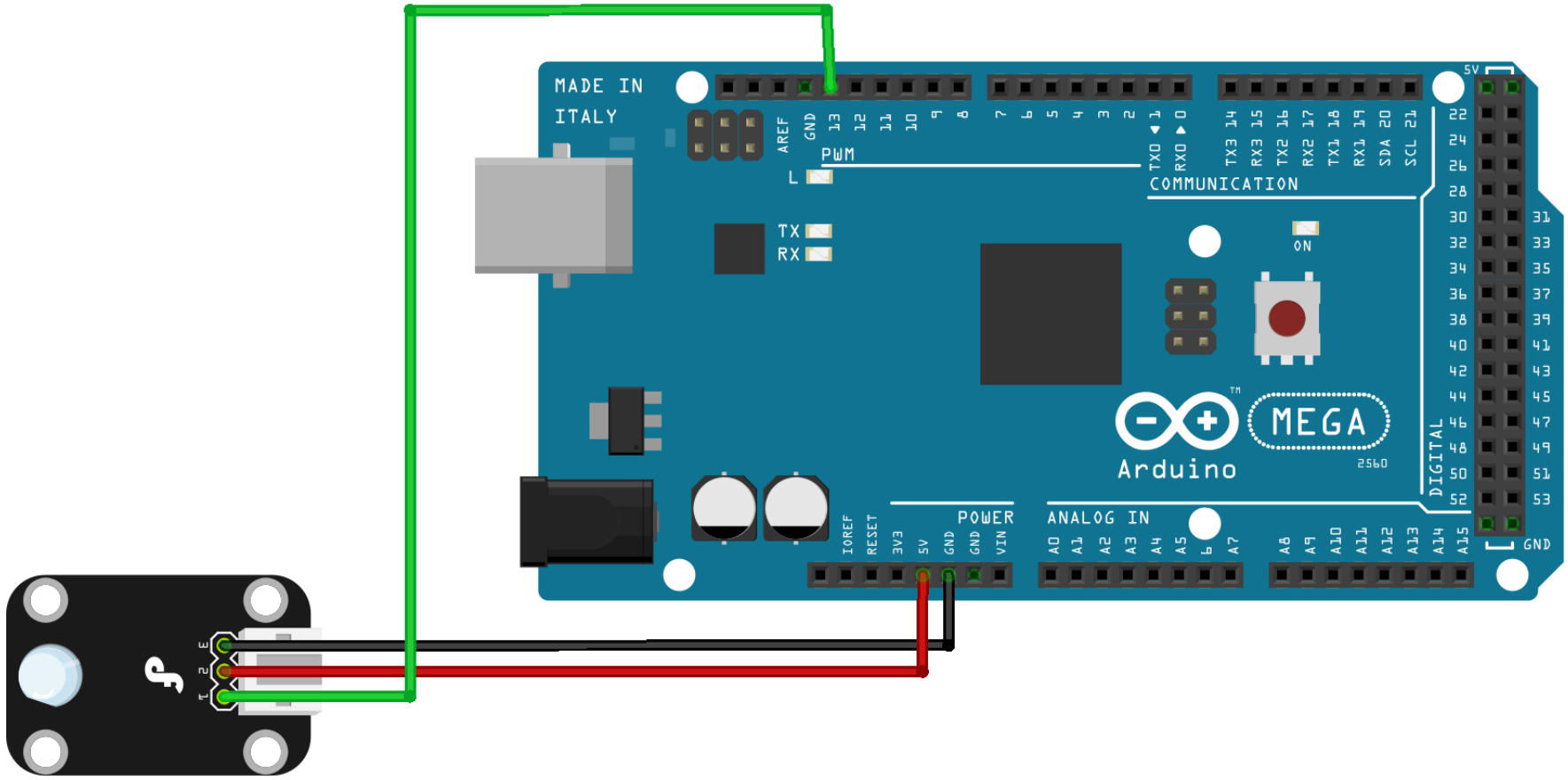
- The function `var = millis();` returns the total execution time in milliseconds since the Arduino was turned ON.
- Allows to do relative timing calculations without using delays.
- To do this relative calculations, **three variables** must be taken into account:
  - ***tact*:** It's the actual execution time. This variable needs to be refreshed always and is exactly equal to what is returned by `millis()`. In short terms, ***tact*** = `millis();`
  - ***tini*:** It's the initial time for relative timing count. Similar to push Start in a chronometer to begin the timing. This variable is only assigned once, at the **beginning** when it's desired to do the timing calculation.
  - ***trel*:** Relative time. It's the subtraction between the ***tact*** and ***tini***. In short terms, $trel = tact - tini$. This variable allows to know exactly how much time has passed since the beginning of the relative timing calculation.

