STM32F429ZI Núcleo Photobooth

Rosas Morandeira, Laura de la Caridad

Universidad Latina de Panamá

Resumen – Un microcontrolador hace referencia a un circuito integrable que puede ser programado para realizar diversas tareas y funciones, dependiendo del código y las librerías que se tengan en el mismo. Este trabajo se enfoca en la realización de un módulo de photobooth o carrete de imágenes.

I. INTRODUCCIÓN

Se llama microcontrolador a un circuito integrado digital que puede ser usado para muy diversos propósitos debido a que es *programable*. Está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos).

Un microcontrolador puede ser utilizado para diversas aplicaciones, algunas de incluyen el manejo de sensores, controladores, juegos, calculadoras, agendas, avisos lumínicos, secuenciador de luces, cerrojos electrónicos, control de motores, relojes, alarmas, robots, entre otros.

Para este caso en particular el proyecto que se lleva a cabo es la realización de un photobooth o carrete de fotos que muestre fotos y cuente con botones para pasar entre ellas.

El proyecto tiene como objetivo general la realización de un photobooth o carrete de fotos.

Como objetivos específicos, se incluyen los siguientes:

- Lograr el manejo de los leds del microcontrolador.
- Configurar apropiadamente una unidad FAT-32 para guardar datos.
- Configurar la Interfaz Gráfica de Usuario en el microcontrolador.
- Acceder a la pantalla táctil del microcontrolador.
- Desarrollar un photobooth mediante código.

II. MARCO TEÓRICO

a. ¿Qué es un microcontrolador?

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.

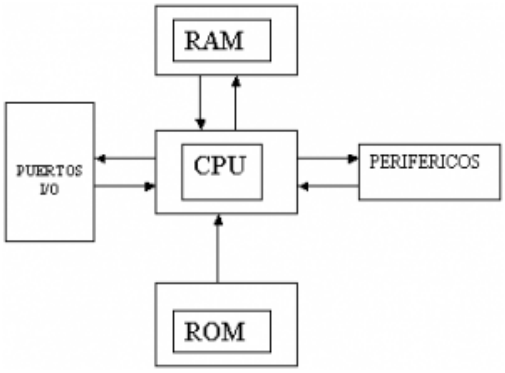

Figura 1. *Componentes de un microcontrolador.*

Toda microcomputadora requiere de un programa para que realice una función específica. Este se almacena normalmente en la memoria ROM. No está de más mencionar que sin un programa, los microcontroladores carecen de utilidad.

El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe, es por esto que la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. El carácter programable de los microcontroladores simplifica el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador.

b. STM32F429 - Discovery

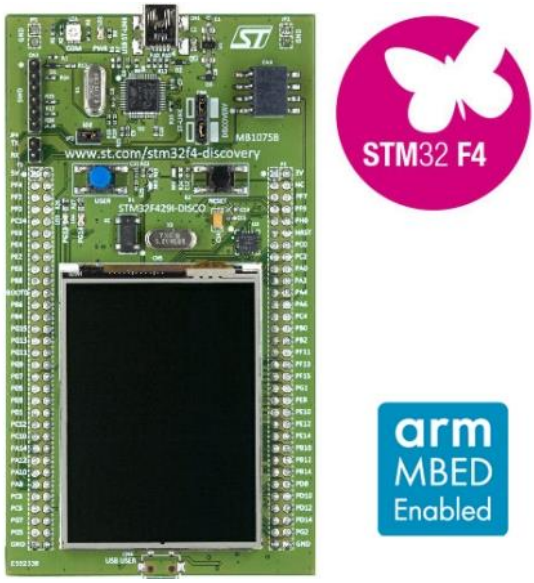

Figura 2. *STM32F429I-Discovery.*

El microcontrolador que se utilizará es el STM32F429 – Discovery, el cual cuenta con 2Mbytes de memoria flash y 256 Kbytes de RAM. Este microcontrolador les da un excelente uso a las capacidades de los controladores STM32F429, permitiendo a los usuarios desarrollar aplicaciones completas con Interfaces Gráficas de Usuario avanzadas, lo cual es sumamente útil para este proyecto, ya que el uso de la pantalla es clave para que se pueda observar la imagen de la cámara.

