Proyecto Final Adquisición de datos

Juan David Sandoval

Juandsandoval8@gmail.com, Ingeniería Electrónica

Resumen—*Las tarjetas de desarrollo son muy usadas hoy día para adquirir señales de sensores externos.*

Abstract*— *Development boards are widely used today to acquire signals from external sensors.

1. INTRODUCCIÓN

Este informe presentará el proyecto final definido como parte final del curso adquisición de datos, en este caso se usará la tarjeta de desarrollo STMF411RE.

2. METODOLOGÍA

A. PROBLEMA

Se ha planteado desde el comienzo del curso de adquisición de datos que el proyecto será una interfaz gráfica en Matlab donde podamos usar mando remoto por comunicación serial, descomposición de colores RGB mediante video y toma recurrente de fotos además de un mando físico con joystick.

B. MATERIALES

*STM32F411RE
*Modulo joystick
*Servomotores
*Matlab

C. COMO USARLOS

- STM32F411RE: Al igual que en otros informes y guías se usará "Mbed Compiler" para programar la tarjeta STM, una vez creado el proyecto se trabajarà el ADC y la comunicación serial todo junto en un solo código.

```
#include "mbed.h"
Serial pc(SERIAL_TX,SERIAL_RX);
AnalogIn Potx(A0);
AnalogIn Poty(A1);
PwmOut servo1(PC_9);
PwmOut servo2(PC_8);
```

Fig.1

- El primer paso será declarar la comunicación serial y las entradas análogas que se quieren convertir, éstas corresponden a las salidas de los dos potenciómetros del módulo joystick.

- Lo siguiente será declarar la salida las salidas del tipo PWM(ancho de pulso o pulse width) , que serán las salidas hacia los servomotores.

- Donde X-axis y Y-axis serán las entradas A0 y

```
int main()
{
    servo1.period_ms(20);
    servo1.pulsewidth_us(1000);
    servo2.period_ms(20);
    servo2.pulsewidth_us(1000);


    uint16_t posx,posy;
    int pwm1,pwm2 = 1750;
    int a = 1000;
    int b = 1000;
```

Fig.2

- En el código principal vendrán las variables de inicio de cada servomotor, estas variables (period y pulsewidth), son tomadas de la ficha técnica del servomotor; y las variables a y b que se utilizaràn para el movimento desde el mando remoto.

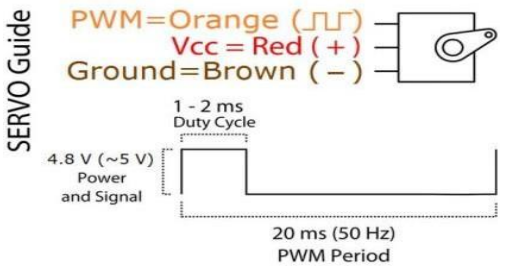

Fig.3

- El voltaje de alimentación de los servos se hará de la siguiente manera:
  Si se conectan desde la tarjeta debe hacerse a 3.3v y sí se hace externamente se debe hecer a 5v.
- El duty cycle de los servos utilizados (sg90) pueden ir de 0 a 2.5mS.

- También se tienen dos variables “posx” y “posy”, estas variables serán el almacenamiento d ela conversión análoga-digital de los dos potenciómetros conectados en los puertos A0 y A1y estarán guardadas en variables del tipo uint_16t.

-Hay dos variables del tipo entero “pwm1” y “pwm2”, que representan los datos a recibir del puerto serial.

```
while(1)
{

    posx=Potx.read_u16();
    posy=Poty.read_u16();


    pc.printf("%d,%d\n",posx, posy);
    wait_ms(20);
    if (pc.readable()==0)
    {
        pc.scanf("%d",&pwm1);
        pc.scanf("%d",&pwm2);
        wait_ms(20);
       }
       a+=pwm1;
       b+=pwm2;
        servo1.pulsewidth_us(a);
        servo2.pulsewidth_us(b);
    }

}
```

Fig.4

- Dentro del ciclo infinito se tiene la conversión análoga a digital, como este dato se recibe de 0 a 65535. Lo que se hará es tratar esa conversión para lograr esos valores y guardarlos en las variables del tipo int que se mencionaron anteriormente (pwm1 y pwm2).
*posx=Potx.read_u16();
*posy=Poty.read_u16();

-Una vez hecho esto se procederá a enviar los datos correspondientes mediante un "printf" en este caso posx y posy como datos
enteros y un salto de línea, seguidamente un tiempo de espera para recibir el dato.