Note: If it's required to compute more precise relative timings, use the function `var = micros();` *which returns the execution time in microseconds.*



Example 5.1 – Relative timing in Arduino with millis()

- Do an Arduino program that allows to blink a LED (L1) ½ sec ON and ½ sec OFF using the millis() function.



Example 5.1 – Relative timing in Arduino with millis()


```
//I/O Pin Labeling
#define L1 13 //L1 connected on pin 13

//Constants declaration
const unsigned long TBLINK = 500; //Blinking time constant (TBLINK) as unsigned long and initialized on 500 miliseconds

//Variables declaration
unsigned long tact = 0; //Actual time (tact) as unsigned long
unsigned long tini = 0;  //Initial time (tini) as unsigned long
unsigned long trel = 0; //Relative time (trel) as unsigned long

void setup() {
  //I/O Pin Configuration
  pinMode(L1, OUTPUT);  //L1 as OUTPUT

  //Physical Output Cleaning
  digitalWrite(L1, LOW);  //Turn OFF L1
  tini = millis();  //Initialize for the first time the tini variable because the first relative timing calculation will take place in the void loop
}

void loop() {
  tact = millis(); //Take always the actual execution time
  trel = tact - tini;  //Calculate the relative time
  if (trel < TBLINK) { //If relative time (trel) is less than the blinking time constant (TBLINK)
    digitalWrite(L1, HIGH);  //Turn ON L1
  }
  else if (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)
    digitalWrite(L1, LOW);  //Turn OFF L1
  }
  else { //In other case (if trel is greater than 2 times the blinking time constant)
    tini = millis();  //Take a new initial time in order to begin again the blinking cycle (reset rel time to 0 in next iteration)
  }
}
```

FSM Template - Arduino

LIBRARIES Declaration (e.g: `#include <SFEMP3Shield.h>`)

FSM STATES Labeling (e.g: `#define SINI 0`)

I/O PINS Labeling (e.g `#define LED 13`)

CONSTANTS Declaration (e.g `const unsigned int MAXCYCLES = 6;`)

VARIABLES Declaration (e.g `float temperature = 0;`)

- **CURRENT STATE Variable Declaration** (`unsigned int state = SINI;`)
- **TIMING VARIABLES Declaration** (e.g `unsigned long tini = 0;`)

SUBROUTINES OR FUNCTIONS Declaration (e.g `void blink()`) (e.g `unsigned int sum(unsigned int A, unsigned int B)`)