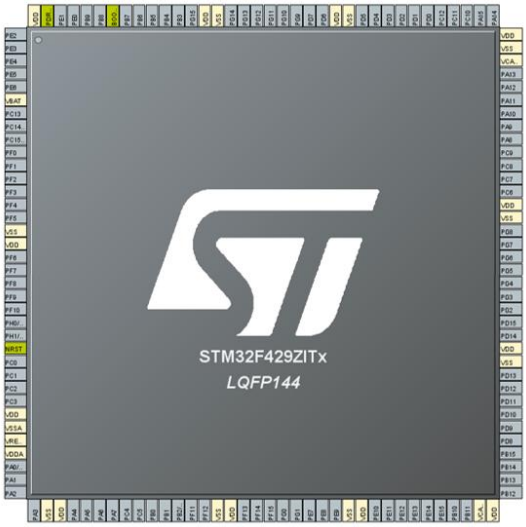

Figura 3. Pines del *STM32F429I-Discovery.*

En la imagen superior se observan los diversos pines del microcontrolador, los cuales al ser habilitados son los que permiten se configuren los LEDS, la pantalla, el puerto USB y otros aspectos más.

III. DESARROLLO

a. Configuración de los LEDS

Para lograr la configuración apropiada de los leds, lo primero que se debe examinar es el Manual de Usuario del microcontrolador. En el mismo se nos dice el los LD3 y LD4 son los leds de usuario los cuales están conectados al PG13 y PG14 del microcontrolador respectivamente.

Estos son los pines que utilizaremos para habilitar los leds del microcontrolador. Para ello, hay que tomar en cuenta que se deben habilitar los HAL_GPIO, es decir los General Purpose Input/Output. Lo mismo se realiza con el siguiente código:

```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOF_CLK_ENABLE();
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
__HAL_RCC_GPIOG_CLK_ENABLE();
__HAL_RCC_GPIOE_CLK_ENABLE();
__HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13|GPIO_PIN_14, GPIO_PIN_RESET);

  /*Configure GPIO pins : LD3_Pin LD4_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
}
```

Ya al estar activados, pueden ser llamados cuando sea necesario. En nuestro caso lo llamamos para que parpadee con la siguiente línea de código:

```
HAL_GPIO_TogglePin(GPIOG,GPIO_PIN_13|GPIO_PIN_14);
```

La misma llama a los pines 13 y 14 del GPIOG, que es una división de los pines del General Purpose Input/Output, haciendo que los leds de usuario parpadeen.

b. Configurando el Sistema de Datos FAT

Para configurar este sistema, primero es clave sepamos qué es. FAT es el acrónimo de "File Allocation Table" - tabla de localización de archivos, en inglés. Y es simplemente eso. Una especie de índice, que tu sistema operativo utiliza para guardar la localización real (en el disco duro) de cada archivo individual.

El sistema utiliza un área del disco (que no está disponible para el usuario) para guardar este índice. Es un área restringida, que NO puedes usar para tus archivos. Este es el motivo por el que muchos discos duros tienen menos estacio disponible que su capacidad real teórica.

El sistema FAT almacena la posición concreta del comienzo de cada archivo (y

esto significa sector, cilindro y disco, si hay varios) en el disco duro.

Ahora que se tiene la noción de qué es el sistema FAT procedemos a configurarlo. En el caso de nuestro microcontrolador, se utiliza el puerto USB en lugar de un MicroSD, simplemente porque es más rápido y sencillo de configurar y no es necesario anclar un componente externo a la placa como si se tuviera que soldar un puerto MicroSD.

El puerto USB en el microcontrolador es el CN6, el mismo permite el usuario conecte un componente, que en nuestro caso es un cable que permita se inserte una memoria USB en la que se crearán y sobrescribirán archivos. Para ello dentro de los componentes de software es clave se active la base del sistema de archivos, marcando el siguiente cuadrante:

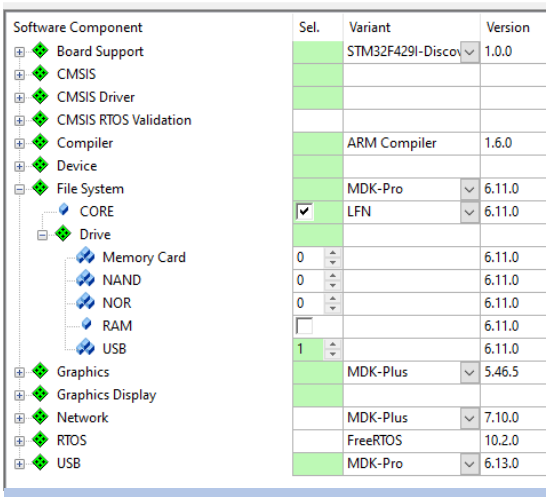

Figura 4. *Configurar FAT.*

Como se observa, se activó el sistema de archivos, habilitando a su vez el puerto USB. El código para el mismo es el siguiente:

```
  usb_status                    =
USBH_Initialize(0U);

// Initialize USB Host 0

     char   text[]   =   "Cámara
Digital,      Laura      Rosas
Morandeira,         Universidad
Latina";


  if (usb_status != usbOK) {

    for         (;;)         {}
//  Handle  USB  Host  0  init
failure

  }


  for (;;) {

    msc_status                  =
USBH_MSC_DriveGetMediaStatus("
U0:");

// Get MSC device status

    if      (msc_status      ==
USBH_MSC_OK) {

     osThreadNew(GUIThread,
NULL, &GUIThread_attr);

     HAL_GPIO_TogglePin(GPIOG,
GPIO_PIN_13|GPIO_PIN_14);


if (con == 0U) {              //
If  stick  was  not  connected
previously

        con        =        1U;
// Stick got connected

        msc_status             =
USBH_MSC_DriveMount("U0:");

        if    (msc_status    !=
USBH_MSC_OK) {
```

```
          continue;
// Handle U0: mount failure
        }
        f  =  fopen("Test.txt",
"w");           // Open/create
file for writing


        if (f == NULL) {
          continue;
// Handle file opening/creation
failure
        }
        fprintf(f, "%s", text);
        fclose (f);
// Close file
        msc_status           =
USBH_MSC_DriveUnmount("U0:")


        if    (msc_status    !=
USBH_MSC_OK) {
          continue;
// Handle U0: dismount failure
        }
      }
    } else {
      if   (con   ==   1U)   {
//  If  stick  was  connected
previously
        con        =        0U;
// Stick got disconnected
      }
    }
    osDelay(100U);
  }
}
```

Este código básicamente verifica constantemente si hay una memoria USB conectada al microprocesador y de ser así, lo notifica encendiendo los dos leds de usuario. Al hacer esto crea un archivo de texto llamado “Text.txt” y en él escribe:

“Cámara Digital, Laura Rosas Morandeira, Universidad Latina”

Luego lo guarda y cuando revisamos la memoria USB en un ordenador se puede observar el archivo.

c. Implementación del GUI

La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una maquina o computador.

En nuestro microcontrolador, el GUI es activado mediante las librerías de los componentes del mismo, de la siguiente manera:

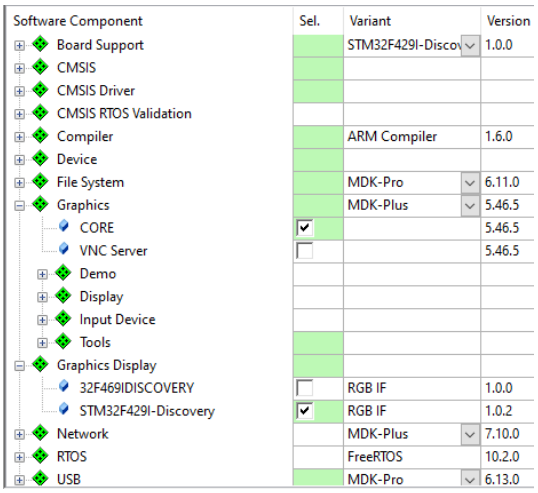

Figura 5. *Configurar GUI.*

Aquí se activa la interfaz gráfica en RGB del STM32F429i-Discovery. Configurándose con el siguiente código:

```
extern                    WM_HWIN
CreateImageViewerDialog (void);

#define GUI_THREAD_STK_SZ
      (2048U)

static void
       GUIThread          (void
*argument);
        /* thread function */

static              osThreadId_t
GUIThread_tid;

                  /* thread id */

static uint64_t
GUIThread_stk[GUI_THREAD_STK_S
Z/8];  /* thread stack */


static   const   osThreadAttr_t
GUIThread_attr = {

      .stack_mem =
&GUIThread_stk[0],

      .stack_size               =
sizeof(GUIThread_stk),

      .priority   =
osPriorityIdle

};


int Init_GUIThread (void) {

      GUIThread_tid             =
osThreadNew(GUIThread,    NULL,
&GUIThread_attr);

      if (GUIThread_tid == NULL)
{

            return(-1);

      }

      return(0);

}


__NO_RETURN     static     void
GUIThread (void *argument) {

      (void)argument;

      GUI_Init();
                  /* Initialize
the Graphics Component */

      /* Add GUI setup code here
*/

      CreateImageViewerDialog()
;

      while (1) {

            /*  All  GUI  related
activities might only be called
from here */
```

```
            GUI_TOUCH_Exec();

            GUI_Exec();
                  /*    Execute
all GUI jobs ... Return 0 if
nothing was done. */

            GUI_X_ExecIdle();
       /* Nothing left to do for
the moment ... Idle processing
*/

      }

}
```

