U.B.A. FACULTAD DE INGENIERÍA

Departamento de Electrónica

ESTRUCTURA DEL COMPUTADOR 66-70

TRABAJO PRÁCTICO

AUTÓMATA DE CARGA DE TANQUE DE AGUA

Ribotta, Mariano

Padrón 86052

Curso 2011 - 1° Cuatrimestre

Turno: Miércoles

Desarrollo

Análisis del sistema

El primer paso, es determinar, de acuerdo a los estados de los tres sensores principales (dos en el tanque, uno en el reservorio), si debería bombear, o no.

Entradas

- **N_MIN**: Sensor que indica con un 1 que el agua del tanque está por debajo del nivel mínimo.
- **N_MAX**: Sensor que indica con un 1 que el agua del tanque está por encima del nivel máximo.
- **HAY_AGUA**: Sensor que indica con un 1 que en el reservorio hay suficiente agua para una carga completa del tanque.

Salidas

- **BOMBEAR**: Función que, con un 1, indica que la bomba debería encenderse.
- **LUZ_ROJA**: Función que, con un 1, enciende una luz roja que indica que debería prenderse la bomba, pero no hay suficiente agua en el reservorio.

Dado que necesito considerar el estado anterior de la bomba (n), para determinar el siguiente (n+1), realizo la tabla completa, tomando el estado anterior como entrada. (Tabla 1, ver apéndice)

Y a partir de ella, realizo la reducida, incluyendo como salidas, los valores que deberían tener las entradas J y K del Flip Flop que usaré. (Tabla 2)

Utilizando Karnaugh (ver apéndice), obtengo J y K, en función de las variables de entrada, N_MIN, N_MAX y HAY_AGUA:

- **J** = N_MIN . HAY_AGUA.
- **K** = N_MAX + ~HAY_AGUA.

Y la función LUZ_ROJA:

- **LUZ_ROJA** = N_MIN . ~HAY_AGUA.

Una vez obtenida la función que determina si debo encender la bomba, debo tener en cuenta el

temporizador (explicado más adelante).

Entradas

- **BOMBEAR**: Variable obtenida del paso anterior, que indica con un 1 que la bomba debería estar encendida.
- **8_MIN**: Variable que toma el valor 1 cuando se cumplieron 8 minutos bombeando.

Salidas

- **MOTOR**: Función que, con un 1, indica que el motor debe encenderse.
- **LUZ_AMARILLA**: Función que, con un 1, enciende una luz amarilla que indica que el sensor de nivel máximo debe ser revisado.

Realizando la tabla correspondiente (Tabla 3), y simplificando por Karnaugh, obtengo:

- **MOTOR** = ~8_MIN . BOMBEAR
- **LUZ_AMARILLA** = 8_MIN

Por último, el botón RESET es una entrada que afecta directamente a las salidas, poniendo a cero el temporizador, y apagando el motor y las luces amarilla y roja.

Si la entrada RESET va a la entrada de CLEAR del Flip Flop JK (activada a nivel bajo), la variable de salida BOMBEAR pasa a ser 0, por lo que el contador vuelve a cero, y el motor y la luz amarilla se apagan (pasan a valer 0).

Y en cuanto a la luz roja, si llamamos LUZ_ROJA_AUX a la variable obtenida anteriormente, la función quedaría definida:

LUZ_ROJA = ~RESET . LUZ_ROJA_AUX

En cuanto al temporizador, posee una entrada de RESET activada por nivel bajo, que vuelve la cuenta a 0. Esa entrada se define de la siguiente manera:

RESET = BOMBEAR

Ya que cuando no se está bombeando, el contador debe mantenerse en 0, y cuando sí se está bombeando, se libera para que empiece a contar.

