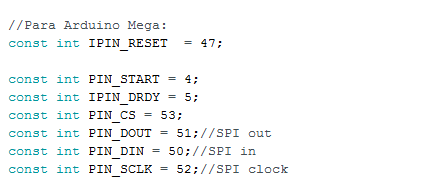
Los códigos de programación de la tarjeta arduino están basados en la librería ADS129x-tools del autor Adam Feuer de StarCat Biosignals, Seattle, WA, USA. Esta librería incluye un código llamado ads1298\_hello\_world, que sirve para revisar si existe comunicación entre el arduino y el front-end y otro llamado ads129x\_driver, con el cual se puede realizar la lectura simultanea de los ocho canales del front-end.

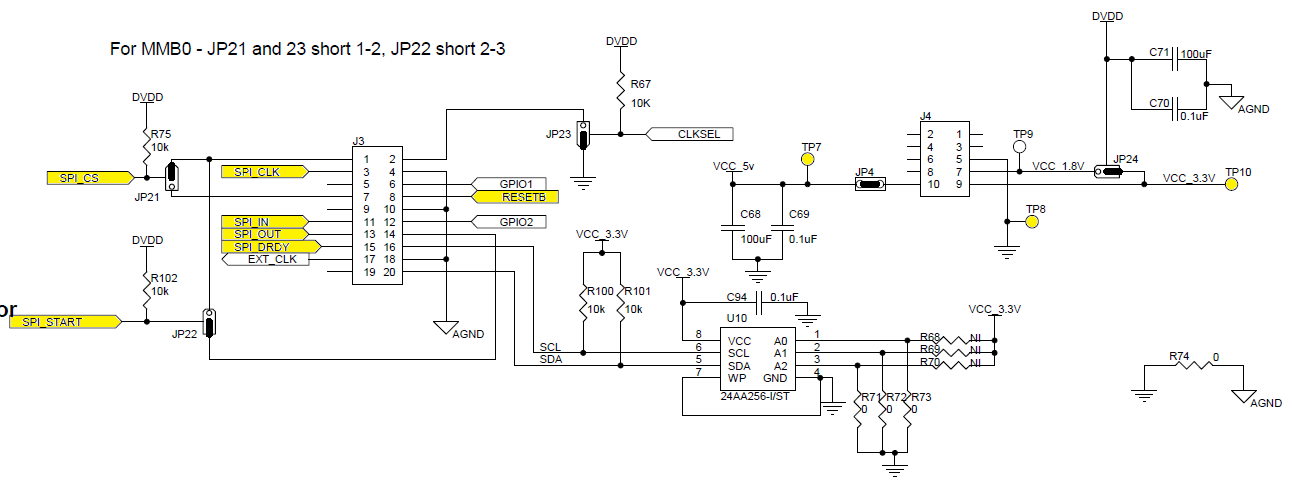
Inicialmente para realizar la comunicación entre el maestro y el esclavo (arduino y front-end respectivamente), se debe usar el programa ads1298\_ hello\_world, con el cual se realiza la lectura de información del registro ID del front-end, obteniéndose el número de canales existentes e identificación de la tarjeta. Si el programa entrega la identificación correcta, quiere decir que los dos dispositivos lograron comunicarse correctamente, permitiendo enviar comandos, leer y escribir registros.

Para conectar el arduino mega con el front-end 1298R se debe tener en consideración que se usarán 7 pines del soquet J3 para la comunicación SPI y 3 pines del soquet J4 para el suministro de energía de la tarjeta.

Configuramos la disposición de pines que serán usados en la comunicación SPI, mediante el programa adsCMD.h, tal como se muestra en el siguiente fragmento de código:



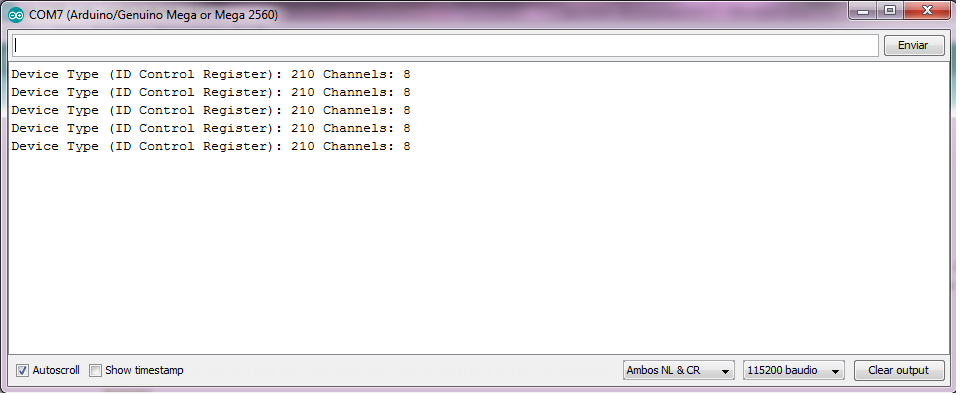
Luego guiándonos del esquema de a continuación, conectamos con cables dupont tipo macho-macho los pines destacados en amarillo del front-end 1298R al arduino.



