TRANSMISIÓN DE IMÁGENES DE MATLAB A ARDUINO VÍA PUERTO SERIA

Gabriel Cubas Perfecto,
gabrielcubain@hotmail.com
M. en C. Rafael Santiago Godoy
rsantiagog@ipn.mx
Álvaro Anzueto Ríos,
anzuetor aanzuetor@ipn.mx

Laboratorio de Biomecánica,UPIITA-IPN

RESUMEN

En este trabajo se da una visión general sobre la comunicación serial, sus principales parámetros así como su aplicación dentro del entorno de programación Matlab, se da una descripción de los comandos principales con los que contamos para establecer la conexión. Se explica el proceso mediante el cual se mandan los datos de una imagen en escala de grises de 24x25 pixeles a través de Matlab para su posterior envío en una placa Arduino Uno, para este proceso se muestran los comandos necesarios en el software de Arduino, su implementación y el almacenamiento de los datos en el microcontrolador que a pesar de tener poca capacidad de memoria SRAM para guardar imágenes de mayores dimensiones se ejemplifica bien el procesamiento de imágenes en sistemas embebidos. Finalmente, se envían los datos del Arduino al programa de Matlab, con la finalidad de comprobar que durante la transmisión no exista ruido o pérdida de la información, con lo cual se llega a la conclusión de que se puede realizar el procesamiento de imágenes de reducidas dimensiones dentro de un microcontrolador obteniendo los datos mediante algún periférico de forma fácil y eficiente.

1. Introducción

Matlab es un software orientado a realizar cálculos matemáticos basándose principalmente en operaciones matriciales. Por tanto, será más eficiente si se diseñan los algoritmos en términos de matrices y vectores, lo que hace adecuado a este programa para el procesamiento de imágenes. Arduino es una plataforma electrónica open-source, basada en un hardware y software fácil de usar, que ha ido ganando adeptos a lo largo de todo el mundo debido a que la plataforma está destinada a cualquier persona pueda realizar proyectos interactivos, sin la necesidad de ser experto en el manejo de microcontroladores, y en electrónica en general. Con estas dos herramientas se puede implementar un sistema de procesamiento de imágenes en el cual se obtengan los datos por medio de una computadora o webcam y se transmitan mediante Matlab al hardware Arduino. Primero hay que saber que la comunicación serial es un protocolo para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. Para realizar la comunicación se utilizan 3 líneas de transmisión: Tierra (o referencia), Transmitir y Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. A continuación se describen los parámetros más importantes de la comunicación serial.

a. Velocidad de transmisión: Indica el número de bits por segundo que se transfieren, y se mide en baudios.

b. Bits de datos: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits.

c. Bits de parada: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits.

d. Paridad: Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible.

Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

2. Desarrollo

Para el desarrollo de este trabajo se utilizó una laptop Asus con AMD A4, Windows 7 y la versión Matlab R2012b, asimismo se usó la placa Arduino Uno y la versión 1.6.5 de su software. Primero, para cargar una imagen en Matlab se emplea el comando imread, especificando el nombre de la imagen, la dirección donde se ubica (solamente si se encuentra en otra carpeta) así como su formato (.jpg, .png, .tif, etc.)

```
>> imagen=imread('cuarto_0.jpg');
```

Las imágenes en formato RGB se representan con tres matrices que contienen el valor para los colores rojo, verde y azul, por lo tanto, para facilitar su procesamiento, la imagen se convierte a escala de grises, esto se hace especialmente cuando se quieren detectar bordes, figuras geométricas, extracción de líneas, entre otros.

La siguiente línea de código muestra el comando para convertir a escala de grises la imagen anteriormente cargada.

```
>> img=rgb2gray(imagen);
```

Asimismo, para enviar los datos de la imagen a otro dispositivo es necesario conocer las dimensiones de la matriz que representa a la imagen, lo que se hace con la siguiente línea de código

```
>> [m,n]=size(img);
```

Donde m recibe el número de filas de la imagen y n el número de columnas Transmisión de datos y recepción de datos Para mandar los datos de la imagen por medio de comunicación serial, se especifican parámetros tales como el puerto, la velocidad de transmisión, la paridad, entre otros. En Matlab, una forma de establecer estos parámetros es la siguiente:

```
>> delete(instrfind({'Port'},{'COM3'}))
>> puerto= serial('COM3','BaudRate',9600,'Terminator','LF');
>> warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
```

Donde, primero se limpia el puerto donde está conectada la placa Arduino, luego se establece los baudios y el tipo de fin de dato, los parámetros más comunes están definidos por defecto, así que no es necesario definirlos, pero si es requerido, consultar la página de Mathworks para mayor información.

Si ocurre algún error al inicializar el puerto se desplegará un mensaje de advertencia en el commmad window de Matlab. Para mandar los datos deseados, se abre el puerto con el siguiente comando

```
>> fopen(puerto);
```

Posterior a esto, se manda un dato cualquiera para iniciar la transmisión, esto es requerido en algunas computadoras para que la conexión sea exitosa, por lo que puede no ser necesario y pasar directamente a la transmisión de la imagen.

```
¿p() * 1.0000@
```

```
>> dummy_byte=0;
>> fwrite(puerto,dummy_byte,'uint8')
>> pause(1)
```

Después se van leyendo uno por uno los datos de la imagen en escala de grises y se van mandando a través del puerto serial. Es importante especificar el formato del dato a enviar (uint8: Enteros de 8 bits en el rango de [0,255]), para que pueda ser leído sin problemas en la placa Arduino. Esta placa cuenta con un único puerto serial el cual se puede utilizar solamente conectando el Arduino al PC con el cable USB, sin necesidad de ninguna otra conexión.