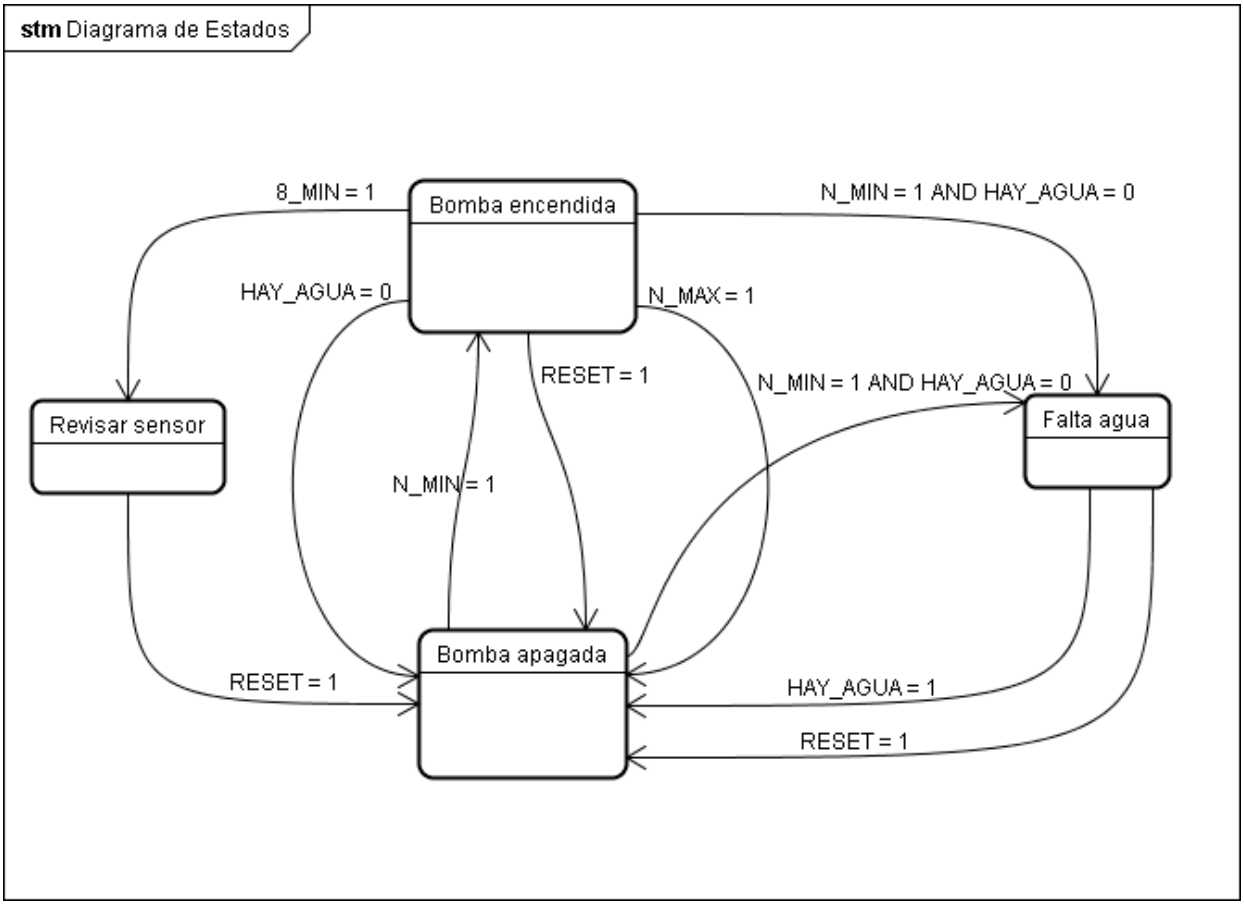
Además, el temporizador posee una salida que pasa a ser 1 cuando se alcanzan los 8 minutos, momento en el cual se detiene la cuenta. Y sólo se saldrá de dicho estado cuando BOMBEAR pase nuevamente a ser 0 (ya sea porque se llenó el tanque, o porque se presionó el botón de *reseteo*).