Interpretación del esquema:

|  |  |  |
| --- | --- | --- |
| Pines Front\_end 1298R | | Pines Arduino mega |
| Soquet | N° de pin, nombre de pin | N° de pin, nombre de pin |
| J3 | 1, SPI\_CS | 53, PIN\_ CS |
| J3 | 3, SPI\_CLK | 52, PIN\_SCLK |
| J3 | 8, RESETB | 47, IPIN\_RESET |
| J3 | 11, SPI\_IN | 51, PIN\_DOUT |
| J3 | 13, SPI\_OUT | 50, PIN\_DIN |
| J3 | 14, SPI\_START | 4, PIN\_START |
| J3 | 15, SPI\_DRDY | 5, IPIN\_DRDY |
| J4 | 5, TP8 | GND |
| J4 | 9, TP10 | 3.3[V] |
| J4 | 10, TP7 | 5[V] |

Una vez teniendo todo conectado correctamente, se sube el programa a la tarjeta arduino y en el monitor serial tendría que desplegarse lo siguiente:

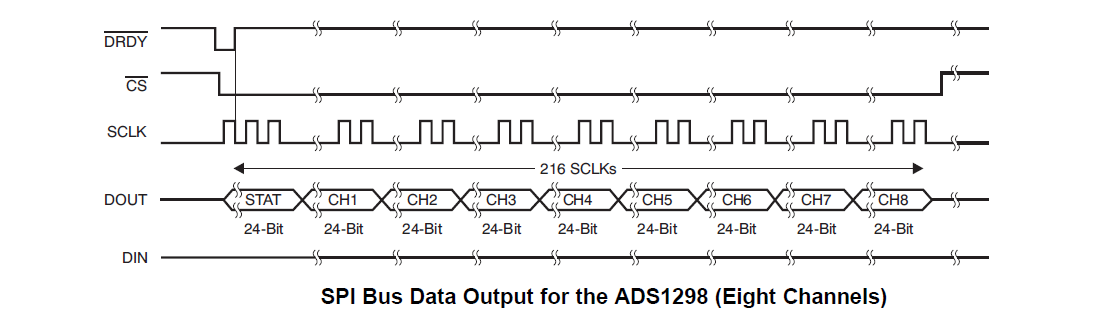


Ahora podemos comenzar a trabajar con el programa ads129x\_driver. Este programa así como el anterior, necesita que configuremos los pines de comunicación SPI, para esto debemos usar la misma definición de pines, en otras palabras, copiar el código usado en adsCMD.h al programa adsCommand.h, logrando la comunicación SPI para el programa que ahora estaremos usando.

Debido a que es necesario ir probando por partes la lectura de los canales, debemos modificar gran parte del código fuente, configurando inicialmente los ocho canales como señales de prueba (TEST\_SIGNAL) con la función void adsSetup(), en la cual una fuente de señal cuadrada interna puede generar dicha señal con una amplitud de 1 [mVpp] a frecuencia de 0 [Hz], 1 [Hz] o 2 [Hz] y la salida de esta fuente es conectada a las entradas de los canales.

Una vez realizadas las configuraciones, se deberá ejecutar una lectura continua de muestras con la función sendSamples(), esto es para saber si la señal de prueba está siendo detectada y también para visualizar la gráfica resultante desde el Serialplotter.

La conversión a digital de una muestra por cada canal es enviada al dispositivo maestro siguiendo el formato de comunicación SPI arduino – ADS1298R, recibiendo un total de 27 bytes por cada muestra realizada en los 8 canales.



Luego, se extraen los 3 bytes correspondientes a la muestra de cada canal, estas son convertidas a formato complemento dos binario, el cual para 24 bits tiene valores que rondan entre -8.388.608 y 8.388.607, el valor resultante de esta conversión será impreso por el puerto serial declarado, pudiéndose visualizar desde el monitor serial las muestras de todos los canales en tiempo real.