- Al recibir datos desde Matlab la tarjeta generar un espacio en blanco para esto haremos una condición if seguida de una instrucción como lo es readable()==0, que lo que hará es decir que si no hay nada leyendo pase, haga salto de línea, y siga leyendo los datos provenientes del puerto serial.

- Se recibirán ahora si los datos con un "scanf" donde recibiremos las variables anteriormente mencionadas pwm1 y pwm2; estos datos se van a recibir de forma separada para no generar inconvenientes al momento de recibir los datos por el puerto serial también recibiremos datos del tipo entero, como adicional usaremos el comando & lo que hará es un puntero para que la tarjeta sepa que variable es la que proviene de Matlab, después daremos una espera de 20mS.

- Por ultimo será tener las variables a y b que tienen un valor de 1000 y las sumaremos con lo que proviene del puerto serial, esta variable será la que vamos a enviar a las salidas pwm.

- En informes anteriores se explicó como recibir y enviar datos desde el puerto serial mediante Matlab.

Ya se explicará cómo y qué datos se enviará desde Matlab para el mando remoto.

*** Matlab y GUI**

La comunicación serial, y el uso del GUI ya fueron implementados con éxito en antiguos proyectos. Lo siguiente es hacer la adquisición desde una cámara web, tomar una foto y posteriormente hacer su debida descomposición, solo mostrando el color requerido.

-Descomposición RGB
Usando los conocimientos aprendidos en los cursos anteriores sobre resolución de matrices, se procederá a descomponer una imagen en sus tres colores principales capas (RGB).

```
%Juan Sandoval 63078
vid=videoinput('winvideo',1,'YUY2_640x480','ReturnedColorSpace','rgb');
a=getsnapshot(vid);
figure
imshow(a)
b = rgb2gray(a);
R = double(a(:,:,1));
V = double(a(:,:,2));
A = double(a(:,:,3));
f1 = A - V - A;
bin = f1 > 25;
figure
imshow(bin)
filtro = medfilt2(bin);
figure
imshow(filtro)
mascara= 1 - filtro;
figure
imshow(mascara)
AZUL = double(b)/255;
VERDE = double(b).*mascara/255;
ROJO = double(b).*mascara/255;
final = cat(3,ROJO,VERDE,AZUL);
figure
imshow(final)
clc;
clear;
```

Fig.5.

- Lo primero será cargar el video como ya explicó en guías anteriores.
- Lo siguiente será tomar una foto del video mediante el comando "getsnapshot", una vez hecho esto se podrá visualizar la imagen mediante un "figure" seguido de un "imshow" para ver la imagen en un recuadro diferente; la imagen que se usará en esta ocasión será la siguiente:

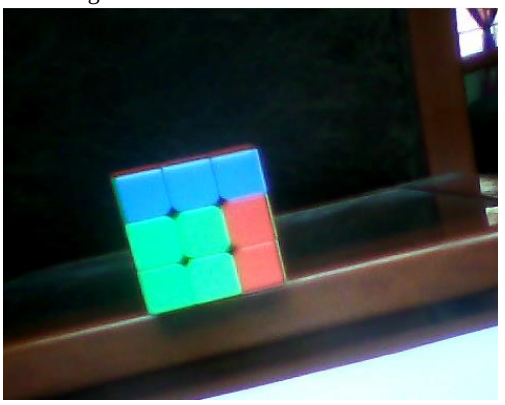

Fig.6 Cubo Rubic.