Para la recepción de los datos se realiza un programa en el software de Arduino, se deben especificar los siguientes parámetros: Tipo de dato a recibir: byte, char, int, float
Tamaño de la matriz: array [filas][columnas]
Velocidad de transmisión: Serial.begin (baudios);

Una vez establecidos los parámetros, se procede a escribir el siguiente código para comprobar si hay datos en el puerto serial, si es así, leer el dato recibido y almacenarlo en la matriz declarada anteriormente.

¿p() * 1.0000@

```
if (Serial.available()>0){
Dato = Serial.read(); array[i][j]=Dato;
if (j>=n){j=0; i++;}
j++; data++;
}
```

Para asegurarnos de solo recibir los datos enviados, colocamos un contador que se incremente cada vez que se lee un dato en el puerto, de esta forma al llegar al número total de datos enviados se dejará de leer el puerto y se procederá a realizar otra rutina.

3. Comprobación de la transmisión

Para comprobar que realmente se están leyendo los datos de forma correcta una vez que se han recibido todos los datos en la placa Arduino, se procede a enviarlos de regreso a la PC, se despliegan las imágenes, tanto la enviada como la recibida, y mediante inspección visual se determina si funciona o no el proceso de transmisión.

Para enviar los datos del Arduino a la PC se lee el dato guardado en la matriz array y se envía mediante el puerto serial de la siguiente forma:

[]@l@

```
envio=array[i][j]; Serial.println(envio)
```

Para leer los datos en Matlab se usa el comando fscanf de la siguiente forma y los datos se van guardando en una matriz del mismo tamaño que la original.

dato_recibido=fscanf(puerto,'%f');

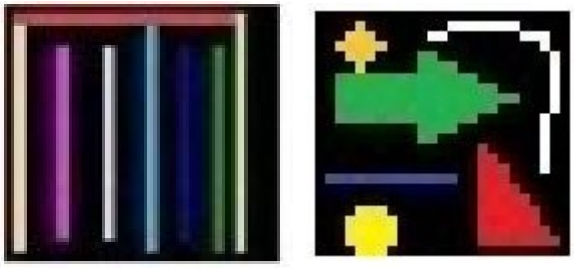
Es importante mencionar que al final del programa se debe cerrar el puerto serial, para que pueda ser usado correctamente por otros dispositivos.

[]@ ¿p() * 1.0000@

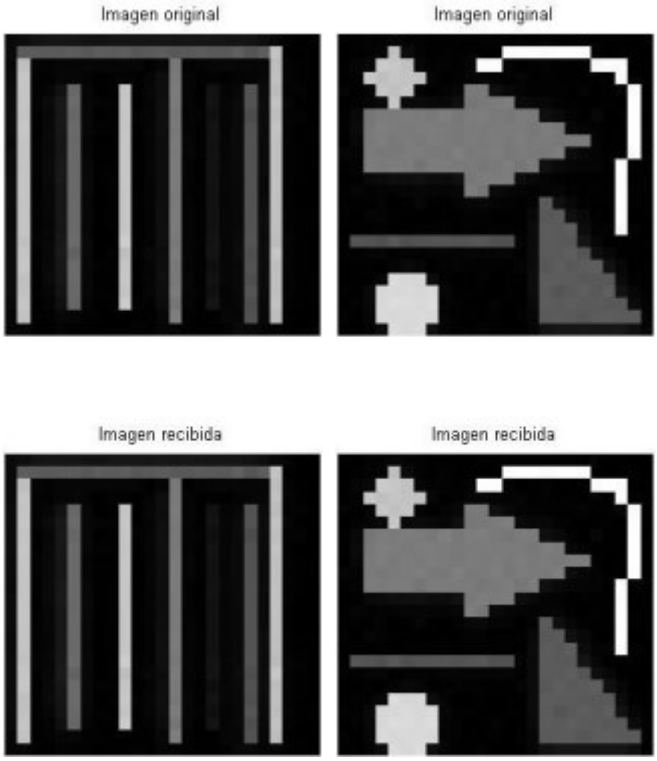
```
>> fclose(puerto); delete(puerto)
clear puerto
```

4. Resultados

Para verificar el funcionamiento de los programas realizados, se probaron con imágenes a color (Figura 1) de 24x25 pixeles, para contar con suficiente memoria SRAM disponible y se pueda ejecutar todo el programa sin errores.



Como se puede apreciar en la Figura 2, las imágenes recibidas del Arduino no presentan variaciones significativas con respecto a las originales que sean apreciables a simple viste, tampoco se observa que las imágenes estén incompletas o modificadas lo que significa que no existe discontinuidad durante la transmisión de los datos.



5. Conclusiones

La transmisión de imágenes por medio del puerto serial a otro dispositivo, en este caso la placa Arduino, se puede realizar sin presentar variaciones significativas en los datos enviados por lo que es una opción altamente recomendable para mandar este tipo de información hacia otros periféricos y realizar un procesamiento confiable en ellos.

En este programa existe la limitante de la cantidad de memoria disponible en el microcontrolador para almacenar la información de la imagen a procesar, por lo que las imágenes de prueba son de dimensiones reducidas (24x25 pixeles). Esto hace necesario tener que usar una tarjeta con mayor capacidad de memoria además de velocidad de procesamiento para poder trabajar con imágenes de mayores dimensiones, que son las que normalmente usamos a diario y de las que se pueden extraer mayor cantidad de información.

6. Referencias

Arduino. Referencia, Serial. Recuperado: Septiembre 21 de 2015 de https://www.arduino.cc/en/Reference/Serial

MathWorks. Documentación, Imágenes. Recuperado: Septiembre 22 de 2015, de http://www.mathworks.com/help/matlab/images_btfntr_-1.html

National Instruments (Junio 6, 2006). Documentos de soporte. Recuperado: Septiembre 21 de 2015 de http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1