CONFIGURATION

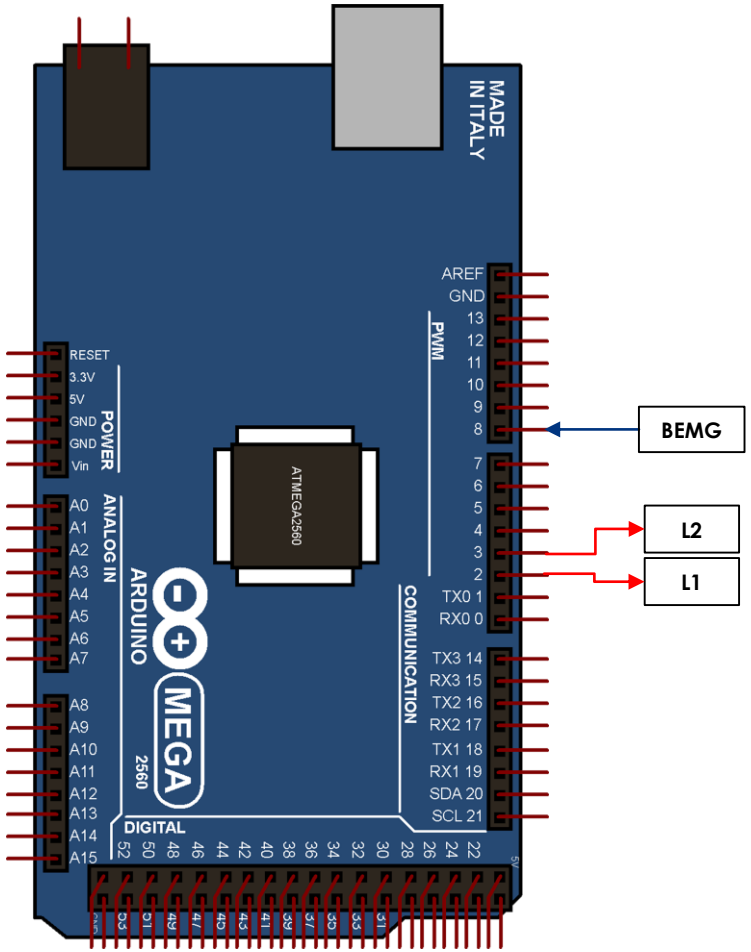
```
void setup() {
        //I/O Pin Configuration
        //Physical Outputs Cleaning
        //Communications
        tini = millis();  //(Optional) First assignation of tini time (Only if it's required, if the first relative timing calculation will take place at the beginning of the
        void loop
}
```

EXECUTION OR RUN-TIME

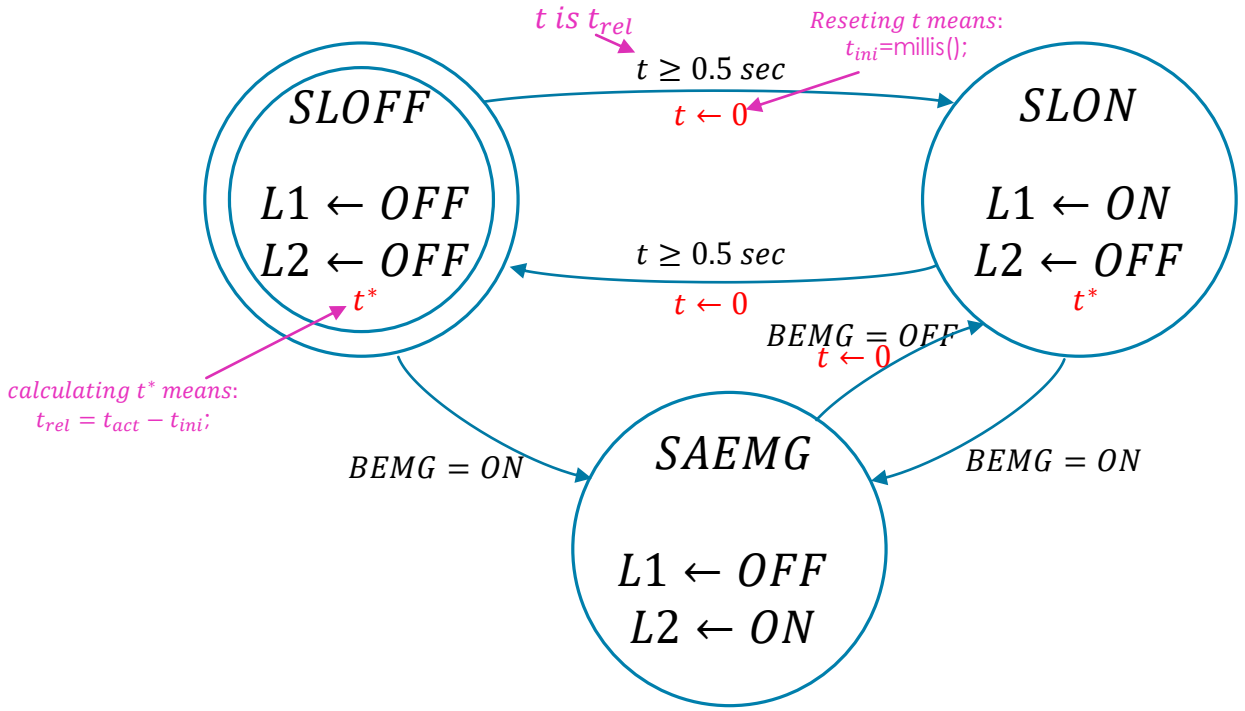
```
void loop() {
        tact = millis(); //Calculate the actual execution time (if FSM requires
        timing)
        //FSM
        switch (state) {
                case SINI:  //Initial State
                  //Physical Outputs state assignation
                  digitalWrite(L1, LOW);  //Turn OFF LED L1 in the SINI state

                  //Internal Variables Computation (mapping, readings, etc)
                  //->Relative timing calculation (if required by the state)
                  trel = tact - tini;

                  //Transition questions
                  if (trel >= VALUE) { //If relative time is greater or equal than
                VALUE constant
                      state = SLEDON;  //Next state is SLEDON (State LED ON)
                      tini = millis();   //Reset the relative timing by taking a new
                tini time (Only necessary if it's required to do timing in the next
                state
                  }
                  else if (digitalRead(BTNEMG) == HIGH) {  //If the emergency button
                is ON
                      state = EALERT;  //Next state is SALERT
                  }
                break;

                case ELEDON:
                  //Physical Outputs state assignation …
                break;
        }
}
```