- Se usó un cubo Rubic por las facilidades para hallar colores vivos sin ser demasiado brillantes, además de su amplia gama de colores.Lo siguiente que haremos será convertir la foto tomada (fig.3);En escala de grises mediante el comando rgb2gray, comando ya usado en informes anteriores.
  Gris=rgb2gray
  (fig3);

- Luego se descompondrá la imagen original(fig.3); en sus tres capas o matrices principales como se verá a continuación.
  - ***R = double(fig6(:,:;1); % donde R es rojo***
  - ***G= double(fig6(:,:,2); % donde G es verde***
  - ***B=double(fig6(:,:,3); % donde B es azul***

- Lo siguiente será hacer una resta de colores para lograr el color deseado.
  Ej.: Para color rojo será, resta=R-G-B;
  Para color verde será, resta=G-R-B;
  Y para color azul será, resta=B-G-R;
  - Para lograr sacar el color que se necesita lo único es que debe ir al principio de la resta.

  Lo siguiente será binarizar la imagen dándole valores para la cantidad de pixeles que se quieren tomar, esto lo lograremos con una operación de mayor que:
  - binarizacion= resta > 25;
- Podemos tomar valores de 0 a 255, entre más cercano sea el valor al 0 más pixeles de ese color tomaremos y entre mas cercanos tomamos valores a 255 menos pixeles tomaremos, para este caso usaremos un valor de 25.

  Lo siguiente que se hará será aplicar un filtro de media a toda la matriz entera ya binarizada con el comando medfilt2.
  filtro = medfilt2(binarizacion);

-Esto nos ayudara con pixeles sueltos y a suavizar un poco la imagen.

- Si queremos ver el proceso de filtrado de la imagen usaremos "figure" seguido de "imshow(filtro)".

Pero se necesita una máscara de la figura proveniente de filtro, esto se logra haciendo una resta sencilla, como se verá a continuación:

- Mascara=1-filtro;

Esto dará como resultado un fondo blanco y la figura deseada en color negro.

Una vez hecho estos procesos lo que se haá es seleccionar el color a resaltar en la imagen final, este proceso se hará mediante unas operaciones aritméticas sencillas.

NOTA: Mostraremos los 3 colores al final del código.

Rojo=double(Gris)/255; % esta operación muestra el color
Verde=double(Gris). *mascara/255; % lo pone en gris
Azul=double(Gris). *mascara/255; % lo pone en gris

Por último, se tendrán tres variables resultantes que son Rojo, Verde y Azul; se debe concatenar las tres variables en una sola esto se logra mediante el comando “cat”.
Final=cat(3, Rojo,Verde,Azul);
Y para ver la imagen resultante “figure” seguido de “imshow(Final)”.

NOTA: El comando “figure” y “imshow”, solo se usará para cuando se tienen imágenes precargadas o que no se requieran mostrar en un axis.

Implementación en GUI

Como ya se vió en la descomposición de colores RGB, tenemos tres códigos uno para rojo, otro para verde y otro para azule.

- Se deben crear 3 botones para lograr esto.
Se crean tres botones uno para Rojo otro para Verde y otro para Azul.
- Pero no sin olvidar crear un botón para cargar video y ajustarlo con nuestro mando ya sea remoto o físico que ya se explicará.

- Veremos la configuración de cada botón:

- Cargar video.
-

```
84      % --- Executes on button press in Cargar_Video.
85      function Cargar_Video_Callback(hObject, eventdata, handles)
86 -    global vid;
87 -    vid=videoinput('winvideo',1,'YUY2_640x480','ReturnedColorSpace','rgb');
88      % vid.FramesPerTrigger=1;
89      % vid.ReturnedColorSpace= 'rgb';
90 -    preview(vid);
```

Fig.8

Solo se inicializa la cámara y siempre estará encendida hasta que cerremos la ventana del “preview”.

- Desc.Rojo

```
93      % --- Executes on button press in ROJO.
94      function ROJO_Callback(hObject, eventdata, handles)
95 -    global vid;
96 -    a=getsnapshot(vid);
97 -    b1 = rgb2gray(a);
98 -    R1 = double(a(:,:,1));
99 -    V1 = double(a(:,:,2));
100 -   A1 = double(a(:,:,3));
101 -   f1 = R1 - V1 - A1;
102 -   bin1 = f1 > 25;
103 -   filtro1 = medfilt2(bin1);
104 -   mascara1= 1 - filtro1;
105 -   AZUL1 = double(b1).*mascara1/255;
106 -   VERDE1 = double(b1).*mascara1/255;
107 -   ROJO1 = double(b1)/255;
108 -   final = cat(3,ROJO1,VERDE1,AZUL1);
109 -   image(final)
110 -   axis image;
```

Fig.9

Cuando se presione el botón rojo tomará una instantánea del preview y hará todo el proceso para descomponer la capa de rojos. Además, mostrará la imagen final en el axis previamente creado en el GUI.

