

# Electrónica Básica

## Clase 10

FUNCIÓN MILLIS

PLANTILLA MEF EN ARDUINO

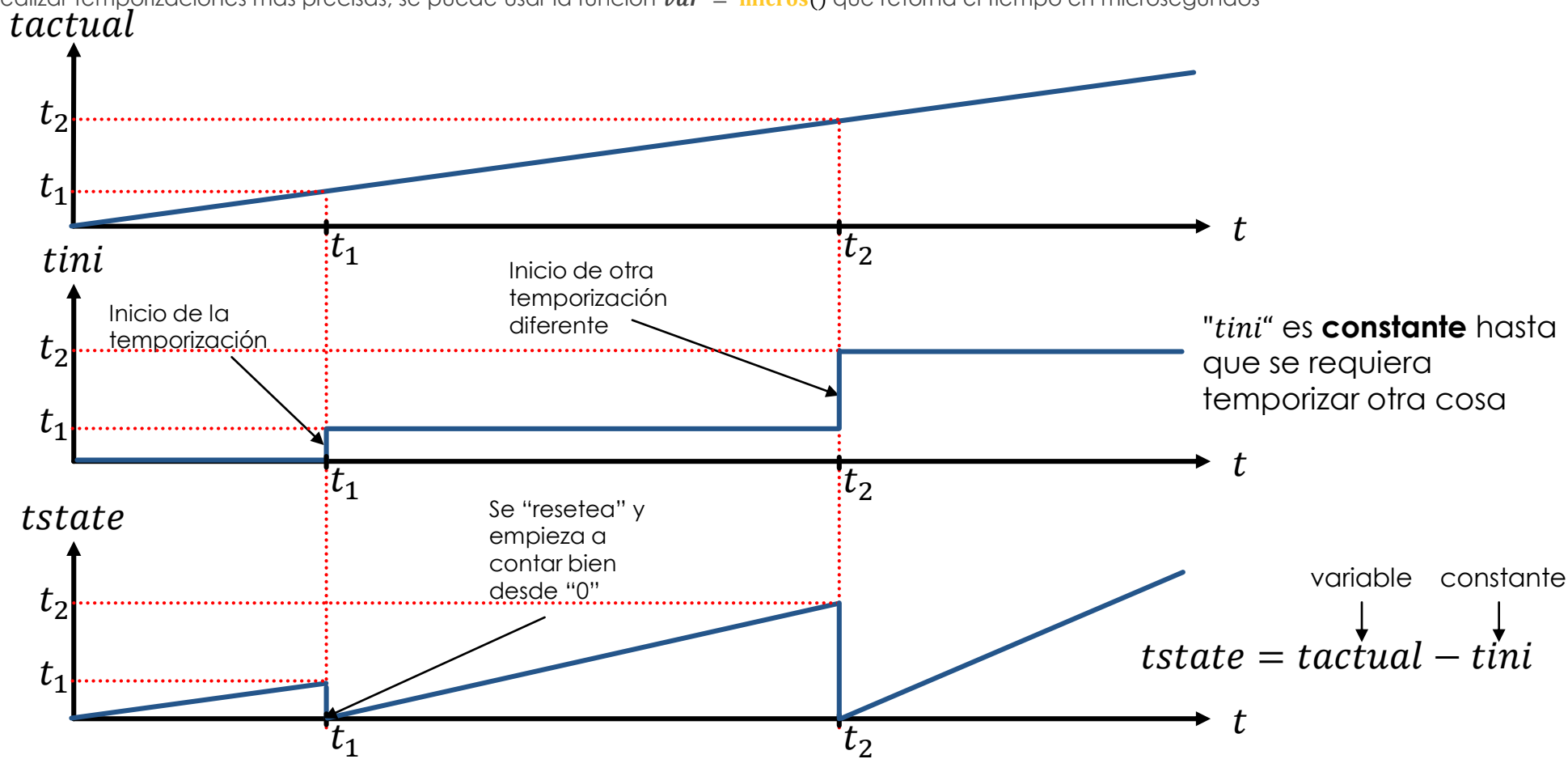
SIMULADOR DE ARDUINO [123d.circuits.io](https://123d.circuits.io)

SIMULADOR DE ARDUINO PROTEUS

# Funcion millis()

- La función **var** = **millis()** retorna el tiempo en milisegundos que lleva ejecutando el arduino desde que se prendió.
- Permite realizar conteos de tiempo relativos sin necesidad de hacer un retardo.
- Para utilizarla se necesitan tres variables:
  - tactual**: es el tiempo actual que hay, este se toma siempre y es exactamente igual a lo que devuelve **millis()**. Es decir  $tactual = millis()$ .
  - tini**: es el tiempo de inicio de conteo, se asemeja a cuando uno toma un cronometro y pulsa para empezar a cronometrar. Este solamente se asigna una vez en el INICIO de cuando deseo temporizar.
  - tstate**: es el tiempo del estado relativo, es la diferencia entre el **tactual** y el **tini**. Es decir:  $tstate = tactual - tini$ . Esta variable es la que me permite saber exactamente cuanto tiempo ha pasado desde que comencé a temporizar.

Nota: Si se desean realizar temporizaciones mas precisas, se puede usar la función **var** = **micros()** que retorna el tiempo en microsegundos



# Funcion millis() - Ejemplo 1

- Realice un programa en ARDUINO que haga titilar un LED (L1) ½ seg prendido y ½ seg apagado utilizando la instrucción millis().

```
//Definición de pines de I/O
#define L1 13 //L1 en el pin 13

//Definición de constantes
const unsigned long tpar = 500; //Defino la constante tiempo de parpadeo (tpar) como unsigned long y la inicializo en 500 milisegundos

//Definición de variables de temporización
unsigned long tini = 0; //Defino tini como unsigned long
unsigned long tactual = 0; //Defino tactual como unsigned long
unsigned long tstate = 0; //Defino tstate como unsigned long

void setup()
{
    //Definición de que pin es entrada y que es salida
    pinMode(L1, OUTPUT); //L1 como salida

    //Limpieza de salidas
    digitalWrite(L1, LOW); //Apago L1
    tini = millis(); //Inicializo por primera vez tini debido a que se usara de una vez en el void loop
}

void loop()
{
    tactual = millis(); //Tomo el tactual
    tstate = tactual - tini; //Calculo el tiempo relativo
    if (tstate < tpar) //Si el tstate es menor al tiempo de parpadeo
    {
        digitalWrite(L1, HIGH); //Prendo L1
    }
    else if (tstate < tpar*2) //Si el tstate es mayor al tiempo de parpadeo y menor al tiempo de parpadeo x 2 (mismo tiempo de encendido que de apagado)
    {
        digitalWrite(L1, LOW); //Apago L1
    }
    else //De resto si es mayor a 2 veces el tiempo de parpadeo
    {
        tini = millis(); //Reseteo nuevamente el tini para empezar con el ciclo de parpadeo nuevamente
    }
}
```

