Sistemas embebidos 7M - TLAR

MANEJO DE ENTRADAS Y SALIDAS DIGITALES

Nicolas Alberto Arias[1], Pedro Andrés García[2], & Cristian Salazar Valencia[3]

[1] *estudiantes tecnología automatización y robótica industrial – Escuela Colombiana de Carreras Industriales, Bogotá, Colombia*

I. INTRODUCCION

En esta práctica de laboratorio se implementa el uso de una tarjeta STM32 NULCLEO F411RE, en el cual debemos conocer su variedad de entradas y salidas digitales, con el fin de ver su comportamiento en dicha salida por medio de leds y un display de 7 segmentos.

II. OBJETIVOS

Conocer y aprender las entradas y salidas digitales del nucleo F411RE, realizando progamas usando leds y display 7 segmentos.

III. OBJETIVOS ESPECÍFICO

- Aplicar el concepto de entradas y salidas digitales.
- Fortalecer el uso de mbed de manera offline a través de Keil.
- Conocer los periféricos básicos de la tarjeta de desarrollo (led de usuarios y Botón de usuario)

IV. MATERIALES

- STM32 NUCLEO F411RE.
- 8 leds.
- 2 display de 7 segmentos.
- Dips witch
- 8 resistencias 220 ohmios.

V. EJERCICIO 1

. Genere un código que le permita obtener visualizar un contador de 0 a 16. Para las entradas del contador puede utilizar un Dip Switch.

ANALISIS:

Para ellos generamos un algoritmo en el cual nos brindara una guía de cómo debemos ejecutar el contador con el display de 7 segmentos. (2), con el fin de logran un conteo ascendente y repetitivo para ellos se planteó el siguiente algoritmo:

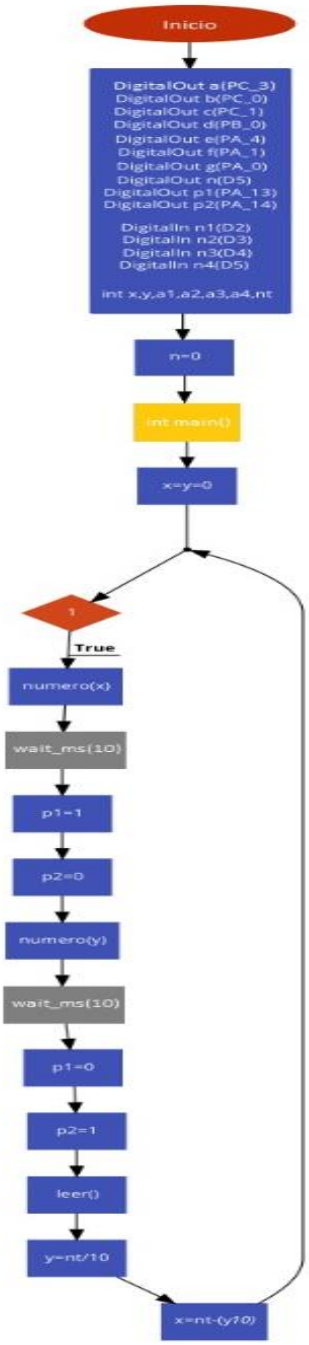
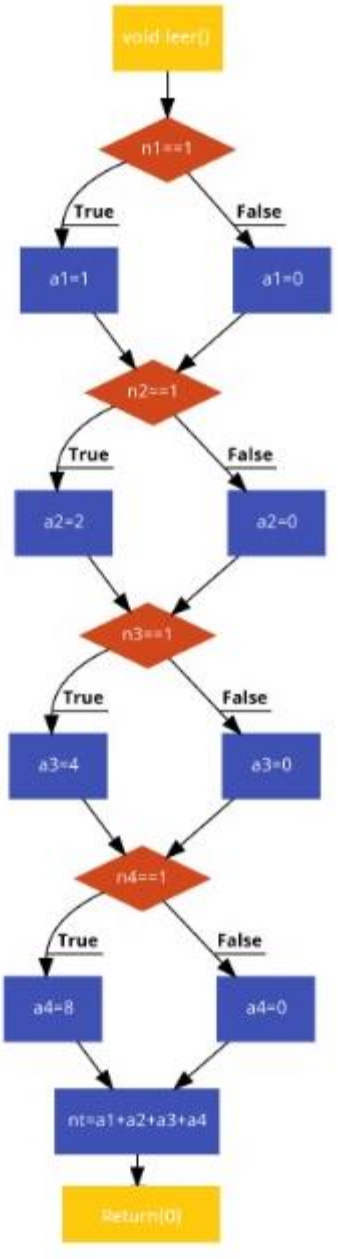


