MyAnalogWrite

Diseño de Bloques

Por Juan-Carlos Giraldo

Diseño de Sistemas de Computador Embebido

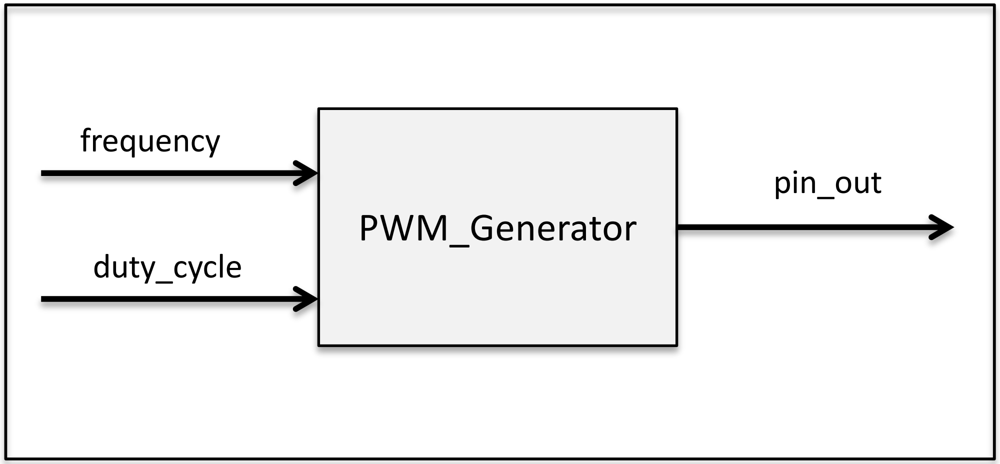
Departamento de Electrónica

Pontificia Universidad Javeriana

Última versión a Octubre de 2019

Fase de Concepción

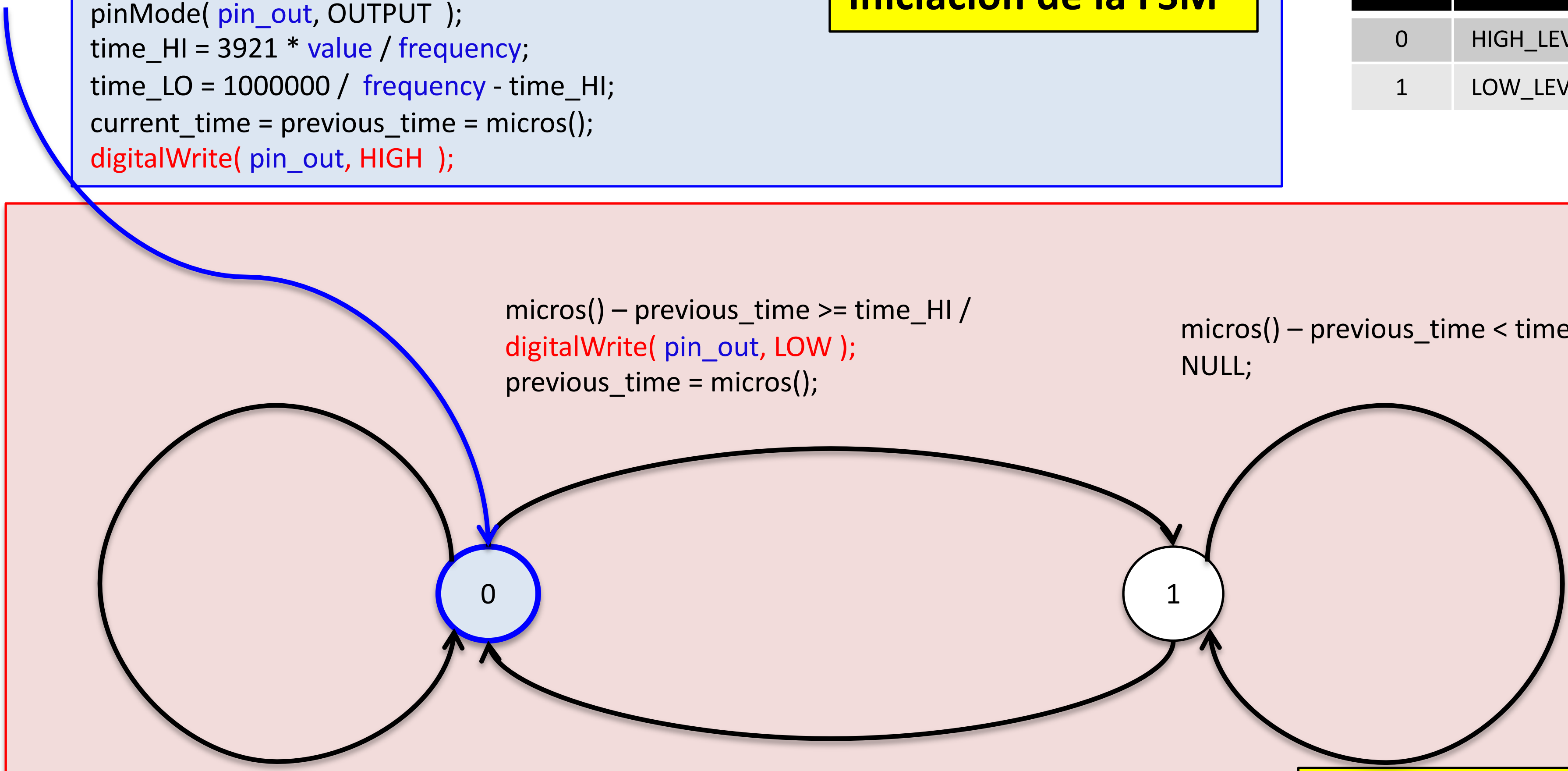
PWM_Generator



Iniciación de la FSM

NULL /
pinMode(pin_out, OUTPUT);
time_HI = 3921 * value / frequency;
time_LO = 1000000 / frequency - time_HI;
current_time = previous_time = micros();
digitalWrite(pin_out, HIGH);

No.	ESTADO
0	HIGH_LEVEL
1	LOW_LEVEL



micros() – previous_time >= time_HI /
digitalWrite(pin_out, LOW);
previous_time = micros();

micros() – previous_time < time_LO /
NULL;

micros() – previous_time < time_HI /
NULL;

micros() – previous_time >= time_LO /
digitalWrite(pin_out, HIGH);
previous_time = micros();
time_HI = 3921 * value / frequency;
time_LO = 1000000 / frequency - time_HI;

La propia FSM

(Transiciones entre estados)

Variables de la FSM

CONTEXTO:
state + conjunto de vbles prIvadas:
previous_time, current_time,
time_HI, time_LO;
pin_out;

Embedded Computer Systems @ “*Electrónica Javeriana*” – Bogotá – Colombia con Juan C. Giraldo		
FSM: *PWM Generator*	**Dedicación**: *15 min*	**Elaborado por**: *Ing. Juan C. Giraldo Ph.D.*

```
typedef enum { HIGH_LEVEL, LOW_LEVEL } PWM_STATE_T;

void MyAnalogWrite() /* PWM with frequency @ 490 Hz */
{------------------------------------------------------
uint8_t  value = 127, previous_value;
int      pin_out = 13;

PWM_STATE_T state = HIGH_LEVEL;
uint32_t previous_time = micros();