# Funcion millis() - Ejemplo 2

- Realice un programa en ARDUINO que cuente siempre el tiempo desde que se encendió el arduino en segundos y si en algún momento se presiona el botón (btn) en el pin 2, este comience nuevamente a temporizar desde 0. El valor del temporizador debe imprimirse en el monitor serial. Si el tiempo es superior a 5 segundos prende el LED de alarma (L1) ubicado en el pin 13.

```
//Definición de pines de I/O
#define btn 2 //btn en el pin 2
#define L1 13 //L1 en el pin 13

//Definición de constantes
const unsigned long talarm = 5000; //Defino la constante tiempo
de alarma como unsigned long y la inicializo en 5000
milisegundos

//Definición de variables de temporización
unsigned long tini = 0; //Defino tini como unsigned long
unsigned long tactual = 0; //Defino tactual como unsigned long
unsigned long tstate = 0; //Defino tstate como unsigned long

void setup()
{
    //Definición de que pin es entrada y que es salida
    pinMode(btn, INPUT); //btn como entrada
    pinMode(L1, OUTPUT); //L1 como salida

    //Limpieza de salidas
    digitalWrite(L1, LOW); //Apago L1

    //Inicializacion comunicacion serial
    Serial.begin(9600);
    tini = millis(); //Inicializo por primera vez tini debido a
    que se usara de una vez en el void loop
}

void loop()
{
```

```
tactual = millis(); //Tomo el tactual
tstate = tactual - tini; //Calculo el tiempo relativo
Serial.print("tstate: "); //Imprimo el texto tstate:
Serial.println(tstate/1000); //Imprimo la variable tstate en
segundos
if (digitalRead(btn) == HIGH) //Si btn esta en HIGH
{
    delay(200); //Retardo para antirebote del boton
    digitalWrite(L1, LOW); //Apago L1
    while (digitalRead(btn) == HIGH)
    {
        //No hago nada mientras se este presionando el boton
    }
    tini = millis(); //Reseteo nuevamente el tini para comenzar
a temporizar desde cero nuevamente
}
else if (tstate >= talarm) //Si el tstate es mayor o igual
que el tiempo de alarma
{
    digitalWrite(L1, HIGH); //Prendo L1
}
else //De resto si es mayor a 2 veces el tiempo de parpadeo
{
    digitalWrite(L1, LOW); //Apago L1
}
}
```

# Funcion millis() - Ejemplo 3

- Modifique el programa del cronómetro para que solo se le escriba el valor del tiempo al monitor serial cada segundo.

```
//Definición de pines de I/O
#define btn 34 //btn en el pin 2
#define L1 22 //L1 en el pin 13

//Definición de constantes
const unsigned long talarm = 5000; //Defino la constante tiempo de alarma como
unsigned long y la inicializo en 5000 milisegundos
const unsigned long tserial = 1000; //Defino la constante tiempo de impresión
del monitor serial como unsigned long y la inicializo en 1000 milisegundos

//Definición de variables de temporización para el cronómetro
unsigned long tini = 0; //Defino tini como unsigned long
unsigned long tactual = 0; //Defino tactual como unsigned long
unsigned long tstate = 0; //Defino tstate como unsigned long

//Definición de variables de temporización para el monitor serial
unsigned long tini2 = 0; //Defino tini como unsigned long
unsigned long tstate2 = 0; //Defino tstate como unsigned long

void setup()
{
    //Definición de que pin es entrada y que es salida
    pinMode(btn, INPUT); //btn como entrada
    pinMode(L1, OUTPUT); //L1 como salida

    //Limpieza de salidas
    digitalWrite(L1, LOW); //Apago L1

    //Inicializacion comunicacion serial
    Serial.begin(9600);
    tini = millis(); //Inicializo por primera vez tini debido a que se usara de
    una vez en el void loop
    tini2 = millis(); //Inicializo por primera vez tini2 debido a que se usara de
    una vez en el void loop para imprimir serialmente
}

void loop()
{
```

```
tactual = millis(); //Tomo el tactual
tstate = tactual - tini; //Calculo el tiempo relativo del cronómetro
tstate2 = tactual - tini2; //Calculo el tiempo relativo de impresion del
monitor serial

if (tstate2 >= tserial)
{
    Serial.print("tstate: "); //Imprimo el texto tstate:
    Serial.println(tstate/1000); //Imprimo la variable tstate en segundos
    tini2 = millis(); //Reseteo nuevamente el tini2 para comenzar a temporizar
    desde cero nuevamente
}
if (digitalRead(btn) == HIGH) //Si btn esta en HIGH
{
    delay(200); //Retardo para antirebote del boton
    digitalWrite(L1, LOW); //Apago L1
    while (digitalRead(btn) == HIGH)
    {
        //No hago nada mientras se este presionando el boton
    }
    tini = millis(); //Reseteo nuevamente el tini para comenzar a temporizar
    desde cero nuevamente
}
else if (tstate >= talarm) //Si el tstate es mayor o igual que el tiempo de
alarma
{
    digitalWrite(L1, HIGH); //Prendo L1
}
else //De resto si es mayor a 2 veces el tiempo de parpadeo
{
    digitalWrite(L1, LOW); //Apago L1
}
}
```