- Desc.Verde

```
114     % --- Executes on button press in VERDE.
115     function VERDE_Callback(hObject, eventdata, handles)
116 -   global vid;
117 -   c=getsnapshot(vid);
118 -   b2 = rgb2gray(c);
119 -   R2 = double(c(:,:,1));
120 -   V2 = double(c(:,:,2));
121 -   A2 = double(c(:,:,3));
122 -   f2 = V2 - R2 - A2;
123 -   bin2 = f2 > 15;
124 -   filtro2 = medfilt2(bin2);
125 -   mascara2= 1 - filtro2;
126 -   AZUL2 = double(b2).*mascara2/255;
127 -   VERDE2 =double(b2)/255;
128 -   ROJO2 = double(b2).*mascara2/255;
129 -   final2 = cat(3,ROJO2,VERDE2,AZUL2);
130 -   image(final2);
131 -   axis image;
```

Fig.10

Realiza el mismo procedimiento del botón rojo, con la diferencia de que esta descompone solo color verde.

- Desc.Azul

```
135     % --- Executes on button press in AZUL.
136     function AZUL_Callback(hObject, eventdata, handles)
137 -   global vid;
138 -   e=getsnapshot(vid);
139 -   b3 = rgb2gray(e);
140 -   R3 = double(e(:,:,1));
141 -   V3 = double(e(:,:,2));
142 -   A3 = double(e(:,:,3));
143 -   f3 = A3 - V3 - R3;
144 -   bin3 = f3 > 25;
145 -   filtro3 = medfilt2(bin3);
146 -   mascara3= 1 - filtro3;
147 -   AZUL3 = double(b3)/255;
148 -   VERDE3 = double(b3).*mascara3/255;
149 -   ROJO3 = double(b3).*mascara3/255;
150 -   final3 = cat(3,ROJO3,VERDE3,AZUL3);
151 -   image(final3);
152 -   axis image;
```

Fig.11

Realiza el mismo procedimiento que los demás, con la diferencia que este solo descompone el azul.

EL movimiento de los servos por Comunicación Serial.

Una vez se han enviado y recibido datos del puerto serial. Lo que sigue será enviar valores para recibir mediante el mando a distancia.

```
155     % --- Executes on button press in Y_Positivos.
156     function Y_Positivos_Callback(hObject, eventdata, handles)
157 -   global puerto_serial;
158 -   global a; %a son x
159 -   global b; %b son y
160 -   a = 0;
161 -   b = 100;
162 -   fprintf(puerto_serial,'%d\n',a);
163 -   fprintf(puerto_serial,'%d\n',b);
164 -   if a<1000
165 -   a=1000;
166 -   end
167 -   if a>2500
168 -   a=2500;
169 -   end
170 -   if b>2500
171 -   b=2500;
172 -   end
173 -   if b<1000
174 -   b=1000;
175 -   end
```

Fig.12

- Acá se verá que hay dos variables llamadas a y b, donde a será X y b será Y, para este caso vamos a enviar valores positivos para Y en ese caso como nuestra tarjeta ya está inicializada desde Mbed en 1750 no enviaremos nada a la posición X(“a”), y en cambio le enviaremos un 100 a la posición en Y(“b”), enviaremos mediante “fprintf” y haremos nustro limitante de servos mediante la sentencia "if".

Ahora se verá un ejemplo para enviar valores negativos:

```
200      % --- Executes on button press in X_Negativas.
201    function X_Negativas_Callback(hObject, eventdata, handles)
202 -    global puerto_serial;
203 -    global a;
204 -    global b;
205 -    a = -100;
206 -    b = 0;
207 -    fprintf(puerto_serial,'%d\n',a);
208 -    fprintf(puerto_serial,'%d\n',b);
209 -    pause(0.01);
210 -    if a<1000
211 -    a=1000;
212 -    end
213 -    if a>2500
214 -    a=2500;
215 -    end
216 -    if b>2500
217 -    b=2500;
218 -    end
219 -    if b<1000
220 -    b=1000;
221 -    end
```

Fig.13

Para este ejemplo se envió un -100 a X(“a”) y un 0 a Y(“b”), el procedimiento es completamente el mismo al del botón anterior.