uint32_t time_HI = ( (uint32_t)value << 3 ),
         time_LO = 2040 – time_HI;

pinMode( pin_out, OUTPUT  );
digitalWrite( pin_out, HIGH  );
 ------------------------------------------------------
while( FOREVER ) {

  switch( state ) {

    case HIGH_LEVEL:
      if( micros() – previous_time >= time_HI ) {
        digitalWrite( pin_out, LOW );
        previous_time = micros();
        state = LOW_LEVEL;
      }
    break;

    case LOW_LEVEL:
      if( micros() – previous_time >= time_LO ) {
        digitalWrite( pin_out, HIGH );
        previous_time = micros();
        if( value != previous_value ) {
          time_HI = ( (uint32_t)value << 3 );
          time_LO = 2040 - time_HI;
          previous_value = value;
        }
        state = HIGH_LEVEL;
      }
    break;

  } /* switch */
} /* while */
} /* MyAnalogWrite */
```

analogWrite en 3 partes:

En Arduino:

Variables de la FSM

+

Iniciación de la FSM

(Condiciones iniciales)

Contexto

+

setup()

La propia FSM

(Transiciones entre estados)

loop()

¡La función previa se “ROMPIÓ” en 3 partes!

```
typedef struct {
  uint8_t     previous_value;
  int         pin_out;
  uint32_t    previous_time;
  uint32_t    time_HI, time_LO;
  PWM_STATE_T state;
} ANALOG_WRITE_T;
```

ctx

```
void MyAnalogWriteSetup( ANALOG_WRITE_T *fsm, uint8_t value, int pin ) {
fsm->previous_value = value;
fsm->pin_out = pin;
fsm->previous_time = micros();
fsm->time_HI = ( (uint32_t)value << 3 );
fsm->time_LO = 2040 - time_HI;
fsm->state = HIGH_LEVEL;
pinMode( pin, OUTPUT );
digitalWrite( pin, HIGH  );
} /* MyAnalogWriteSetup */
```