# Plantilla MEF - Arduino

**Declaración de LIBRERÍAS** (Ej: #include <SFEMP3Shield.h>)

**Definición de ESTADOS de la MEF** (Ej: #define EINI 0)

**Definición de PINES** (Ej: #define ledPin 13)

**Declaración de CONSTANTES** (Ej: const int numSensores = 6;)

**Declaración de VARIABLES** (Ej: float temperatura = 0;)  
unsigned int nxstate = EINI; //Declaración de la variable para almacenar el estado actual

**Declaración de VARIABLES DE TEMPORIZACIÓN** (Ej: unsigned long tini = 0;)

**Declaración de SUBROUTINAS o FUNCIONES** (Ej: void titilar()) (Ej: unsigned int sumar (unsigned int A, unsigned int B))

**void setup()** (Configuración de puertos y limpieza de puertos)  
void setup()  
{  
    //Configuración de que es entrada y que es salida  
    //Limpieza de salidas  
    tini = millis(); //Inicializacion de tini (si se requiere y si es utilizado a partir del estado inicial)  
}

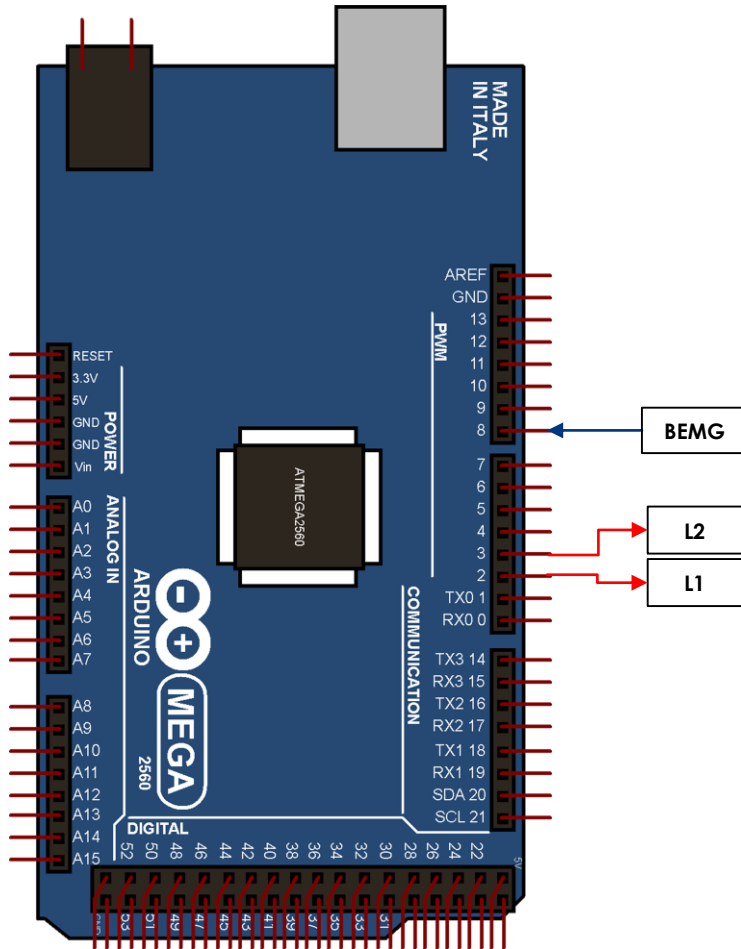
**void loop()** (Programa principal)  
void loop()  
{  
    tactual = millis(); //Cálculo del tiempo actual (si se requiere)  
    //MEF  
    switch (nxstate)  
    {  
        case EINI: //Estado Inicial  
            //Configuración de salidas o variables internas según el estado  
            digitalWrite(L1, LOW); //Por ejemplo apago el LED L1 en el estado EINI  
  
            //Calculo del tiempo del estado (Si el estado requiere temporización)  
            tstate = tactual - tini;  
  
            //Preguntas de transición  
            if (tstate >= valor) //Si el tiempo es mayor que una constante llamada valor  
            {  
                nxstate = ELEDON; //El siguiente estado es ELEDON  
                tini = millis(); //Reinicio del temporizador (Solo necesario si se requiere temporizar en el siguiente estado)  
            }  
            else if (digitalRead(btnEMG) == HIGH) //Si se pulso el botón de emergencia  
            {  
                nxstate = EALERTA; //El siguiente estado es EALERTA  
            }  
    }  
}

else //Si no se cumple ninguna de las anteriores condiciones de transición  
{  
    nxstate = EINI; //Me quedo en EINI  
}  
break;  
  
case ELEDON:  
    //Configuración de salidas ...  
break;