FIGURA 1; EJERCICIO 1

Primero debemos tener todas nuestras condiciones iniciales y definir la entras y salidas que requerimos, luego de ellos usamos una serie de variables enteras (int) que establecimos ya que al inicial en el int main() tomamos unas condiciones iniciales en 0 y desde hay se hace una un seguimiento del cual nos brindara un conteo de lectura y empleamos un arreglo para dividir el una variable (y) en 10 y continuamos con la siguiente variable (x) y restamos por la variable anterior y así para cumplir dicho ciclo del conteo.



FIGURAL 2: EJERCICIO 1.

Determinamos una subrutina con el fin de hacer una secuencia para que nos ayude hacer el conteo de 8 bits en el cual se asignara un resultado que se dirigirá a la secuencia establecida y requerida.

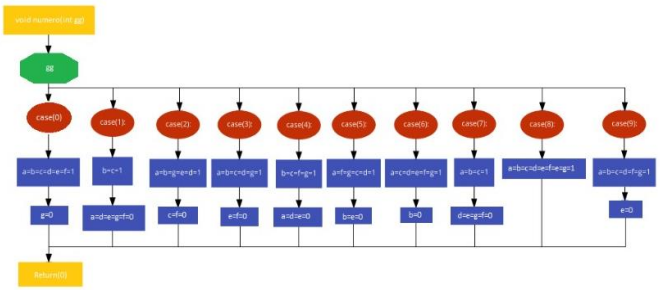


FIGURA 3: EJERCICIOS 1

En este diagrama establecimos una serie de caso en el cual el display tomara una variedad de valores digitales ya se 0 o 1 en todos los casos posibles y así poder cumplir con la visualización del contador y cumplir con el fin de 0 a 15 ya que es una secuencia digital.

El criterio del switch case establece una serie de criterios que debemos tener encuentra a la hora de implementar esta ayuda, ya que nos da un orden de los casos establecidos con parámetros definidos y con una secuencia para cumplir dichas condiciones a establecer en este primer laboratorio.

CODIGO:

```
#include "mbed.h"
// DEFINICION DE PUERTOS

DigitalOut a(PC_3);
DigitalOut b(PC_0);
DigitalOut c(PC_1);
DigitalOut d(PB_0);
DigitalOut e(PA_4);
DigitalOut f(PA_1);
DigitalOut g(PA_0);
DigitalOut n(D5);
DigitalOut p1(PA_13);
DigitalOut p2(PA_14);
DigitalIn n1(D2);
DigitalIn n2(D3);
DigitalIn n3(D4);
DigitalIn n4(D5);

int x,y,a1,a2,a3,a4,nt;
```

```
void numero(int gg)
{
    switch(gg)
    {
        case(0):
        {
            a=b=c=d=e=f=1;
            g=0;
            break;
        }
        case(1):
        {
            b=c=1;
            a=d=e=g=f=0;
            break;
        }
        case(2):
        {
            a=b=g=e=d=1;
            c=f=0;
            break;
        }
        case(3):
        {
            a=b=c=d=g=1;
            e=f=0;
            break;
        }
        case(4):
        {
            b=c=f=g=1;
            a=d=e=0;
            break;
        }
        case(5):
        {
            a=f=g=c=d=1;
            b=e=0;
            break;
        }
        case(6):
        {
            a=c=d=e=f=g=1;
            b=0;
            break;
        }
        case(7):
        {
            a=b=c=1;
            d=e=g=f=0;
            break;
        }
        case(8):
        {
            a=b=c=d=e=f=g=1;
            break;
        }
        case(9):
        {
            a=b=c=d=f=g=1;
            e=0;
            break;
        }
    }
    n=0;
}
void leer()
{
    if(n1==1) {
        a1=1;
    } else {
        a1=0;
    }
    if(n2==1) {
        a2=2;
    } else {
        a2=0;
    }
    if(n3==1) {
        a3=4;
    } else {
        a3=0;
    }
    if(n4==1) {
        a4=8;
    } else {
        a4=0;
    }
    nt=a1+a2+a3+a4;
}
```

```
int main()
{
    x=y=0;
    while (1)
    {
        numero(x);
        wait_ms(10);
        p1=1;
        p2=0;
        numero(y);
        wait_ms(10);
        p1=0;
        p2=1;
        leer();
        y=nt/10;
        x=nt-(y*10);
    }
}
```