Este código es usado junto con la herramienta llamada GUIBuilder, mediante la cual se creó una interfaz, como la siguiente:

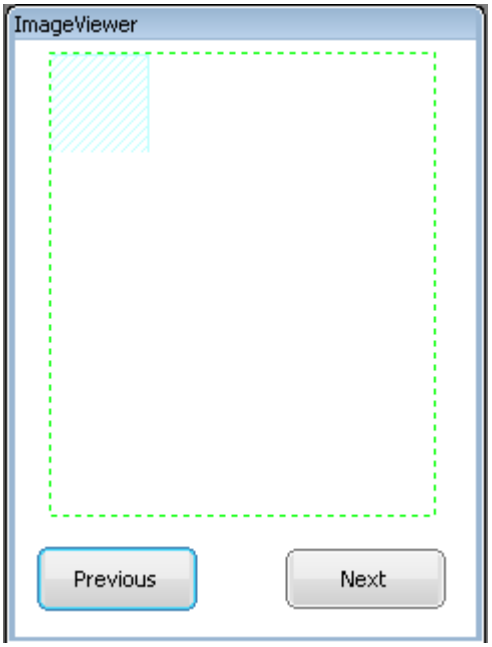

Figura 6. *Interfaz de ImageViewer.*

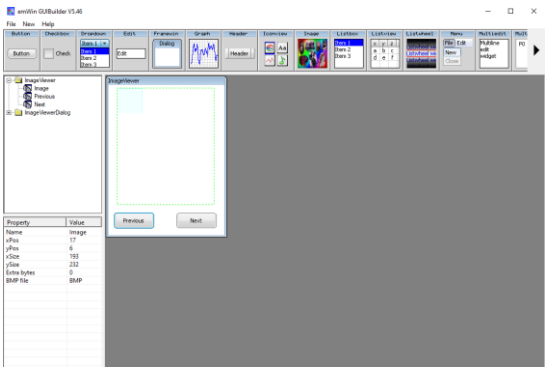

Figura 7. *GUIBuilder.*

Creando esta interfaz, se genera un archivo de tipo c, el cual es luego utilizado para generar el código que permite cambiar de imagen.

Para este caso en particular, debido a la memoria del microcontrolador y la falta de RAM en la computadora en que se desarrolló el código, se utilizan 3 imágenes como prueba y se utiliza el siguiente código:

```
switch (pMsg->MsgId) {

  case WM_INIT_DIALOG:

    hItem                          =
WM_GetDialogItem(pMsg->hWin,
ID_IMAGE_0);

    pData                          =
_GetImageById(ID_IMAGE_0_IMAGE
_0, &FileSize);

    IMAGE_SetBMP(hItem,  pData,
FileSize);

    break;

  case WM_NOTIFY_PARENT:

    Id        =  WM_GetId(pMsg-
>hWinSrc);

    NCode = pMsg->Data.v;

    switch(Id) {
```

```
    case ID_BUTTON_0: // Notifications sent by 'Previous'
      switch(NCode) {
      case WM_NOTIFICATION_CLICKED;
        break;
      case WM_NOTIFICATION_RELEASED:
                  selectedImg = selectedImg - 1;
                      if (selectedImg < 0){
      selectedImg = 2;
                            }
                     hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);
      switch(selectedImg) {
       case 0:
                  pData = _GetImageById(ID_IMAGE_0_IMAGE_0, &FileSize);
       break;
       case 1:
                  pData = _GetImageById(ID_IMAGE_1_IMAGE_0, &FileSize);
       break;
       case 2:
                        pData = _GetImageById(ID_IMAGE_2_IMAGE_0, &FileSize);
       break;
                              }
           IMAGE_SetBMP(hItem, pData, FileSize);
        // USER END
        break;
      break;
    case ID_BUTTON_1: // Notifications sent by 'Next'
      switch(NCode) {
      case WM_NOTIFICATION_CLICKED:
        break;
      case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
                  selectedImg = selectedImg + 1;
                  if (selectedImg > 2){
      selectedImg = 2;
```

```
                    }
                    hItem = WM_GetDialogItem(pMsg->hWin, ID_IMAGE_0);
      switch(selectedImg) {
       case 0:
      pData = _GetImageById(ID_IMAGE_0_IMAGE_0, &FileSize);
       break;
       case 1:
                    pData = _GetImageById(ID_IMAGE_1_IMAGE_0, &FileSize);
       break;
       case 2:
                    pData = _GetImageById(ID_IMAGE_2_IMAGE_0, &FileSize);
       break;
                    }
      IMAGE_SetBMP(hItem, pData, FileSize);
        // USER END
        break;
       }
      break;
        }
        break;
      default:
        WM_DefaultProc(pMsg);
        break;
    }
}
```

d. Configuración del Touchscreen

Luego de esto, lo único que queda es activar el táctil de la pantalla, para ello se configura el Run-Time Environment de la siguiente manera:

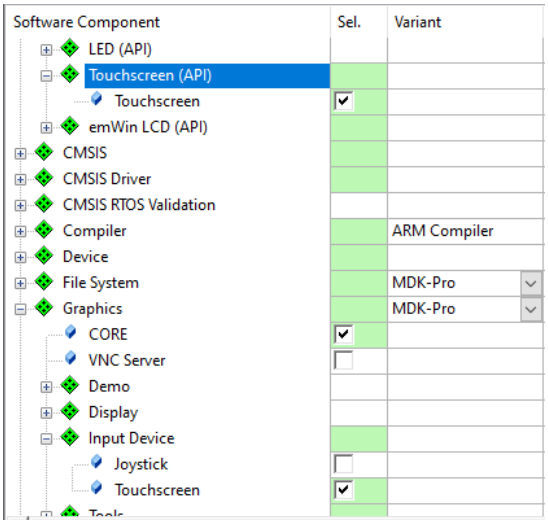

Figura 8. *Configurar Touchscreen.*

Y se ajusta el archivo RTE_Device.h, de la siguiente manera:

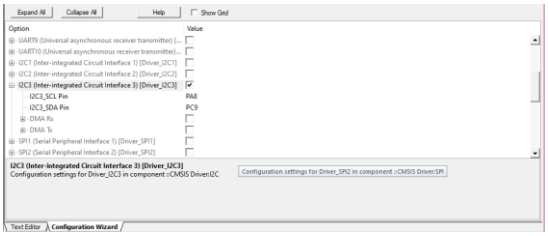

Figura 9. *Editar RTE_Device.h*

Esto ya que, estos puertos son los designados para el touchscreen.

e. Resultados

El proyecto se vería de la siguiente manera:

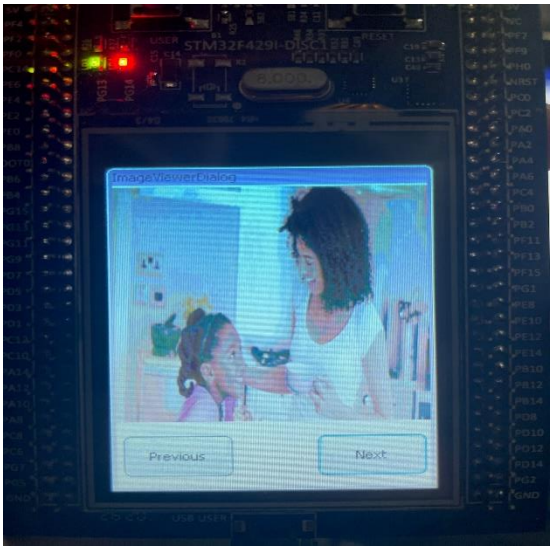

Figura 10. *Proyecto Final.*

IV. CONCLUSIÓN

Mediante este proyecto se concluye que los microprocesadores al ser programados adecuadamente pueden ser utilizados para múltiples propósitos que pueden facilitar la vida del usuario, proporcionando nuevos recursos interesantes de utilizar, como es el caso del photobooth o el carrete de imágenes que puede ser mejorado e implementado para acceder a memorias y visualizar archivos de las mismas.

Esto demuestra los microcontroladores son una herramienta sumamente útil con grandes utilidades.

V. BIBLIOGRAFÍA

[1] A. García, «¿Qué es el formato FAT?,» 21 febrero 2012. [En línea]. Available: http://albertog.over-blog.es/article-que-es-el-formato-fat-100123447.html. [Último acceso: 16 abril 2020].

[2] Sistemas, «Definición de GUI,» 2009. [En línea]. Available: https://sistemas.com/gui.php. [Último acceso: 16 abril 2020].

[3] J. Martínez, «GUI,» 2009. [En línea]. Available: http://dis.um.es/~lopezquesada/documentos/IES_1415/LMSGI/curso/xhtml/html12/html/definicion.html. [Último acceso: 16 abril 2020].

[4] ElectrónicaEstudio.com, «¿Qué es un microcontrolador?,» 2018. [En línea]. Available: https://www.electronicaestudio.com/que-es-un-microcontrolador/. [Último acceso: 16 abril 2020].

[5] ST, «32F429IDISCOVERY,» 2020. [En línea]. Available: https://www.st.com/en/evaluation-tools/32f429idiscovery.html. [Último acceso: 16 abril 2020].