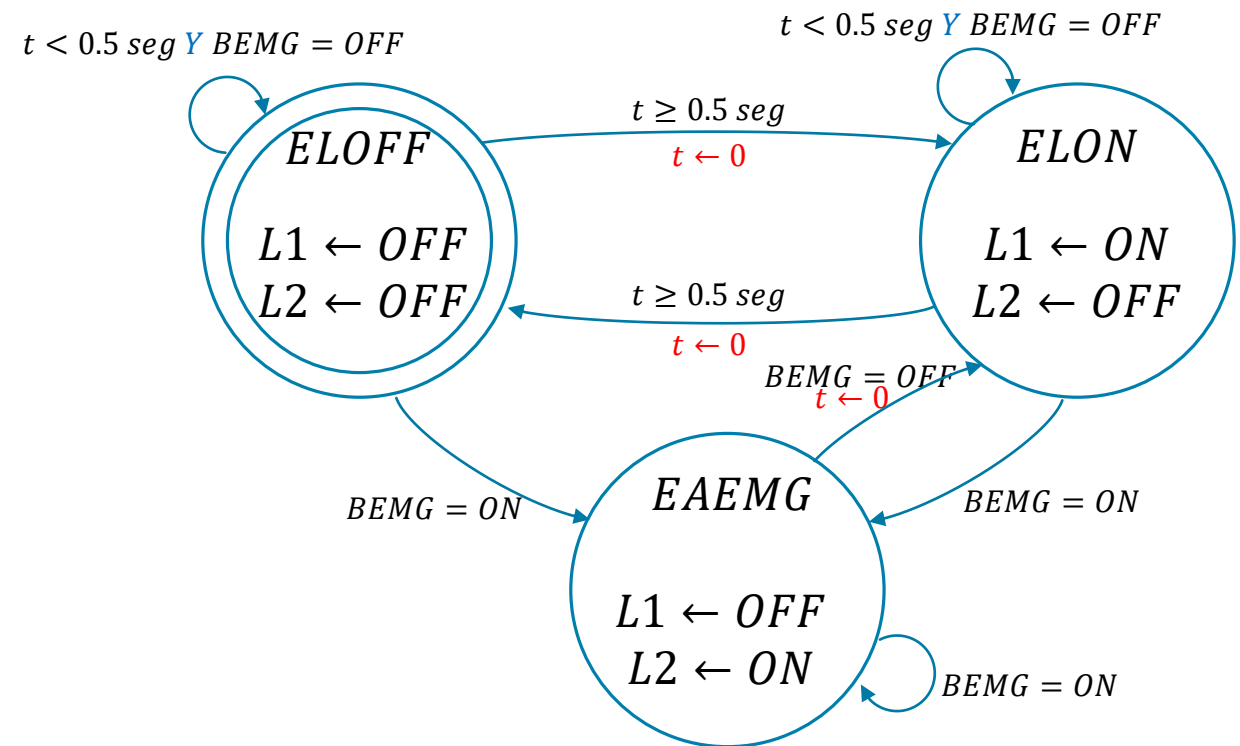
}

# Ejemplo 1 - MEF

- Elabore un programa que haga “titilar” un LED (**L1**) en el **pin 2**, 1/2 *seg* prendido y 1/2 *seg* apagado mientras que no se accione el suiche de emergencia (**BEMG**) en el **pin 8**. El LED (**L1**) queda apagado y se prende un LED Rojo (**L2**) en el **pin 3** hasta que se libere el suiche de emergencia. Una vez se libere el suiche de emergencia, el proceso a su funcionamiento normal.



ENTRADAS			SALIDAS		
Nombre	Descripción	Tipo	Nombre	Descripción	Tipo
<i>BEMG</i>	Suiche de emergencia	Booleana (Digital)	<i>L1</i>	LED	Booleana (Digital)
			<i>L2</i>	LED Rojo	Booleana (Digital)
			<i>t</i>	Temporizador	Variable Interna



# Ejemplo 1 - MEF

```
//Definicion de estados de la MEF
#define ELOFF 0
#define ELON 1
#define EAEMG 2

//Definicion de posicion de pines de I/O
#define BEMG 8 //BEMG en el pin 8
#define L1 2 //L1 en el pin 2
#define L2 3 //L2 en el pin 3

//Definición de constantes
const unsigned long tpar = 500; //Defino el tiempo de parpadeo como constante unsigned long y la inicio en 500 milisecs

//Definicion de variables
unsigned int nxstate = ELOFF; //Declaro la variable nxstate (next state) como entero y la inicializo en ELOFF (0)

//Definicion de variables de temporizacion
unsigned long tini = 0;
unsigned long tactual = 0;
unsigned long tstate = 0;

void setup() {
    //Definicion de que es entrada y que es salida
    pinMode(BEMG, INPUT); //BEMG como entrada
    pinMode(L1, OUTPUT); //L1 como salida
    pinMode(L2, OUTPUT); //L2 como salida

    //Limpieza de salidas
    digitalWrite(L1, LOW); //Apago L1
    digitalWrite(L2, LOW); //Apago L2
    tini = millis(); //Inicializacion de tini
}

void loop() {
    tactual = millis(); //Cálculo del tiempo actual
    //MEF
    switch (nxstate) {
        case ELOFF:
            //Configuracion de salidas o var internas
            digitalWrite(L1, LOW); //Apago L1
            digitalWrite(L2, LOW); //Apago L2

            //Calculo del tiempo del estado
            tstate = tactual - tini;