VI. EJERCICO 2

Hacer un código para obtener por un puerto del microcontrolador: el dato del puerto A desplazando un bit hacia la izquierda ejemplo: Ingreso A ‘xxx11001’ entonces:'xx110011’

ANALISIS:

Establecemos los leds de salida que van a mostrar los bits que están entrando por el dip switch, para ello se establecen dos partes, la primera parte es cuando los 4 primeros bits me van a establecer el set con que viene cargado el bus de datos, luego con el botón de usuario se van ingresado uno a uno los bits consecutivos de la trama, para ello en valor del anterior bit se establece en el siguiente y así sucesivamente hasta llegar al último, donde el ultimo va a tomar el valor que tenga el dip switch.

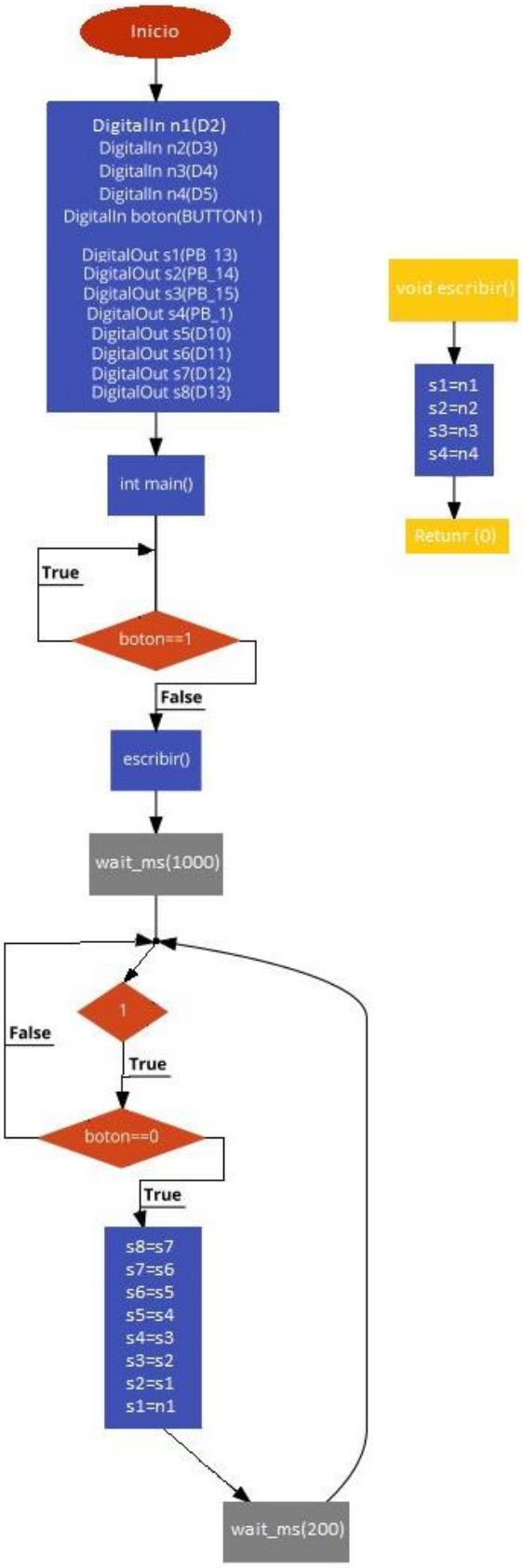


FIGURA 1: EJERCICIO 2

```
#include "mbed.h"
DigitalIn n1(D2);
DigitalIn n2(D3);
DigitalIn n3(D4);
DigitalIn n4(D5);
DigitalOut s1(PB_13);
DigitalOut s2(PB_14);
DigitalOut s3(PB_15);
DigitalOut s4(PB_1);
DigitalOut s5(D10);
DigitalOut s6(D11);
DigitalOut s7(D12);
DigitalOut s8(D13);
DigitalIn boton(BUTTON1);

void escribir()
{
   s1=n1;
   s2=n2;
   s3=n3;
   s4=n4;
}
int main() {
   while (boton==1){}
   escribir();
   wait_ms(1000);
   while(1)
   {
      if(boton==0)
      {
         s8=s7;
         s7=s6;
         s6=s5;
         s5=s4;
         s4=s3;
         s3=s2;
         s2=s1;
         s1=n1;
         wait_ms(200);
      }
   }
}
```