Example 5.2 – FSM with Arduino

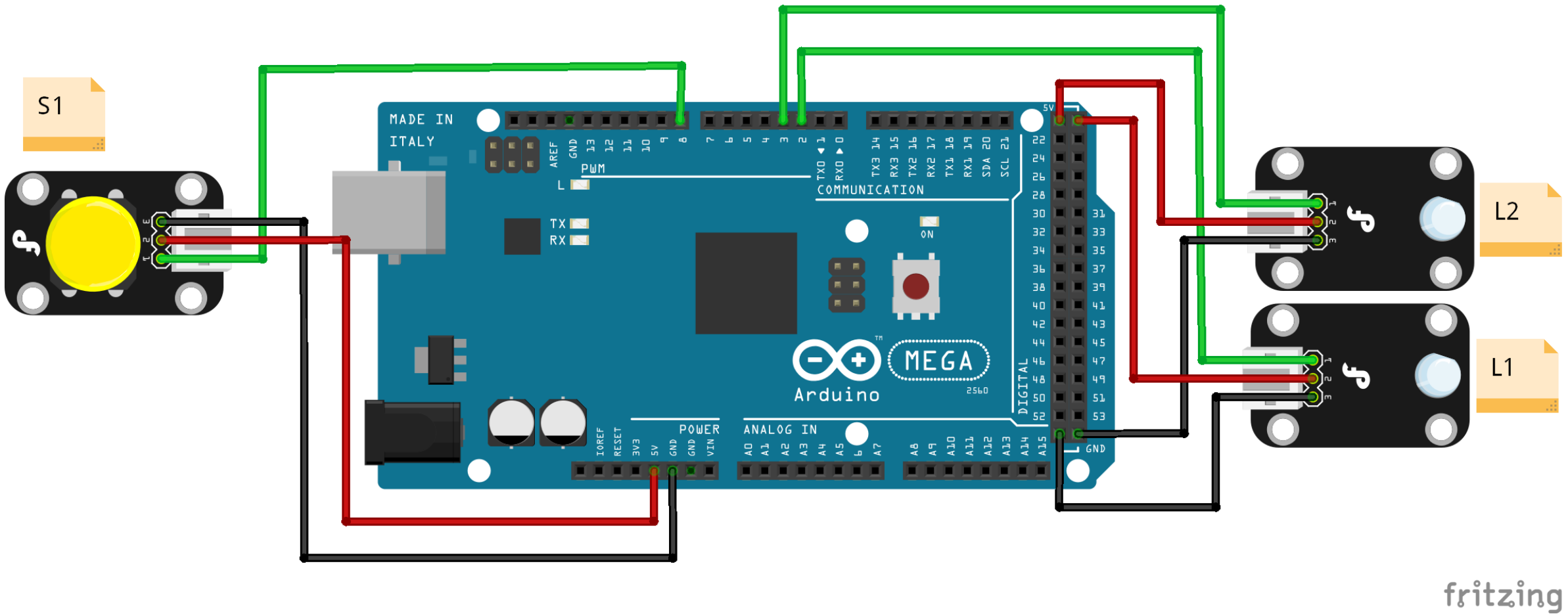
- Do an Arduino program that blinks a LED (***L1***) connected on **pin 2**, **½ sec ON** and **½ sec OFF** while the emergency button is not activated (***BEMG***) connected on **pin 8**. If ***BEMG*** is ON, the LED ***L1*** remains **OFF** and the LED ***L2*** connected on **pin 3** turns **ON**. 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} t_{ini} t_{rel}	Timer	Internal Variable