            //Preguntas de transicion
            if (tstate >= tpar) { //Si el tiempo del estado es mayor que el tiempo de parpadeo
```

```
                nxstate = ELON; //El proximo estado es ELON
                tini = millis(); //Reinicio del temporizador
            }
            else if (digitalRead(BEMG) == HIGH) { //De resto si se acciono el BEMG
                nxstate = EAEMG; //El proximo estado es EAEMG
            }
            else { //De resto si no se cumple nada de lo anterior
                nxstate = ELOFF; //Me quedo en ELOFF
            }
            break;

        case ELON:
            //Configuracion de salidas o var internas
            digitalWrite(L1, HIGH); //Prendo L1
            digitalWrite(L2, LOW); //Apago L2

            //Calculo del tiempo del estado
            tstate = tactual - tini;

            //Preguntas de transicion
            if (tstate >= tpar) { //Si el tiempo del estado es mayor que el tiempo de parpadeo
                nxstate = ELOFF; //El proximo estado es ELOFF
                tini = millis(); //Reinicio del temporizador
            }
            else if (digitalRead(BEMG) == HIGH) { //De resto si se acciono el BEMG
                nxstate = EAEMG; //El proximo estado es EAEMG
            }
            else { //De resto si no se cumple nada de lo anterior
                nxstate = ELON; //Me quedo en ELON
            }
            break;

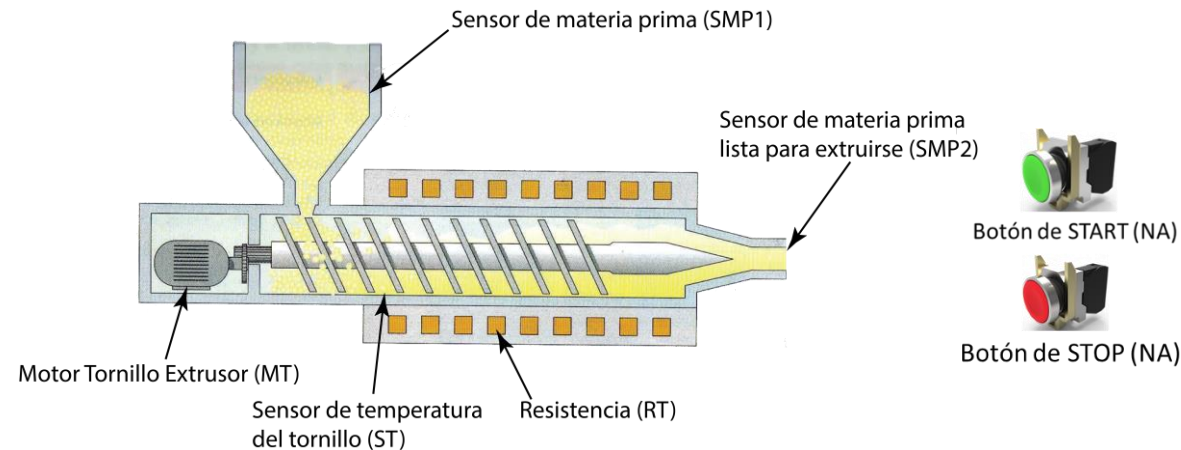
        case EAEMG:
            //Configuracion de salidas o var internas
            digitalWrite(L1, LOW); //Apago L1
            digitalWrite(L2, HIGH); //Prendo L2

            //Preguntas de transicion
            if (digitalRead(BEMG) == LOW) { //Si se libero el BEMG
                nxstate = ELON; //El proximo estado es ELON
                tini = millis(); //Reinicio del temporizador
            }
            else { //De resto si no se cumple nada de lo anterior
                nxstate = EAEMG; //Me quedo en EAEMG
            }
            break;
    }
}
```