VII. EJERCICIO 3

Realice una suma de dos números de 2 bits cada uno, el resultado debe ser mostrado por medio de display 7 segmentos

ANALISIS:

Establecemos los mismo parámetros del ejercicios 1, del cual haremos una serie de subrutinas que nos ayudar a tomar una variedad de condiciones tanto la iniciales como las finales y obtener una variedad de respuestas, teniendo en cuenta los parámetros de la suma (a+b = c) con esto demos tomar los bits tomados de la entrada y hacer una serie de sumas entre la variables enteras que definimos que se establecen en la subrutinas para así ser llamada y obtener una respuesta de dichos bits que establece el usuario.

Algoritmo establecido:

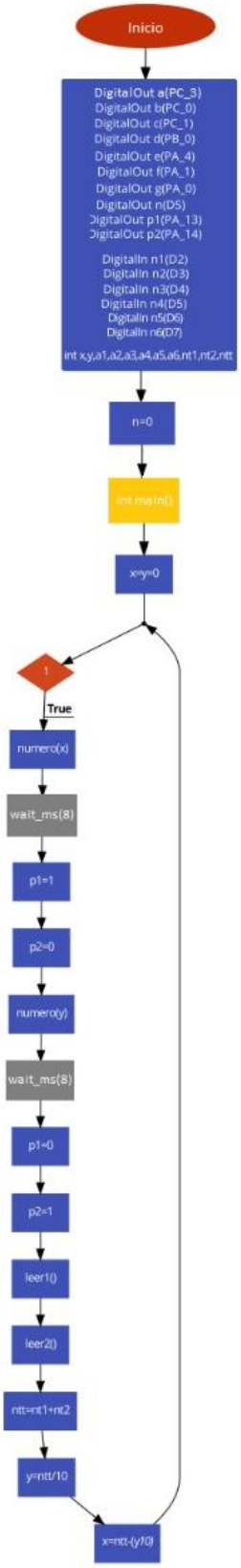


FIGURA 1: EJERCICIO 3

Establecemos la misma condición del ejercicio 1, pero con una variedad de variables enteras que serás las que nos cumplan la serie sumadas en las entradas dadas.

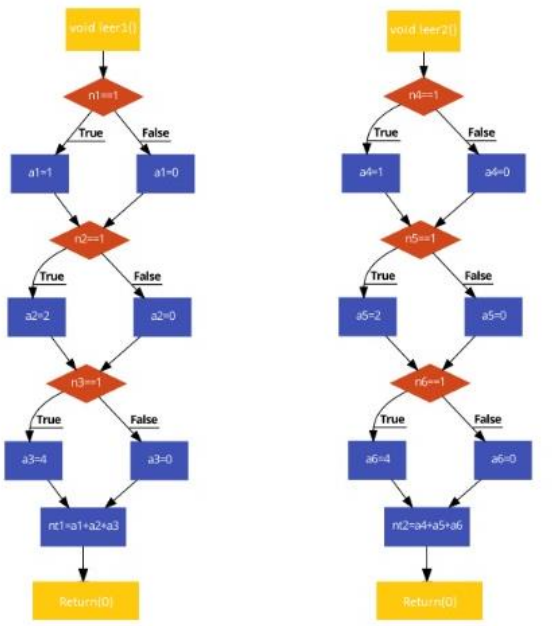


FIGURA 2: EJERCICO 3

Se toma una serie de subrutinas que nos representan de nuestros sumadores con el fin de contar cada bit dado en las entradas y obtener una respuesta digital y visualizada en el display de 7 segmentos, con el fin de si tomamos un valor de entrada en binario en Dip1 = 0001 y Dip2 = 0001 obtendremos una respuesta visual de un 02 en digital en el display de 7 segmentos.