Example 5.2 – FSM with Arduino

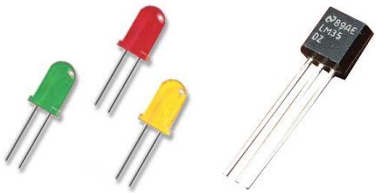


Example 5.2 – FSM with Arduino


```
//FSM States Labeling
#define SLOFF 0 //State LED OFF
#define SLON 1 //State LED ON
#define SAEMG 2 //State Alert Emergency

//I/O Pin Labeling
#define BEMG 8  //BEMG connected on pin 8
#define L1 2  //L1 connected on pin 2
#define L2 3  //L2 connected on pin 3

//Constants declaration
const unsigned long TBLINK = 500;  //Define blinking time constant as unsigned long and
initialize it in 500 milisecs

//Variables declaration
unsigned int state = SLOFF;  //Variable for storing the current FSM state and initialized
in SLOFF
//->Timing Vars
unsigned long tact = 0; //Actual time (tact) as unsigned long
unsigned long tini = 0; //Initial time (tini) as unsigned long
unsigned long trel = 0; //Relative time (trel) as unsigned long

void setup() {
  //I/O Pin Configuration
  pinMode(BEMG, INPUT); //BEMG as INPUT
  pinMode(L1, OUTPUT);  //L1 as OUTPUT
  pinMode(L2, OUTPUT);  //L2 as OUTPUT

  //Physical Output Cleaning
  digitalWrite(L1, LOW);  //Turn OFF L1
  digitalWrite(L2, LOW);  //Turn OFF L2
  tini = millis();  //Initialization of tini (first state ELOFF requires timing)
}

void loop() {
  tact = millis(); //Refresh always actual time
  //FSM
  switch (state) {
    case SLOFF:
      //Physical Outputs state assignation
      digitalWrite(L1, LOW);  //Turn OFF L1
      digitalWrite(L2, LOW);  //Turn OFF L2

      //Internal Variables Computation
      //->Relative timing calculation
      trel = tact - tini;

      //Transition Questions
      if (trel >= TBLINK) { //If the relative time is greater or equal to TBLINK time
        state = SLON; //Next state is then SLON
        tini = millis();  //Reset the timer to a new value for next state timing
      }
      else if (digitalRead(BEMG) == HIGH) { //In other case if the BEMG is activated
        state = SAEMG;  //Next state is then SAEMG
      }
    break;

    case SLON:
      //Physical Outputs state assignation
      digitalWrite(L1, HIGH); //Turn ON L1
      digitalWrite(L2, LOW);  //Turn OFF L2

      //Internal Variables Computation
      //->Relative timing calculation
      trel = tact - tini;

      //Transition Questions
      if (trel >= TBLINK) {//If the relative time is greater or equal to TBLINK time
        state = SLOFF; //Next state is then SLOFF
        tini = millis();  //Reset the timer to a new value for next state timing
      }
      else if (digitalRead(BEMG) == HIGH) { // In other case if the BEMG is activated
        state = SAEMG;  // Next state is then SAEMG
      }
      break;

    case SAEMG:
      //Physical Outputs state assignation
      digitalWrite(L1, LOW); //Turn OFF L1
      digitalWrite(L2, HIGH);  //Turn ON L2

      //Transition Questions
      if (digitalRead(BEMG) == LOW) { //If BEMG is deactivated
        state = SLON;  //Next state is then SLON
        tini = millis(); //Reset the timer to a new value for next state timing
      }
      break;
  }
}
```