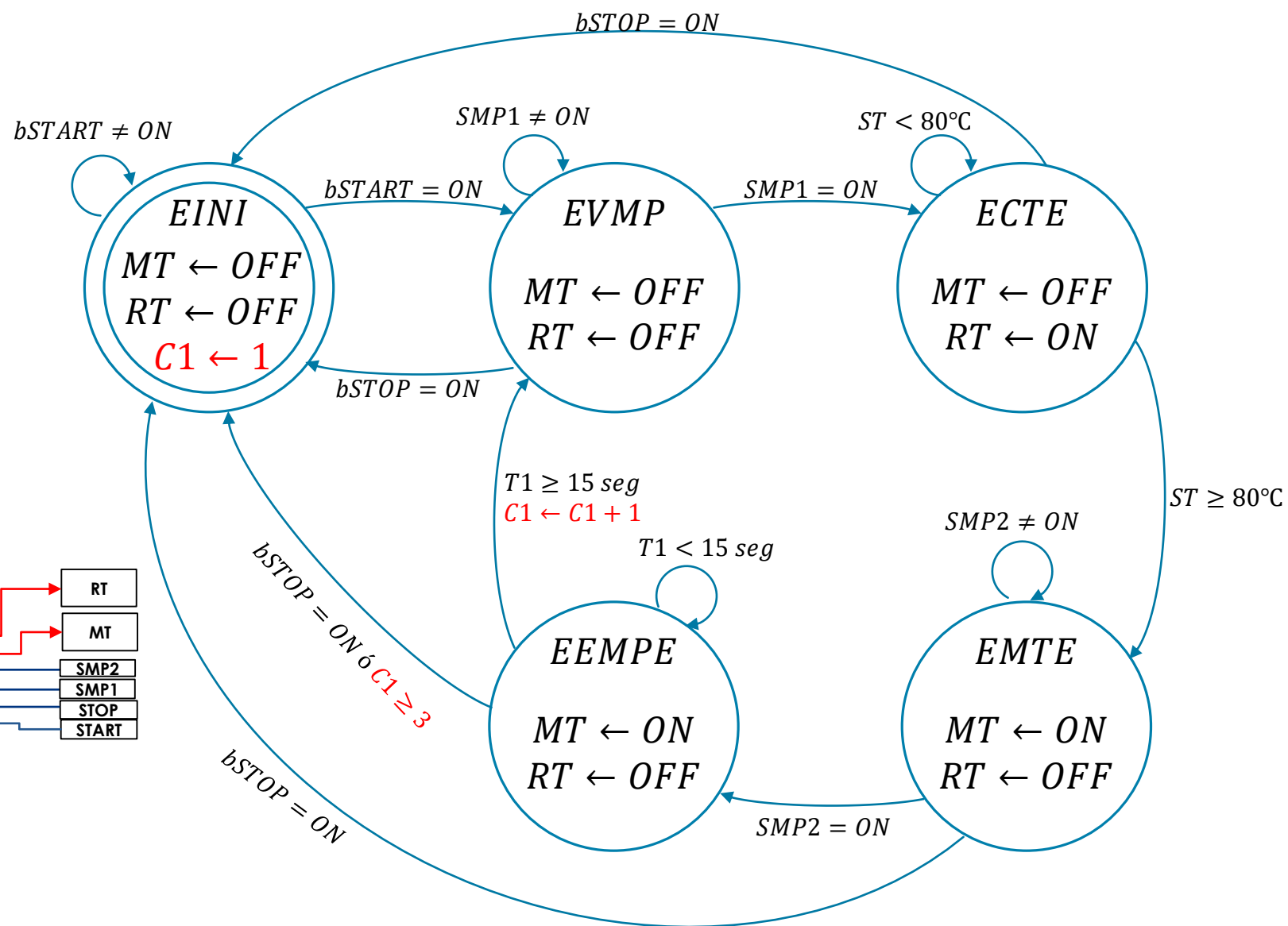
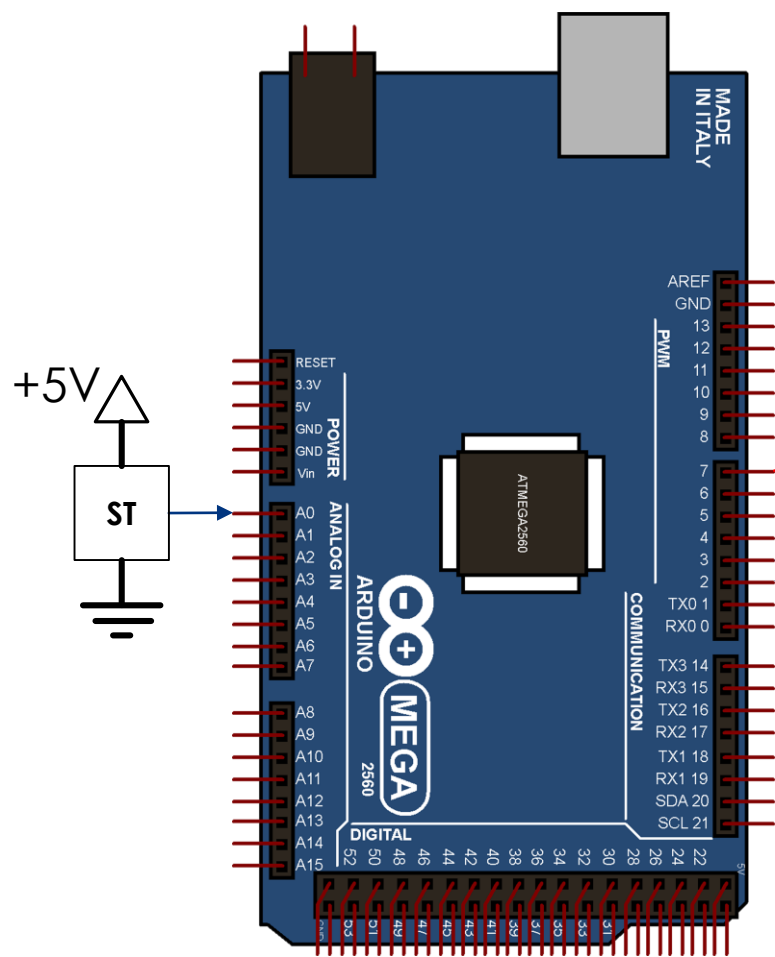
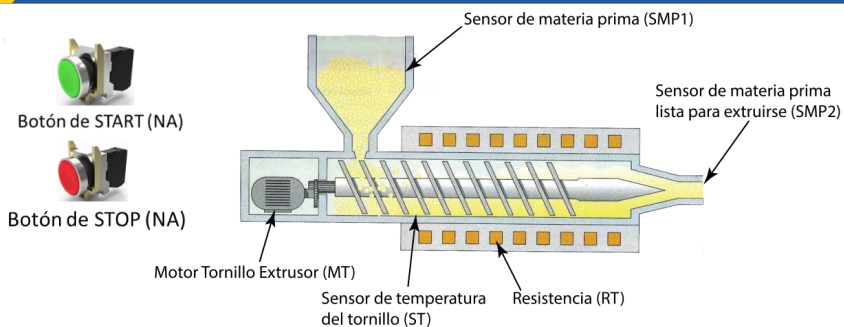