Dado que es un contador binario, y se usa la señal de 50 Hz de frecuencia (20 ms de período), para llegar a 8 minutos ($4{,}8 \times 10^5$ ms) se deben contar 24000 pulsos, por lo que se requerirán 15 flip flops ($15 = \lceil \log_2 24000 \rceil$), y el 24000, en binario, es 101110111000000. Por lo que, si llamamos Q_{14} al bit más significativo, y Q_0 al menos, obtenemos:

$8\_MIN = Q_{14} \cdot \sim Q_{13} \cdot Q_{12} \cdot Q_{11} \cdot Q_{10} \cdot \sim Q_9 \cdot Q_8 \cdot Q_7 \cdot Q_6 \cdot \sim Q_5 \cdot \sim Q_4 \cdot \sim Q_3 \cdot \sim Q_2 \cdot \sim Q_1 \cdot \sim Q_0$

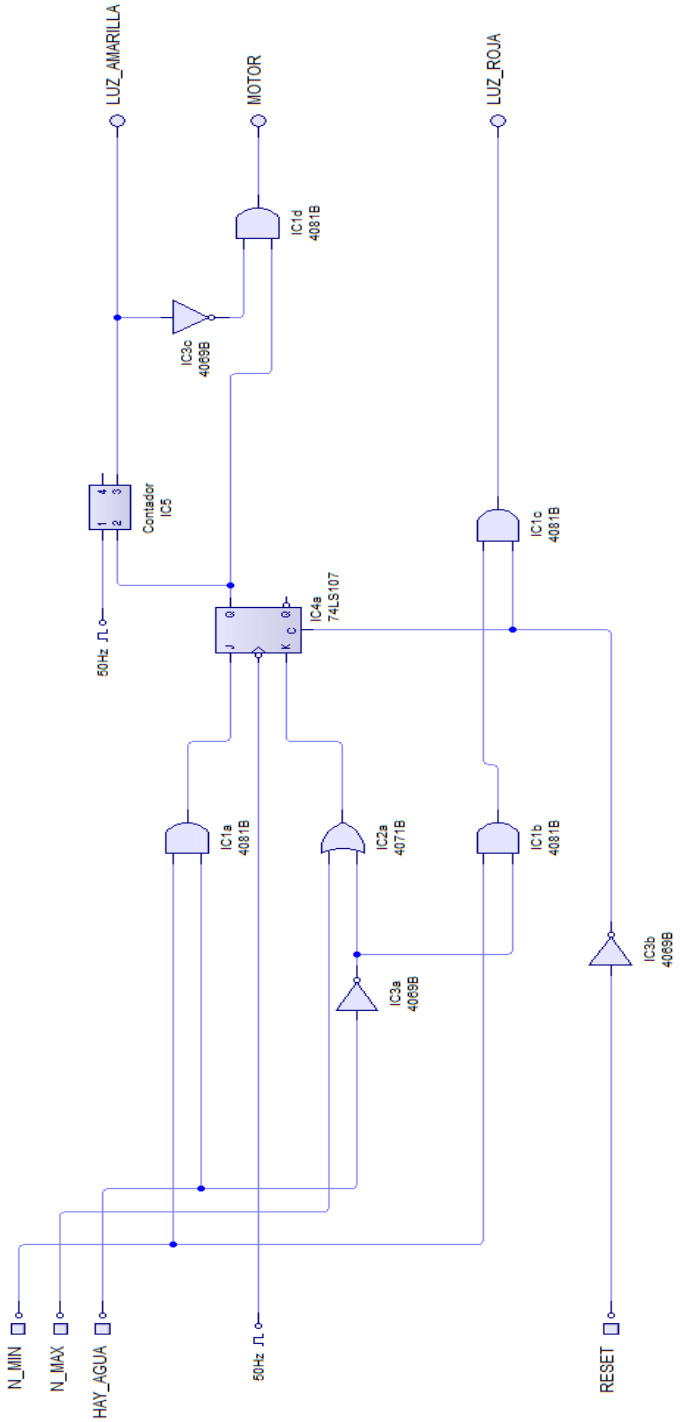
Su funcionamiento interno está detallado en la sección correspondiente a la solución cableada.