Challenge 1 – Traffic light with temperature sensor

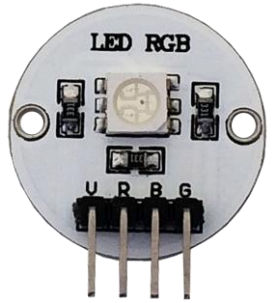
Design a traffic light with 3 LEDs (Red, White and Green) and a LM35 Linear temperature sensor, which works like the following procedure:



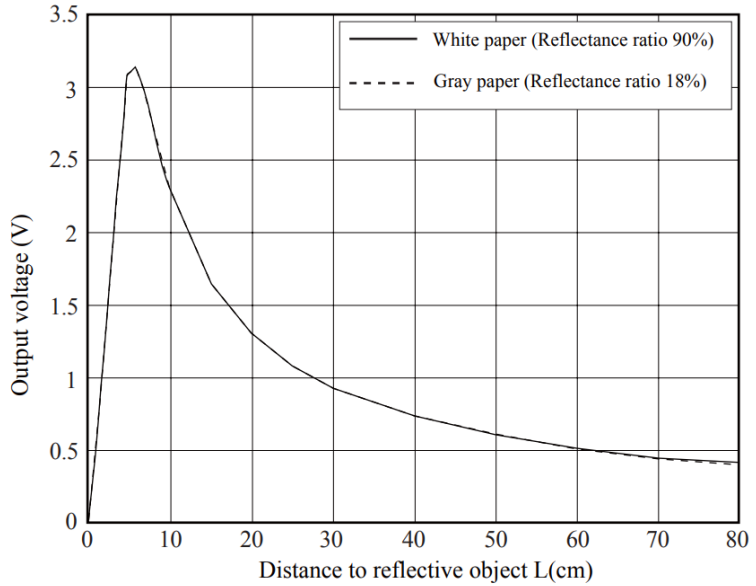
- It must start for safety in RED, wait 20 seconds and then change to GREEN.
- In GREEN, it waits 10 seconds and then changes to WHITE.
- In WHITE, it must wait 5 seconds and then change back to RED.
- If at any time the temperature sensor LM35 detects that the temperature is greater or equal to 30° C inside the traffic light electronic board, The traffic light advises the maintenance personal of a posible failure blinking its three LEDs ½ sec ON and ½ sec OFF. If the temperature goes back to normal conditions, the traffic light returns to the initial state (RED) and works again normally.

Challenge 2 – Proximity Indicator System

Design a proximity indicator system, which alerts certain user of a posible collision to a wall. Use a RGB LED and a Sharp Optic distance analog sensor following the next steps:



- The RGB LED will turn Green if the wall is located at a distance superior to 500 mm (0.5 meters).
- The RGB LED will turn Orange and blink 3 times (½ sec ON, ½ sec OFF) and then stay lit if the wall is located between 300 mm and 500 mm.
- The RGB LED will turn Red and constantly blink (200 msec ON, 200 msec OFF) if the wall is located at less than 300 mm.



Function to convert ADC value to mm value

```
unsigned int adc2mm (unsigned int senval) {
     if (value < 10) value = 10;
     return ((67870.0 / (value - 3.0)) - 40.0);
}
```

Thanks!