ENTREGA DE PRIMER CORTE: CONCEPTOS BÁSICOS DEL JUEGO TETRIS MEDIANTE LA MATRIZ MAX7912 Y STM32F411

LINA ROCÍO ÁVILA RODRÍGUEZ- 30711

ANDRÉS FELIPE CANO SALAMANCA- 58794

ALYSON VALERIA CASTIBLANCO CASTAÑEDA- 40779

PRESENTADO A: FERNEY BELTRÁN

SISTEMAS EMBEBIDOS

UNIVERSIDAD ECCI

2018

Introducción:

A continuación, se mostrará en detalle el funcionamiento inicial del juego Tetris desarrollado a través de la tarjeta STM32F411 y la matriz de Leds 8x8 controlada por el MAX7912, desde la justificación de su uso hasta cómo se programaron las comunicaciones, así como el traslado de la información para finalmente ser visualizada en la matriz antes mencionada.

JUEGO DE TETRIS: FUNDAMENTOS GENERALES Y PROGRAMACIÓN:

Tarjeta: En este caso se usa la tarjeta **STM FM411 RE** [1], usa un procesador **ARM CORTEX M4,** el cual trabaja a una velocidad máxima de 100 MHZ, y adicional mente, contiene las siguientes especificaciones:

Voltaje de alimentación: 1.7v-3.6 V.

Voltaje de salida: 3.3v-5v

Interfaces de comunicación:

3 módulos I2C
3 módulos USART
5 módulos SPI (SPI-I2SS)
1 módulo SDIO
1 módulos a conexión USB, puerto 2.0 de alta velocidad.

Conversor análogo a digital: Posee un conversor hasta de 16 canales con resolución de 12 bits.

Conexión de pines:

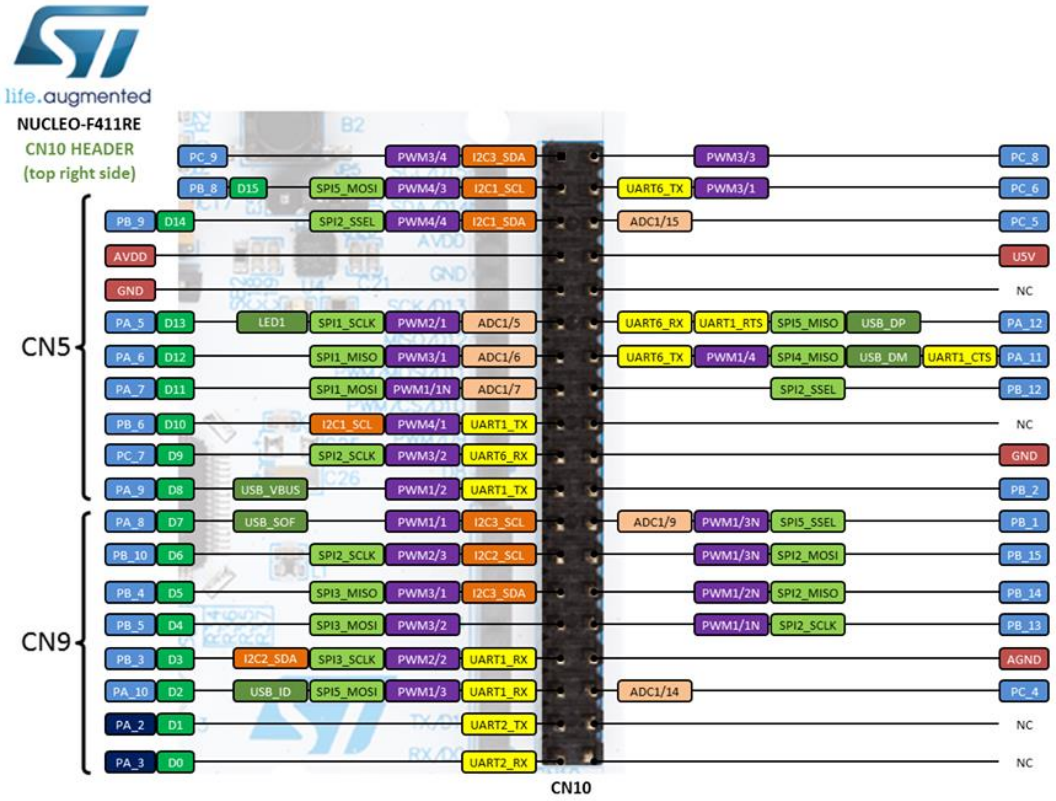


Figura 1: Pines del módulo CN10 de la tarjeta mencionada. [2]

Tarjeta:

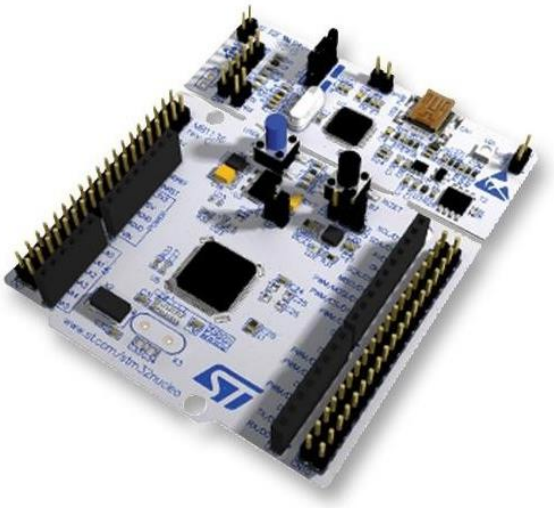


Figura 2: Tarjeta Núcleo STM32F411. [2]

Periféricos

Para este caso, se usa la matriz led de 8*8, adicionalmente, se usa el integrado **max 7219,** el cual facilita la comunicación y e de leds por medio del control multiplexado, permitiendo disminuir la cantidad de pines a la hora de manejar la matriz de led.

La conexión con la tarjeta se realiza de la siguiente forma:

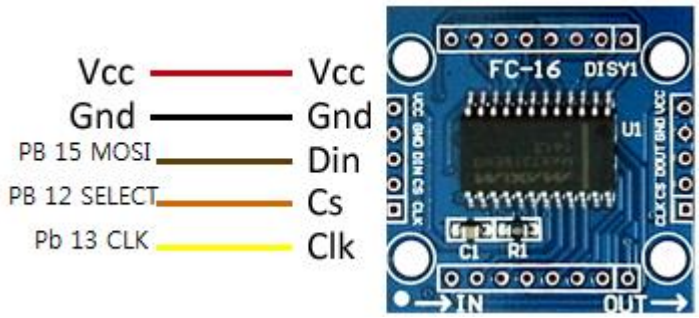


Figura 3: Pines de conexión del MAX7912 en Arduino. Equiparable con la tarjeta a trabajar durante el proyecto a excepción de la numeración de los pines [3]

Pines.

Pin 13: Contiene la conexión al reloj, esta permite la sincronización entre la matriz de led y la tarjeta de forma síncrona.

Pin 15: pin referente a la salida de información de la memoria maestra de la tarjeta, este pin va conectado directamente al pin de la matriz, conocido como MOSI.

Pin 12 : Permite generar la selección de manera digital del ingreso y salida de información, según corresponda.

Código:

Protocolo de comunicación

Se realiza comunicación de la tarjeta a la matriz por medio del protocolo SPI, donde se utiliza el SPI2, con comunicación full dúplex, el cual genera el puente de trasmisión y recepción de datos entre la tarjea y la matriz, esto permite que en ambos elementos se realice la sincronización por medio de tiempos de reloj, para indicar los pines que se van usar , de acuerdo a la funcionalidad de los mismos, se debe generar la siguiente la declaración de puertos, la cual se escribe de la siguiente forma:

```
1
2 #include "mbed.h"
3 #include "figuras.h"
4
5 SPI deviceM(PB_15, PB_14, PB_13);
6 DigitalOut ssel (PB_12);
7 Serial command(USBTX, USBRX);
8
9 #define DEBUG 1
10 #define  VELOCITY 200 // ms
```

Pin 13: Contiene la conexión al reloj, esta permite la sincronización entre la matriz de led y la tarjeta

Pin 14: permite el acceso a la memoria maestra de la tarjeta y a la información de la misma, comúnmente conocida como MISO

Pin 15: pin referente a la salida de información de la memoria maestra de la tarjeta, este pin va conectado directamente al pin de la matriz, conocido como MOSI.

Clase:

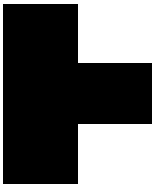
Clase **Figuras:**

Al generar esta clase manualmente, se realizó el diseño de las 4 figuras comunes para un Tetris por medio del uso de matrices, indicando de manera binaria que led se debían encender y cuales, también se escribieron las matrices, de las mismas figuras, pero rotadas, tal como lo muestra la siguiente imagen.

T:

0 1 0
1 1 1
0 0 0

T1

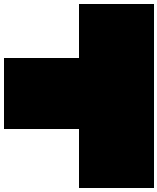


0 1 0
0 1 1
0 1 0

T2

1 1 1
0 1 0
0 0 0

T3



0 1 0
1 1 0
0 1 0

I

1 0 0
1 0 0
1 0 0

I1

1 1 1
0 0 0
0 0 0

I2

0 1 0
0 1 0
0 1 0

I3

0 0 0
1 1 1
0 0 0

L

1 0 0
1 0 0
1 1 0

L1

1 1 1
0 0 1
0 0 1

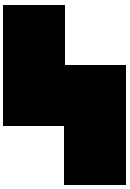
L2

0 0 0
0 0 1
1 1 1

L3

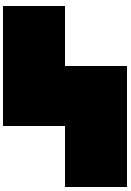
1 1 1
0 0 1
0 0 1

0 1 1
1 1 0
0 0 0



1 1 0
0 1 1
0 0 1

0 1 1
1 1 0
0 0 0



1 1 0
0 1 1
0 0 1

Además de esto, también se puede observar la definición de cada una de las maneras de llamarlas dentro del código principal.