NOTA: Cada vez que se pulse cualquier botón se sumara 100 o -100 a la variable pwm1 o pwm2 según corresponda en la STM, esto hace un movimiento más detallado y en cuanto al mando joystick físico reducirá la velocidad.

Botones abrir y cerrar puerto Serial

Para hacer el código más eficiente y mejor para el usuario se crearon dos botones adicionales abrir y cerrar puerto. Su funcionamiento se verá a continuación.

```
69     function Abrir_Serial_Callback(hObject, eventdata, handles)
70 -    global puerto_serial;
71 -    delete(instrfind({'port'},{'COM4'}));
72 -    puerto_serial=serial('COM4','BaudRate',9600,'Terminator','LF');
73 -    warning('off','MATLAB:serial:fscanf:unsuccessfullRead');
74 -    fopen(puerto_serial);
75
76
77       % --- Executes on button press in Cerrar_Serial.
78     function Cerrar_Serial_Callback(hObject, eventdata, handles)
79 -    global puerto_serial;
80 -    fclose(puerto_serial);
81 -    delete(puerto_serial);
```

Fig.14

- Como se ve son dos botones diferentes con los códigos ya usados anteriormente “fopen” para abrir el puerto serial en el botón encender y “fclose” para cerrar o apagarlo.

- Adicionalmente el “delete” para eliminar residuos que hayan quedado en el puerto serial.

NOTA: estos botones solo sirven para activar y desactivar el mando remoto y el mando físico.

Botón STM

Uno de los problemas más frecuentes durante la realización del proyecto final, fue el cómo activar o desactivar el mando físico, este error se solucionó mediante un “toggle botton” donde sí se presiona se quedará en un 1 lógico y hasta que se presione de nuevo volver a ser un 0 lógico, cuando se activa el botón encender, por defecto se activa el mando remoto y para entrar al mando físico se usará la opción de este botón tipo “toggle”, pues cuando se presiona activará el mando físico y será así hasta que se presione de nuevo y nos de la opción de volver a usar el mando a distancia.

A continuación, el código del botón STM usado:

```
247    function STM_Callback(hObject, eventdata, handles)
248 -    global puerto_serial;
249 -    global Y1;
250 -    global Y2;
251 -    global pwm1;
252 -    global pwm2;
253 -    pwm1=1750;
254 -    pwm2=1750;
255 -    while(1)
256 -    fprintf(puerto_serial,'%d\n',pwm1);
257 -    fprintf(puerto_serial,'%d\n',pwm2);
258 -    a=fscanf(puerto_serial,'%d,%d\n');
259 -    pause(0.01);
260 -    Y1=a(1);
261 -    Y2=a(2);
262 -    pwm1=1000+(Y1/26.41);
263 -    pwm2=1000+(Y2/26.41);
264 -    if pwm1<1000
265 -    pwm1=1000;
266 -    end
267 -    if pwm1>2500
268 -    pwm1=2500;
269 -    end
270 -    if pwm2>2500
271 -    pwm2=2500;
272 -    end
273 -    if pwm2<1000
```

Fig.15

- Se observa como se han inicializado las variables pwm1 y pwm2 a 1750 por facilidad de uso de los códigos.
- Dentro del ciclo infinito lo primero que se hace es enviar un dato en ese caso pwm1 y pwm2 y luego recibimos por “fscanf” y se guardará en una variable, y luego se da una pausa para poder realizar estos procesos de forma adecuada
- En este caso tenemos a=” fscanf”, donde a es un vector que contiene dos datos independientes que pueden ser divididos, como a(1) y a(2) y se guardaran en dos variables nuevas Y1 e Y2 que se actulizan cada vez que llegue un dato nuevo proveniente de la tarjeta STM.
- El siguiente procedimiento será convertir o tratar estos datos que viene de 0-65535 a datos entre 1000 y 2500, esto se logra hacer con una regla de tres simple, como se verá a continuación:

  - 65535/2500=26.41
    Esto nos asegura que si dividimos 65535 entre 26.41 tendremos un valor máximo de 2500, entonces haremos los siguiente, se requiere un valor mínimo de 1000 y máximo de 2500, entonces:
    Pwm1=1000+(Y1/26.41);
    Pwm2=1000+(Y2/26.41);

- Pwm1 y pwm2 serán los valores que enviaremos de vuelta a la tarjeta como grados que serán leídos y enviados mediante el comando “fprintf” a las salidas pwm de nuestra tarjeta STMF411RE.
- Seguido de esto se usará los condicionales limitantes para que nuestros servos no se pasen de los valores establecidos y dañarlos.

Por último, se verá la descomposición de colores en cada capa:

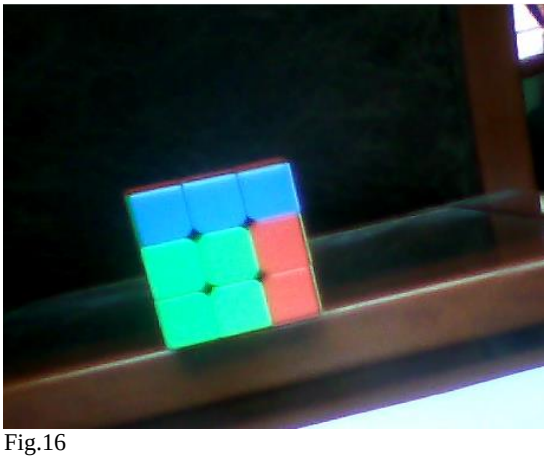

Fig.16
Esta será la foto original tomada del “preview”, ésta imagen se verá en el axis del GUI de Matlab.

- Visualización de color rojo.

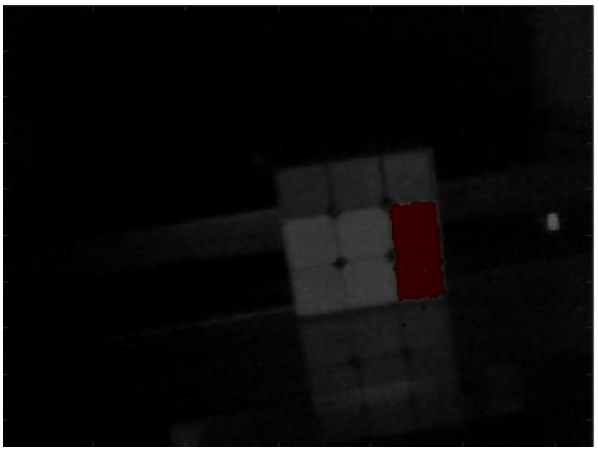

Fig.17

- Una vez tomada la foto del “preview” se usará el botón Desc.Rojo y ahora lo que se verá es esta imagen que nos ha puesto solo lo que tenía color rojo y lo demás en escala de grises.

- Visualización de color Verde.

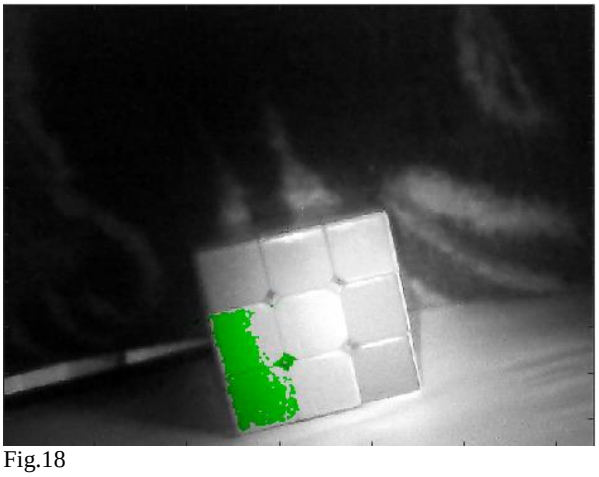

Fig.18

-Se observa que ha tomado una gran cantidad de color verde y lo demás que debería tener color verde lo ha puesto es escala de grises, esto se debe a que hay demasiada luz inducida en ciertas zonas del objeto. **NOTA:** Al inducirle demasiada luz al objeto a tratar, Matlab puede tomar como si fuese blanco mostrando solo gris la imagen tratada.