setup()

```
void MyAnalogWrite( ANALOG_WRITE_T *fsm, uint8_t value ) {
switch( fsm->state ) {
  case HIGH_LEVEL:
    if( micros() – fsm->previous_time >= fsm->time_HI ) {
      digitalWrite( fsm->pin_out, LOW );
      fsm->previous_time = micros();
      fsm->state = LOW_LEVEL;
    }
  break;

  case LOW_LEVEL:
    if( micros() – fsm->previous_time >= fsm->time_LO ) {
      digitalWrite( fsm->pin_out, HIGH );
      fsm->previous_time = micros();
      if( value != fsm->previous_value ) {
        fsm->time_HI = ( (uint32_t)value << 3 );
        fsm->time_LO = 2040 - time_HI;
        fsm->previous_value = value;
      }
      fsm->state = HIGH_LEVEL;
    }
  break;
} /* switch */
} /* MyAnalogWrite */
```

loop()

Fue necesario DESCOMPONER la función previa en tres partes diferentes que van a tres posiciones diferentes en el programa.

```
typedef struct {
  uint8_t     previous_value;
  int         pin_out;
  uint32_t    previous_time;
  uint32_t    time_HI, time_LO;
  PWM_STATE_T state;
} ANALOG_WRITE_T;

void MyAnalogWriteSetup( ANALOG_WRITE_T *fsm, uint8_t value, int pin ) {
fsm->previous_value = value;
fsm->pin_out        = pin;
fsm->previous_time  = micros();
fsm->time_HI = ( (uint32_t)value << 3 );
fsm->time_LO = 2040 - time_HI;
fsm->state   = HIGH_LEVEL;
pinMode( pin, OUTPUT );
digitalWrite( pin, HIGH  );
} /* MyAnalogWriteSetup */

void MyAnalogWrite( ANALOG_WRITE_T *fsm, uint8_t value ) {
switch( fsm->state ) {
  case HIGH_LEVEL:
    if( micros() – fsm->previous_time >= fsm->time_HI ) {
      digitalWrite( fsm->pin_out, LOW );
      fsm->previous_time = micros();
      fsm->state = LOW_LEVEL;
    }
  break;

  case LOW_LEVEL:
    if( micros() – fsm->previous_time >= fsm->time_LO ) {
      digitalWrite( fsm->pin_out, HIGH );
      fsm->previous_time = micros();
      if( value != fsm->previous_value ) {
        fsm->time_HI = ( (uint32_t)value << 3 );
        fsm->time_LO = 2040 - time_HI;
        fsm->previous_value = value;
      }
      fsm->state = HIGH_LEVEL;
    }
  break;
} /* switch */
} /* MyAnalogWrite */
```

```
typedef enum { HIGH_LEVEL, LOW_LEVEL } PWM_STATE_T;
typedef struct { ... } ANALOG_WRITE_T;

void MyAnalogWriteSetup( ANALOG_WRITE_T*,
                                 uint8_t, int );
void MyAnalogWrite( ANALOG_WRITE_T *, uint8_t );

ANALOG_WRITE_T led[8];

void setup() {
MyAnalogWriteSetup( &led[0],   0,  8 );
MyAnalogWriteSetup( &led[1],  36,  9 );
MyAnalogWriteSetup( &led[2],  73, 10 );
MyAnalogWriteSetup( &led[3], 109, 11 );
MyAnalogWriteSetup( &led[4], 146, 12 );
MyAnalogWriteSetup( &led[5], 182, 13 );
MyAnalogWriteSetup( &led[6], 219,  6 );
MyAnalogWriteSetup( &led[7], 255,  7 );
} /* setup */

void loop() {
MyAnalogWrite( &led[0],   0 );
MyAnalogWrite( &led[1],  36 );
MyAnalogWrite( &led[2],  73 );
MyAnalogWrite( &led[3], 109 );
MyAnalogWrite( &led[4], 146 );
MyAnalogWrite( &led[5], 182 );
MyAnalogWrite( &led[6], 219 );
MyAnalogWrite( &led[7], 255 );
} /* loop */
```

Gracias por haber seguido esta presentación hasta este punto

Aquí vale la pena hacer una pausa para plantear sus dudas e inquietudes
a través de mi correo electrónico jcgiraldo@javeriana.edu.co

Me gustaría saber si tiene sugerencias para mejorar esta guía y apreciaría que me permitiera saber, si esta guía ha sido de utilidad para aclarar conceptos y en qué sentido le ayudó a Usted, para hacer mayor énfasis en próximas entregas.