Declaración de funciones

Se declararán con el objetivo de evitar repetir códigos con funciones específicas dentro de la rutina que se está desarrollando, para este caso, las funciones más utilizadas son :

Matrix_act: Recoge la información solicitada

Matrix_tmp: se adiciona la información inicial para imprimir físicamente la información que contiene, en este caso, la codificación de cada una de las figuras de tetris

Array: contiene la trama con la cual se le indica a la tarjeta la información que deseamos que sea impresa en la matriz de led

Init_display: Prepara la matriz para la comunicación con la tarjeta, usando todas las filas y columnas, y configurando la intensidad de la iluminación, para este caso, indicando que no se realizara la decodificación de datos.

Finalmente realiza el testeo de la matriz, solicitando que todos los leds enciendan y se apaguen 4 veces, con el fin de que el usuario verifique que todos los leds funcionen correctamente.

Debug_m: ingresa una variable char, e imprime la información dentro de la variable.

Recibe: declara variable tipo char, la cual guarda la información que ingresa del puerto serial, en este caso, para inicializar la trama, espera a que se use el carácter “0x3C”, después de lo cual, los siguientes 3 datos ingresados, indican la trama indicando la figura, Angulo y columna.

Finalmente, después de ejecutar la información contenida, vuelve a solicitar otro dato del puerto serial, por lo cual, para iniciar la trasmisión, se solicita el ingreso del carácter “0X3E” el cual retorna un “1” en caso de que el carácter sea correcto, de lo contra rio retorna un “0”.

En caso de recibir “0”, no se realizará ninguna ejecución, y el código seguirá solicitando que sea ingresada la información correcta para iniciar la trasmisión.

Cuando se recibe “1”, antes de generar la trasmisión de la información se usa la función **NEW_READ**, la cual procesa la información que tiene en el arreglo de la misma y de tal manera de que se imprima la información solicitada en la trama.

Para seleccionar la figura adecuada, se utiliza un switch anidado, en donde se indaga sobre la posición inicial del arreglo, dentro de este, se selecciona el ángulo requerido, y así mismo, también se indica la posición de la columna que se desea usar tal como se puede visualizar en la siguiente imagen .

main.cpp | main.cpp

```
112
113 void new_read(){
114     switch (array[0]) {
115     case  L_TYPE:
116     switch (array[1])
117          {  case NOVEN:
118              switch(array[2])
119              {case 1:
120              copy_matrix(FIG_L1,1);
121              break;
122              case 2:
123              copy_matrix(FIG_L1,2);
124              break;
125              case 3:
126              copy_matrix(FIG_L1,3);
127              break;
128              case 4:
129              copy_matrix(FIG_L1,4);
130              break;
131              case 5:
132              copy_matrix(FIG_L1,5);
```

Cuando se determina la columna que se va a usar, se envía información a la función **COPY_MATRIX** en la cual se indican dos datos: el tipo de figura (guardada en la librería *figuras.h*) y tipo de columna.

Al obtener la información correspondiente, almacena los datos obtenidos en la **matrix_temp**, la cual tiene un apuntador que permite tomar el registro, y hacer el corrimiento a la columna indicada.

Finalmente se hace la resta de 1, para minimizar el margen de error al generar el corrimiento de la figura, ya que este inicia en 0.

Para trasmitir la información a la matriz, se envía los datos contenidos en matrix_**temp** a la función **int_fig,** la cual se encarga de tomar a **matrix_temp** y colocarla en un for para que se organice la trama y se envíen los registros por medio del SPI, enviando en primera medida, el tipo de figura, y en último, la columna en la cual va a ser visualizada.

SEND_SPI: recibe 2 datos, el registro, y la información que desea ser visualizada, por lo cual en primera medida, se genera la habilitación, e indaga sobre la comunicación en el SPI, y dependiendo del registro que está solicitando la información, se habilita, y realiza en el envió de la misma.

Al finalizar la trasmisión, deshabilita, y espera que nuevamente ingrese información para ser mostrada en la matriz.

Bibliografía

[1] ST, «STM32F411RE Datasheet,» 2018. [En línea]. Available: https://www.st.com/resource/en/datasheet/stm32f411re.pdf.

[2] T. STM, «NUCLEO-F411RE,» 2018. [En línea]. Available: https://os.mbed.com/platforms/ST-Nucleo-F411RE/.

[3] C. D. PINES, «LUIS LLAMAS,» 2016. [En línea]. Available: .