# MEF avanzado - Ejemplo

- Realice un diagrama de flujo que controle la inyectora de plásticos, teniendo en cuenta el siguiente funcionamiento:
  - La inyectora esta totalmente apagada (RT y MT apagados) al comienzo.
  - Para iniciar la inyectora de plásticos se debe presionar el botón de Start.
  - Primero, la inyectora debe verificar que la materia prima este en el nivel adecuado (SMP1 = ON).
  - Luego se debe calentar el tornillo extrusor mediante la resistencia RT hasta que haya alcanzado una temperatura de  $80^{\circ}\text{C}$  ( $ST \geq 80$ ).
  - Una vez el tornillo este en la temperatura adecuada, se debe apagar RT y se debe prender MT para comenzar a llevar la materia prima derretida hasta la punta del inyector.
  - Cuando la materia prima llegue a la punta del inyector (SMP2 = ON), la inyectora debe esperar 15 segundos y luego debe apagar MT para comenzar de nuevo el ciclo (desde verificar materia prima).
  - Si se presiona STOP (bSTOP) en cualquier momento vuelve al estado inicial.
  - Agregue un contador (C1 – Variable Interna) que si el proceso se ha repetido mas de 3 veces, este vuelva al estado inicial (en donde todo vuelve a ceros).



# EJEMPLO – MEF AVANZADO



```
//Definicion de estados de la MEF
#define EINI 0
#define EVMP 1
#define ECTE 2
#define EMTE 3
#define EWMPE 4

//Definicion de posicion de pines de I/O
#define bSTART 2
#define bSTOP 3
#define SMP1 4
#define SMP2 5
#define ST 0 //Sensor de temperatura en la entrada analoga A0
#define MT 6
#define RT 7

//Definicion de variables
unsigned int nxstate = EINI; //Declaro la variable nxstate
(next state - estado presente) como entero y la inicializo en
EINI (0)
unsigned int C1 = 1; //Declaro la variable C1 (contador de
ciclos) como entero y la inicializo en 1
int vST; //Declaro la variable vST (Valor del Sensor de
temperatura) como entero

//Definicion de temporizadores
unsigned long tini = 0;
unsigned long tactual = 0;
unsigned long tstate = 0;

void setup() {
    //Definicion de pines como entradas o salidas
    pinMode(bSTART, INPUT);
    pinMode(bSTOP, INPUT);
    pinMode(SMP1, INPUT);
    pinMode(SMP2, INPUT);
    pinMode(MT, OUTPUT);
    pinMode(RT, OUTPUT);

    //Limpieza de salidas al comienzo
    digitalWrite(MT, LOW);
    digitalWrite(RT, LOW);
}

void loop() {
    tactual = millis(); //Almaceno en tactual el valor del tiempo
    actual

    switch(nxstate) {
        case EINI: //Estado Inicial EINI
            //Especifico el valor de las salidas para este estado
```

```
        digitalWrite(MT, LOW);
        digitalWrite(RT, LOW);
        C1 = 1;
        if(digitalRead(bSTART) == HIGH) { //Pregunto si bSTART
            esta en ON
            nxstate = EVMP; //En caso de bSTART estar en ON hago el
            cambio de estado a EVMP
        }
        else {
            nxstate = EINI; //De resto me quedo en EINI
        }
        break;

        case EVMP: //Estado Verificar Materia Prima EVMP
            //Especifico el valor de las salidas para este estado
            digitalWrite(MT, LOW);
            digitalWrite(RT, LOW);
            if(digitalRead(SMP1) == HIGH) { //Pregunto si SMP1 esta
                en ON
                nxstate = ECTE; //En caso de SMP1 estar en ON hago el
                cambio de estado a ECTE
            }
            else if(digitalRead(bSTOP) == HIGH) { //Pregunto si se
                pulso STOP
                nxstate = EINI; //En caso de que se pulse STOP, hago la
                transicion de vuelta a EINI
            }
            else {
                nxstate = EVMP; //De resto me quedo en EVMP
            }
        }
        break;

        case ECTE: //Estado Calentar tornillo extrusor ECTE
            //Especifico el valor de las salidas para este estado
            digitalWrite(MT, LOW);
            digitalWrite(RT, HIGH); //Prendo la resistencia
            vST = (float)analogRead(ST)*100/1023; //Leo en cuanto se
            encuentra el ST y lo "escalizo" por medio de una regla de tres
            para que quede de 0 a 100°
            if(vST >= 80) { //Pregunto si la temperatura ya es
                superior o igual a 80°C
                nxstate = EMTE; //En caso de que la temperatura si
                supere o iguale los 80°C hago el cambio de estado a EMTE
            }
            else if(digitalRead(bSTOP) == HIGH) { //Pregunto si se
                pulso STOP
                nxstate = EINI; //En caso de que se pulse STOP, hago la
                transicion de vuelta a EINI
            }
            else {
                nxstate = ECTE; //De resto me quedo en ECTE
```

```
        }
        break;

        case EMTE: //Estado Mover Tornillo Extrusor EMTE
            //Especifico el valor de las salidas para este estado
            digitalWrite(MT, HIGH); //Prendo el motor
            digitalWrite(RT, LOW);
            if(digitalRead(SMP2) == HIGH) { //Pregunto si SMP2 esta
                en ON
                nxstate = EWMPE; //En caso de SMP2 estar en ON hago el
                cambio de estado a EEMPE
                tini = millis(); //Tomo el primer tiempo puesto que voy
                a empezar a temporizar en el proximo estado
            }
            else if(digitalRead(bSTOP) == HIGH) { //Pregunto si se
                pulso STOP
                nxstate = EINI; //En caso de que se pulse STOP, hago la
                transicion de vuelta a EINI
            }
            else {
                nxstate = EMTE; //De resto me quedo en EMTE
            }
        }
        break;

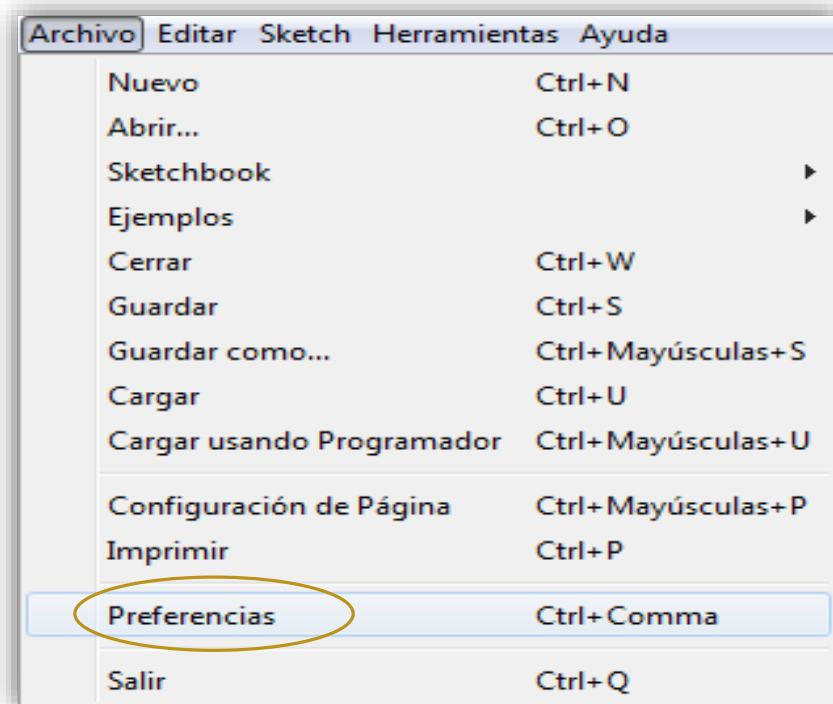
        case EWMPE: //Estado Esperar (Wait) Materia Prima Extruida
            EWMPE
            //Especifico el valor de las salidas para este estado
            digitalWrite(MT, HIGH); //Dejo el motor encendido
            digitalWrite(RT, LOW);
            tstate = tactual - tini; //Calculo en mientras estoy en
            este estado cuanto tiempo llevo en este estado
            if(tstate >= 15000) { //Pregunto si ya han pasado mas de
                15000 milisegundos es decir 15 segundos
                nxstate = EVMP; //En caso de haber pasado mas de 15
                segundos hago el cambio de estado a EVMP
                C1 = C1 + 1; //Incremento el contador
            }
            else if(digitalRead(bSTOP) == HIGH || C1 >= 3) {
                //Pregunto si se pulso STOP ó si el contador ya es mayor o igual
                que 3
                nxstate = EINI; //En caso de que se pulse STOP o el
                contador sea mayor o igual que 3, hago la transicion de vuelta a
                EINI
            }
            else {
                nxstate = EWMPE; //De resto me quedo en EWMPE
            }
        }
        break;
    }
}
```