Diagrama de estados

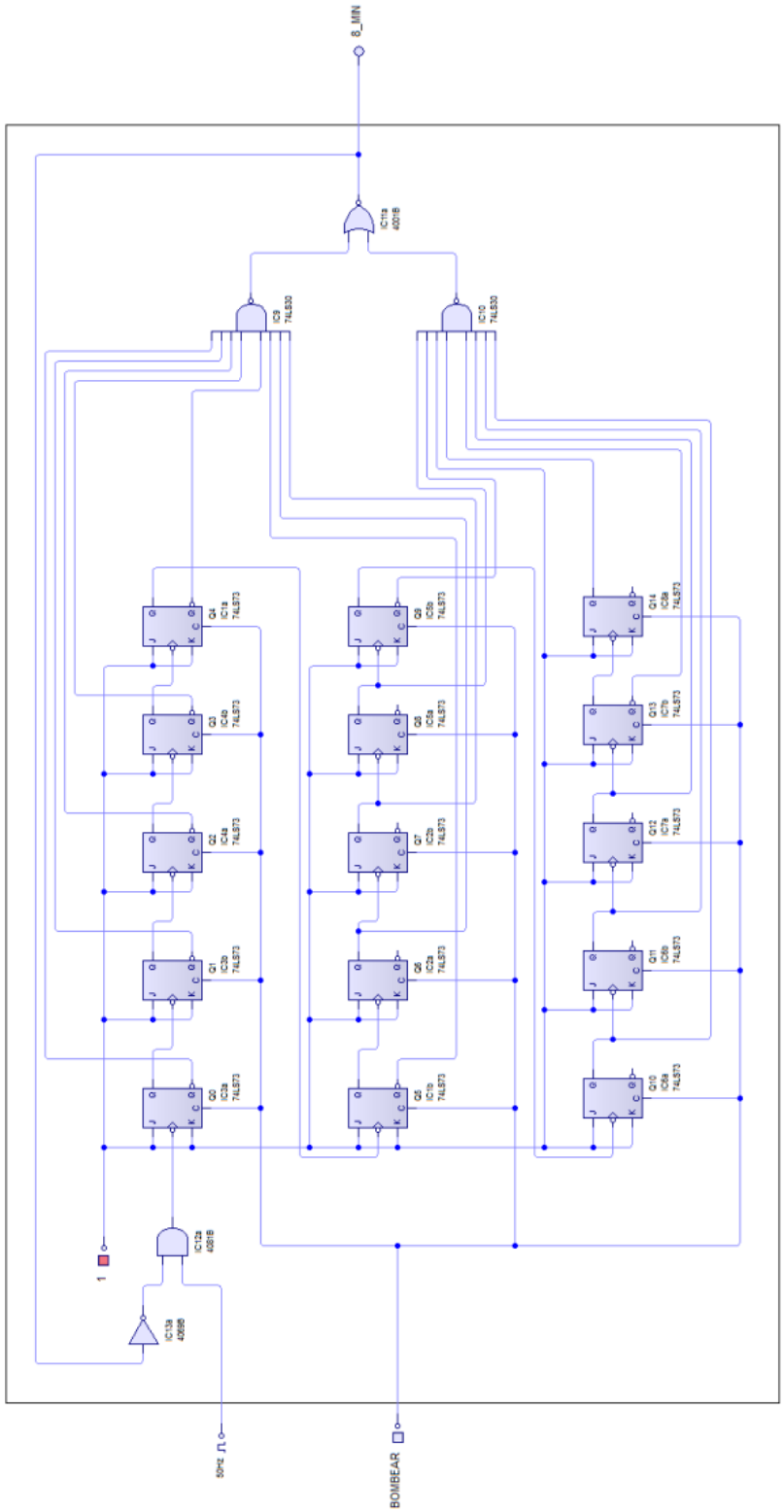


Solución cableada

Circuito Principal



Temporizador



Solución Programada

```
! MEMORIA
! [ENTRADA]: (MSB)...RESET,HAY_AGUA,N_MAX,N_MIN (LSB)
! [SALIDA]: (MSB)...MOTOR,LUZ_ROJA,LUZ_AMARILLA (LSB)

! REGISTROS
! %r1 = Entrada.
! %r2 = Auxiliar para la entrada.
! %r3 = Salida.
! %r4 = Auxiliar para la salida.
! %r5 = Variable para el contador.
! %r6 = Constante de 8_MIN.
! %r7 = Posición de memoria de la entrada.
! %r8 = Posición de memoria de la salida.

            .begin

            .macro      save_32     valor,registro
                        sethi       %hi(valor),registro
                        or          registro,%lo(valor),registro
            .endmacro

            .org 2048

ENTRADA     .equ 0x05000000
SALIDA      .equ 0x06000000
8_MIN       .equ 0x00044EF0
! Micro de 50 MHz, entre todo el ciclo, se realizan 14 operaciones y 2 acceso a
memoria (34 ciclos).
! 1 operación se hace en 1/50 * 10^6 segundos.
! Un ciclo del contador, en 34/50 * 10^6 segundos.
! Entonces, 8 min = 480 segundos equivalen a (480*50 / 34) * 10^6 ciclos = 282352 =
0x44EF0

            ba    INICIALIZACION

LECTURA:    ld    %r7,%r1           ! Guardo en %r1 lo que leo de la entrada.
            andcc %r1,8,%r2         ! %r2 <- (%r1 AND 8), enmascaro el bit del
botón RESET. Si da 1, es porque está apretado.
            be    COMPARO_N_MIN     ! Si no está apretado el RESET, sigo viendo
los demás bits de la entrada.
            ba    RESET             ! Si está apretado, reseteo.

COMPARO_N_MIN:
```

```
            andcc  %r1,1,%r2          ! Enmascaro el bit del nivel mínimo. Si da 1,
      es porque el agua se encuentra por debajo del nivel mínimo.
            be     ANALIZO_MOTOR      ! Si el agua no está por debajo del nivel
mínimo, analizo el estado del motor,
            ba     N_MIN              ! sino, actúo de acuerdo a este evento.