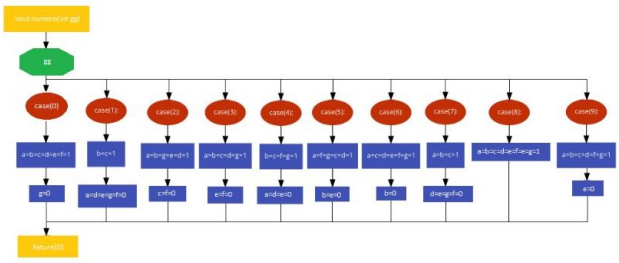


FIGURA 3: EJERCICIO 3

Establecemos los mismos parámetros aplicados y el uso de un switch case, en el cual toma las misma variables y valores dados en una secuencia digital ya sea 0 o 1 para el encendido consecutivo de cada número a visualizar en el display de 7 segmentos de (0, 1, 2, 3, 4, 5, 6,7, 8,9) en el cual obtendremos una respuesta en la salida segundo los valores dados en la entrada.

CODIGO:

```
#include "mbed.h"

DigitalOut a(PC_3);
DigitalOut b(PC_0);
DigitalOut c(PC_1);
DigitalOut d(PB_0);
DigitalOut e(PA_4);
DigitalOut f(PA_1);
DigitalOut g(PA_0);
DigitalOut n(D5);
DigitalOut p1(PA_13);
DigitalOut p2(PA_14);
DigitalIn n1(D2);
DigitalIn n2(D3);
DigitalIn n3(D4);
DigitalIn n4(D5);
DigitalIn n5(D6);
DigitalIn n6(D7);

int x,y,a1,a2,a3,a4,a5,a6,nt1,nt2,ntt;
void numero(int gg)
{
    switch(gg) {
        case(0): {
            a=b=c=d=e=f=1;
            g=0;
            break;
        }
        case(1): {
            b=c=1;
            a=d=e=g=f=0;
            break;
        }
        case(2): {
            a=b=g=e=d=1;
            c=f=0;
            break;
        }
        case(3): {
            a=b=c=d=g=1;
            e=f=0;
            break;
        }
        case(4): {
```

```
            b=c=f=g=1;
            a=d=e=0;
            break;
        }
        case(5): {
            a=f=g=c=d=1;
            b=e=0;
            break;
        }
        case(6): {
            a=c=d=e=f=g=1;
            b=0;
            break;
        }
        case(7): {
            a=b=c=1;
            d=e=g=f=0;
            break;
        }
        case(8): {
            a=b=c=d=e=f=g=1;
            break;
        }
        case(9): {
            a=b=c=d=f=g=1;
            e=0;
            break;
        }
    }
    n=0;
}
void leer1()
{
    if(n1==1)
    {
        a1=1;
    }
    else
    {
        a1=0;
    }
    if(n2==1)
    {
        a2=2;
    }
    else
    {
        a2=0;
    }
    if(n3==1)
    {
        a3=4;
    }
    else
    {
        a3=0;
    }
    nt1=a1+a2+a3;
}
void leer2()
{
    if(n4==1)
    {
        a4=1;
    }
    else
    {
        a4=0;
    }
    if(n5==1)
    {
        a5=2;
    }
    else
    {
        a5=0;
    }
    if(n6==1)
    {
        a6=4;
    }
    else
    {
        a6=0;
    }
    nt2=a4+a5+a6;
}
int main()
{
    x=y=0;
    while (1) {
        numero(x);
        wait_ms(8);
        p1=1;
        p2=0;
```

```
        numero(y);
        wait_ms(8);
        p1=0;
        p2=1;
        leer1();
        leer2();
        ntt=nt1+nt2;
        y=ntt/10;
        x=ntt-(y*10);
    }
}
```

VIII.Conclusión

- La implementación de subrutinas nos ayuda a tener un orden a la hora de hacer un llamado para que cumpla un ciclo determinado.
- Con el uso del switch case establecer una variedad de caso con el fin de obtener una serie de paso a paso de cada programa.
- La secuencia digital es muy necesaria para la cumplir una variedad de propósitos o tareas solo usando llamados de bits a bits ya con ello obtenemos un control mucho más preciso.
- Se pudo implementar el uso de bits para cumplir una serie proporcional en los casos establecidos en el uso del switch case, en cual también se pudo implementar ya sea binario para la realización de la misma,
- Usando Dip Switch pudimos cargar bits uno a uno por medio del User button.
- Usando el criterio del primer programa podemos sumar dos números de tres bits y visualizarlo mediante el display 7 segmentos.