- Visualización de color azul

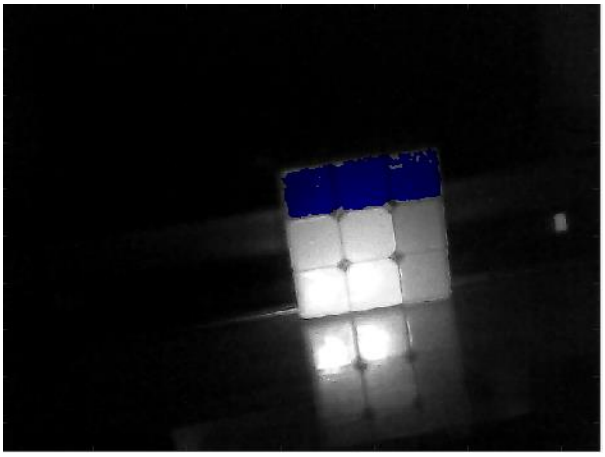

Fig.19

- Por ultimo se obervará la descomposición en azul usando su botón correspondiente.

Pre visualización GUI Matlab y botones

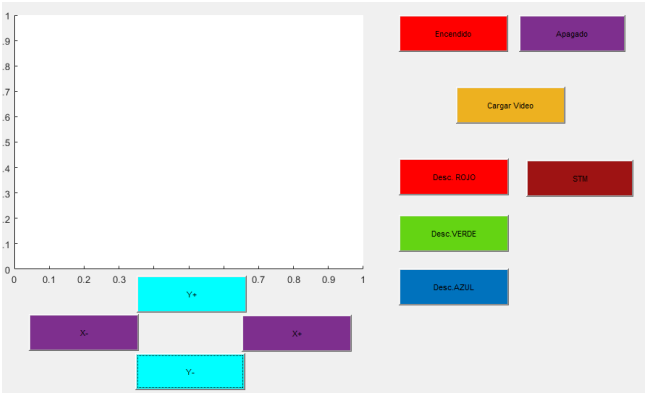

Fig.20

- En la imagen se observar un espacio en blanco que es el axis donde se verán las imágenes tratadas.
- Botón encendido: Abre puerto serial, parte superior de color rojo.
- Botón apagado: Cierra el puerto serial, parte superior de color morado.
- Botón STM: Es el botón tipo “toggle” que da el cambio entre el mando físico y el mando remoto, botón de la derecha color café.
- Desc.Rojo: Toma una foto del “preview” y descompone la imagen como se ve en la figura 17.
- Desc.verde: Toma una foto del “preview” y descompone la imagen como se ve en la figura 18.
- Desc.Azul: Toma una foto del “preview” y descompone la imagen como se ve en la figura 19.
- Botones de mando remoto: son los botones azules claro y morados debajo del axis, usados para las figuras 12 y 13.

Resultados y Conclusiones

Se pudo observar el software Matlab y sus diferentes librerías, como el “computer visión toolbox”, el “image adquisition toolbox” y otros más que nos permiten hacer este tipo de trabajos, que requieren casi de visión artificial, además se aprendió de programación para STM en “MBED Compiler”, como opinión propia para poder realizar este tipo de proyectos se requiere de computadoras no tan viejas o en buen estado ya que requiere mucha memoria ram, por otro lado se cumplió con el objetivo que era sacar el proyecto del curso.

Bibliografías

- *[1]"MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink", La.mathworks.com, 2020. [Online]. Available: https://la.mathworks.com. [Accessed: 20- May- 2020].*

- *[1]"Hackaprende", YouTube, 2020. [Online]. Available: https://www.youtube.com/channel/UCHEDLiyFAwg21STbVtePjeQ/feed?reload=9&disable_polymer=1. [Accessed: 20- May- 2020].*

- *[1]H. Moore, MATLAB para ingenieros.*