ANALIZO_MOTOR:
            andcc  %r3,4,%r4          ! Enmascaro el bit correspondiente al motor.
            be     LECTURA            ! Si está apagado, vuelvo a comenzar, ya que
no me interesa el estado de N_MAX
            ba     COMPARO_N_MAX      ! Sino, analizo N_MAX

COMPARO_N_MAX:
            andcc  %r1,2,%r2          ! Enmascaro el bit del nivel máximo. Si da 1,
      es porque se superó el nivel máximo.
            bne    APAGO_SOLO_MOTOR   ! Se superó el nivel máximo, entonces apago
el motor.
            ba     CONTADOR           ! Si no se superó, sabiendo que está
prendido, analizo el contador.

APAGO_SOLO_MOTOR:
            subcc  %r3,4,%r4          ! Como sé que el motor está encendido, resto
      1 en el bit correspondiente,
            st     %r4,%r8            ! apagando el motor.

RESET:      st     %r0,%r8            ! Apago el motor y todas las luces.
            ba     LECTURA            ! Vuelvo a comenzar.

N_MIN:      andcc  %r1,4,%r2          ! Enmascaro el bit de HAY_AGUA. Si da 1, es
porque no se pasó el límite inferior del agua en el reservorio.
            be     PRENDO_LUZ_ROJA    ! Si no hay agua, prendo la luz roja.
            ba     PRENDO_MOTOR       ! Si hay agua, y se alcanzó el nivel mínimo,
prendo el motor.

PRENDO_MOTOR:and   %r5,%r0,%r5        ! Inicializo el registro que voy a utilizar
como contador.
CONTADOR:   subcc  %r6,%r5,%r0        ! Comparo el estado del contador, con el
número equivalente a 8 minutos.
            be     PRENDO_LUZ_AMARILLA ! Si se cumplieron los 8 minutos, prendo la
luz amarilla.
            ld     %r7,%r1            ! Sino, cargo la entrada,
            add    %r5,1,%r5          ! incremento el contador,
            ba     LECTURA            ! y vuelvo a comenzar el ciclo.

APAGO_MOTOR: st    %r0,%r8            ! Apago el motor.
            ba     LECTURA

PRENDO_LUZ_ROJA:
```