MUCHAS GRACIAS

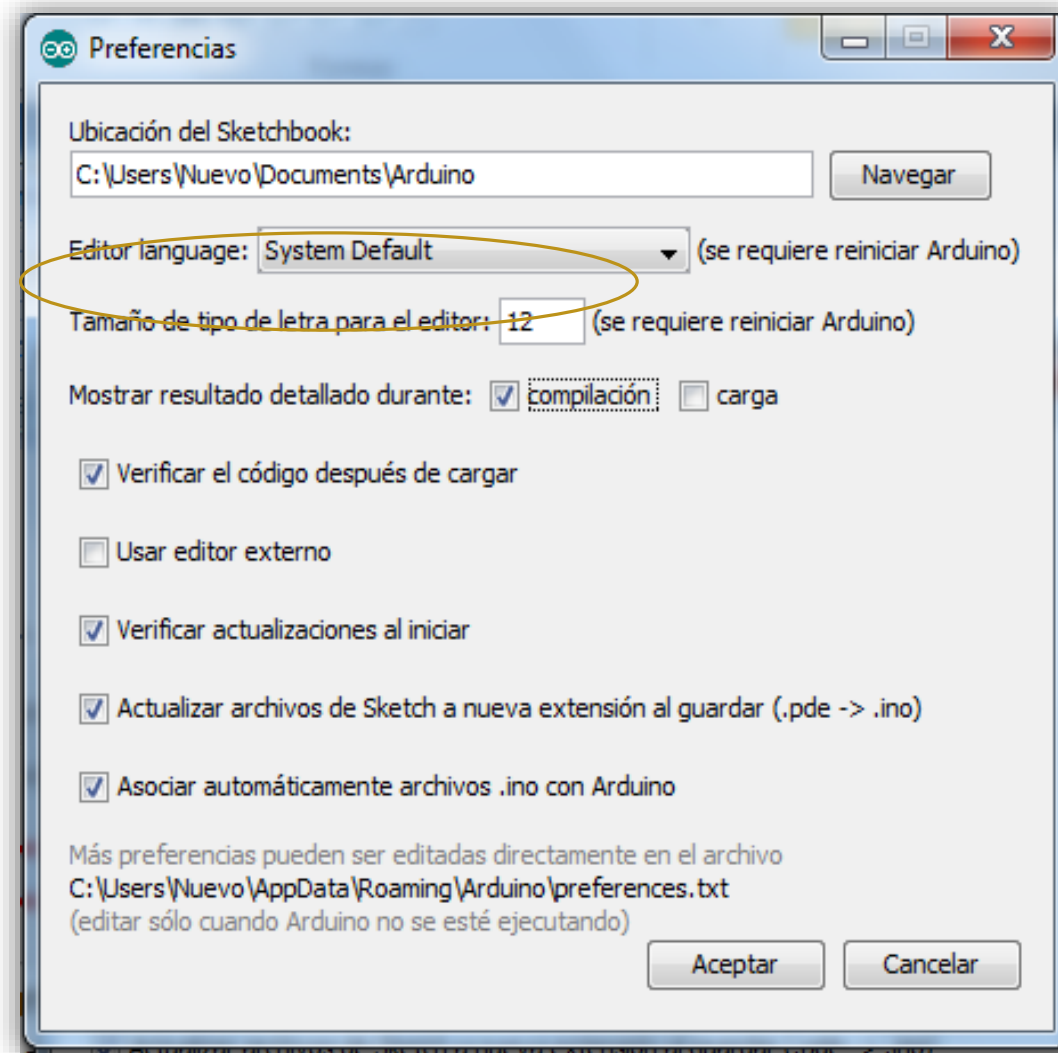
Para simular en proteus, lo primero que hay que hacer es ubicar el archivo **.HEX** que genera el sketch de Arduino y esto se hace de la siguiente manera:

Seleccionar la opción archivo en la barra de herramientas situada en la parte superior del entorno Arduino.

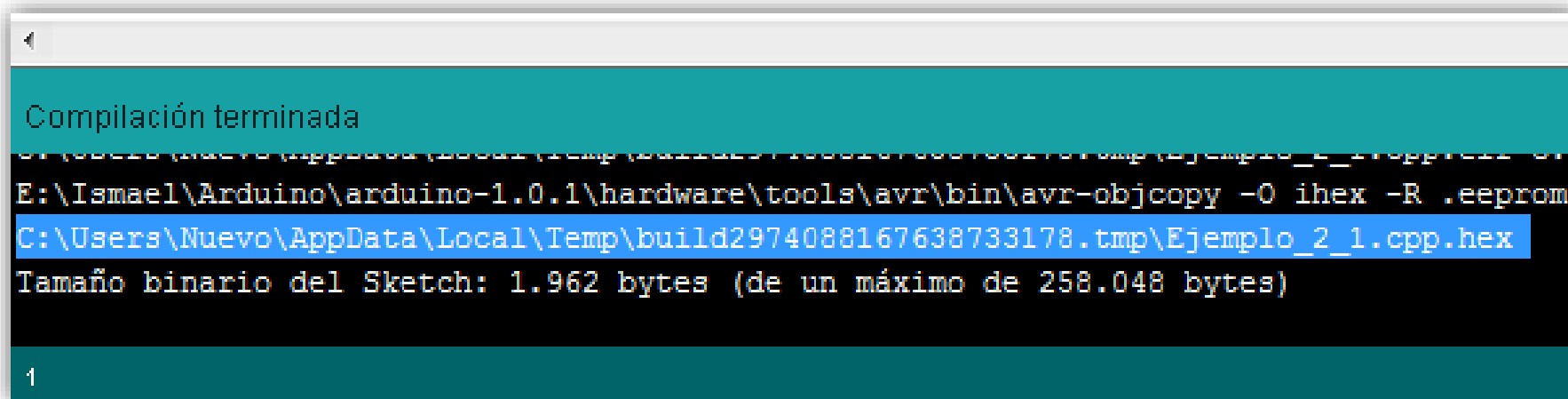
Seleccionar la opción preferencias.



- ▶ En la ventana de preferencias activar la opción mostrar resultados detallados durante la compilación.



- Después de haber hecho estos pasos se procede a compilar el sketch (programa) y en la barra situada en la parte inferior del entorno Arduino se muestra la ubicación del archivo **.HEX**.



```
Compilación terminada
E:\Ismael\Arduino\arduino-1.0.1\hardware\tools\avr\bin\avr-objcopy -O ihex -R .eeprom
C:\Users\Nuevo\AppData\Local\Temp\build2974088167638733178.tmp\Ejemplo_2_1.cpp.hex
Tamaño binario del Sketch: 1.962 bytes (de un máximo de 258.048 bytes)
```

# SIMULACIÓN EN PROTEUS

Los siguientes son los parámetros con los que debe estar configurado el Arduino en proteus para realizar la simulación correctamente, esta ventana se despliega al dar doble clic sobre el Microcontrolador Arduino.