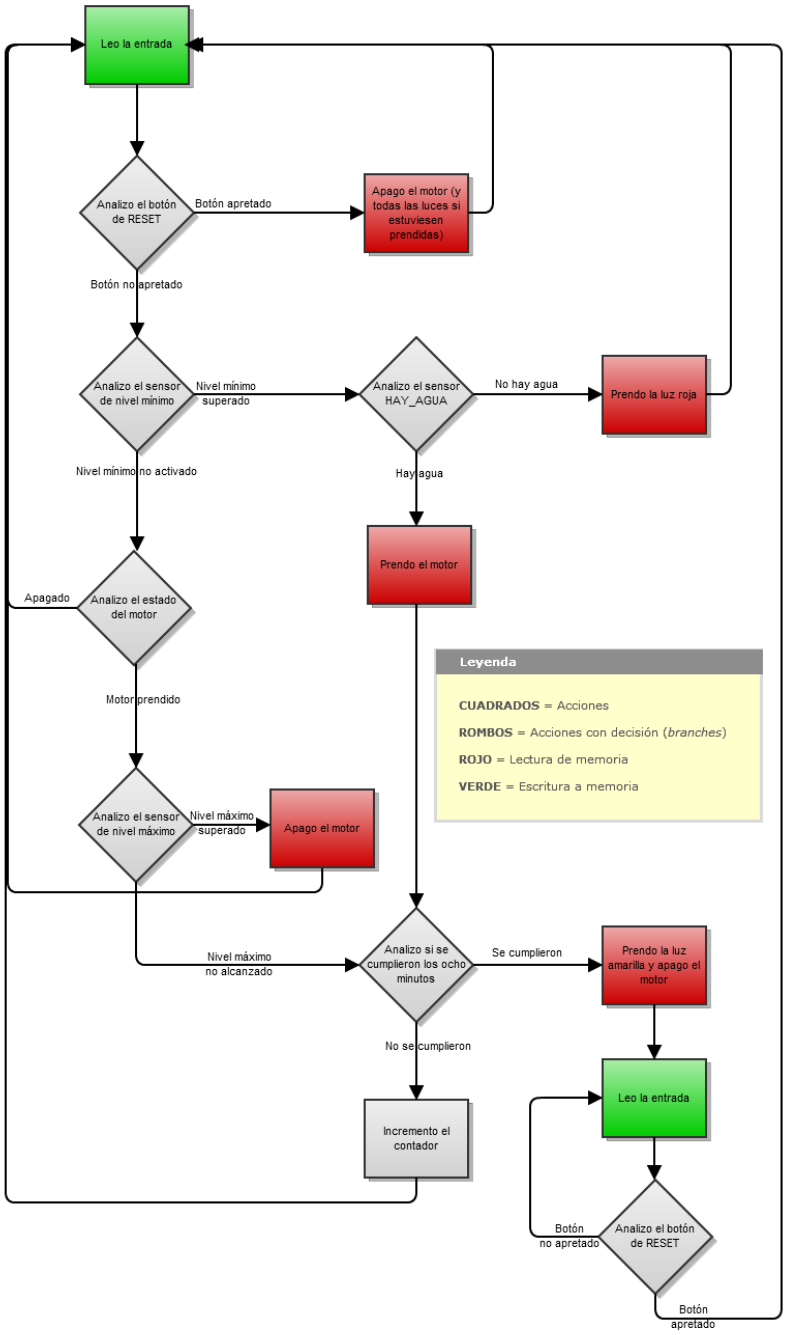
```
            add    %r0,2,%r3          ! Prendo la luz roja.
            st     %r3,%r8            ! Y luego lo mando por la salida.
            ba     LECTURA

PRENDO_LUZ_AMARILLA:
            add    %r0,1,%r3          ! Prendo la luz amarilla (y apago el motor).
            st     %r3,%r8            ! Y luego lo mando por la salida.
ESPERO_RESET:ld    %r7,%r1            ! Cargo la entrada,
            andcc  %r1,8,%r2          ! y enmascaro el bit de RESET.
            be     RESET              ! Si se apretó, reseteo todo.
            ba     ESPERO_RESET       ! Sino, espero. Ya que de esta situación sólo
se sale al apretar RESET.

INICIALIZACION:
            and    %r0,%r1,%r1              ! Inicializo en 0 el registro %r1.
            and    %r0,%r2,%r2              ! Inicializo en 0 el registro %r2.
            and    %r0,%r3,%r3              ! Inicializo en 0 el registro %r3.
            and    %r0,%r4,%r4              ! Inicializo en 0 el registro %r4.
            and    %r0,%r5,%r5              ! Inicializo en 0 el registro %r5.
            save_32      8_MIN, %r6
            save_32      ENTRADA, %r7
            save_32      SALIDA, %r8
            ba           LECTURA

            .end
```

Diagrama de Flujo



Conclusiones

Al momento de determinar cuál de las dos soluciones (cableada y programada) es mejor, se deben tener en cuenta varios factores, y evaluarlos de acuerdo a las necesidades de cada caso.

Entre ellos se encuentran el tiempo. Para el caso de la solución programada, teniendo en cuenta que el ciclo más largo posible, y despreciando el contador, ya que en ambos casos deben transcurrir 8 minutos, el tiempo dado es el siguiente:
Utilizando un clock de 50MHz, se realizan 14 instrucciones de un ciclo, y dos accesos a memoria (considerados de 10 ciclos cada uno), es decir, 34 ciclos. Lo que, para el caso de este clock, equivale a 680ns.
Ese mismo trayecto, en la solución cableada, nuevamente sin tener en cuenta el ciclo de los 8 minutos, toma la suma de las respuestas de cada uno de los componentes, lo que resulta en unos 350ns.

Otro factor importante a considerar, es el precio. En el caso de la solución programada, el costo más importante es el de la mano de obra, de quien programe en lenguaje ensamblador, a lo que se le sumaría el microprocesador.
Por otro lado, para la solución cableada, los costos están dados por los componentes a utilizar, además de, nuevamente, la mano de obra..

Por último, la solución programada **permite su modificación**, mientras que la cableada, por su naturaleza, no. Por lo que en caso de necesitar realizar un cambio, es muy probable que deba ser cambiada en su totalidad. Siendo la única excepción, el caso de agregar funcionalidad que haga uso, pero permita mantener intacta la solución inicial.

Apéndice

Tablas de verdad

N_MIN	N_MAX	HAY_AGUA	$BOMBEAR^{n}$		$BOMBEAR^{n+1}$	LUZ_ROJA
0	0	0	0		0	0
0	0	0	1		0	0
0	0	1	0		0	0
0	0	1	1		1	0
0	1	0	0		0	0
0	1	0	1		0	0
0	1	1	0		0	0
0	1	1	1		0	0
1	0	0	0		0	1
1	0	0	1		0	1
1	0	1	0		1	0
1	0	1	1		1	0
1	1	0	0		x	x
1	1	0	1		x	x
1	1	1	0		x	x
1	1	1	1		x	x

Tabla 1

N_MIN	N_MAX	HAY_AGUA		$BOMBEAR^n$	LUZ_ROJA		J	K
0	0	0		0	0		0	1
0	0	1		B^{n-1}	0		0	0
0	1	0		0	0		0	1
0	1	1		0	0		0	1
1	0	0		0	1		0	1
1	0	1		1	0		1	0
1	1	0		x	x		x	x
1	1	1		x	x		x	x

Tabla 2

8_MIN	BOMBEAR		MOTOR	LUZ_AMARILLA
0	0		0	0
0	1		1	0
1	0		x	x
1	1		0	1

Tabla 3

Mapas de Karnaugh

Para simplificar, utilizo los siguientes cambios de variable:

m = N_MIN
M = N_MAX
H = HAY_AGUA

J

m\M H	**00**	**01**	**11**	**10**
0	0	0	0	0
1	0	1	x	x

J = m . H

K

m\M H	**00**	**01**	**11**	**10**
0	1	0	1	1
1	1	0	x	x

K = M + ~H

LUZ_ROJA

m\M H	**00**	**01**	**11**	**10**
0	0	0	0	0
1	1	0	x	x

LUZ_ROJA = m . ~H

Los cambios de variable realizados en el siguiente mapa son:
8 = 8_MIN
B = BOMBEAR

MOTOR

8\B	**0**	**1**
0	0	1
1	x	0

MOTOR = ~8 . B

LUZ_AMARILLA

8\B	**0**	**1**
0	0	0
1	x	1

LUZ_AMARILLA = 8