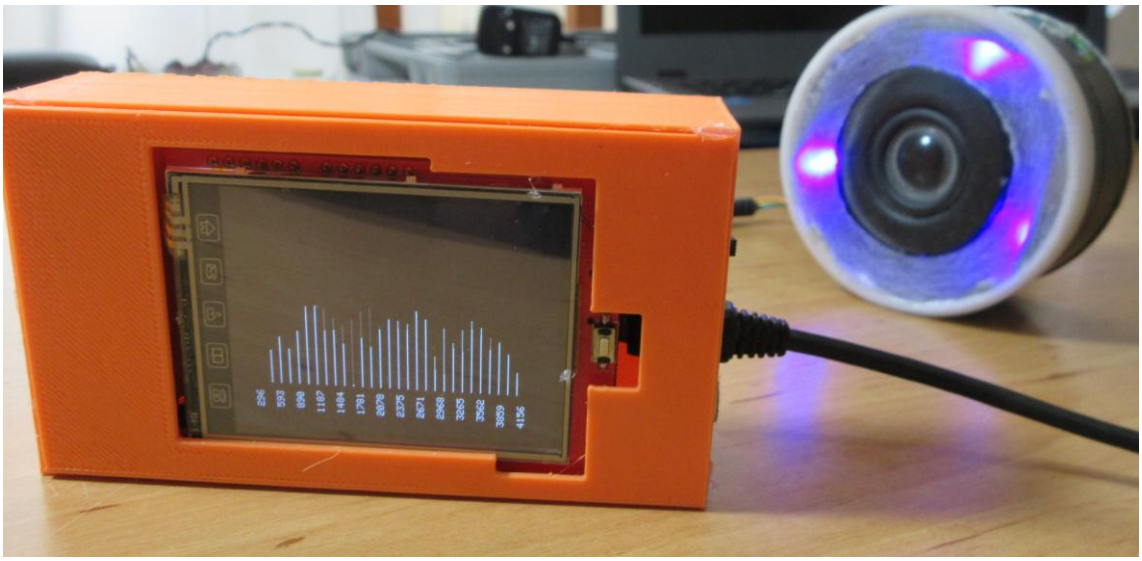
ANALIZADOR DE AUDIO CON ARDUINO UNO



Youtube: https://youtu.be/dKPURQHRQ4k

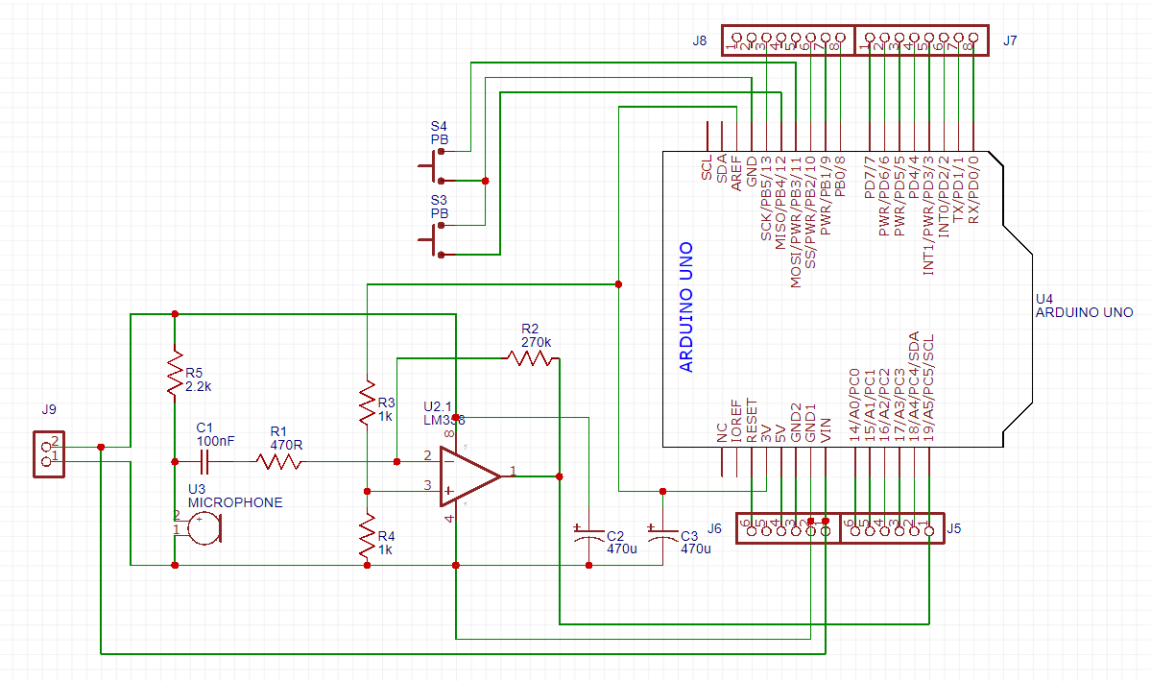
Twitter: https://twitter.com/RoboticArts1

BY ROBOTIC ARTS

1. Introducción

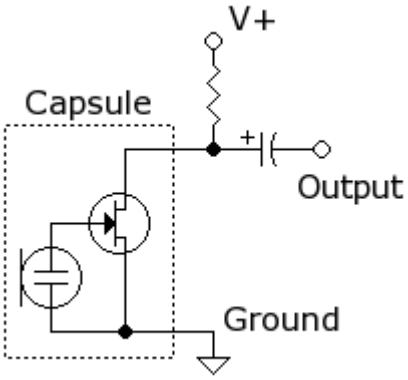
Un analizador de audio es un instrumento de medida que permite conocer varios parámetros del sonido: amplitud, RMS, ganancia logarítmica (decibelios) y frecuencia. Existen en el mercado varios analizadores de audio, llamados sonómetros o analizadores de espectro, sin embargo nosotros nos construiremos en nuestro propio. La finalidad es aprender su funcionamiento y poder analizar señales de audio de una forma humilde. De ninguna forma se pretende sustituir o equiparar a uno comercial ya que como veremos se emplearán componentes muy comunes para un aficionado a la electrónica. Los equipos comerciales están *calibrados* con equipos de *precisión* y emplean microcontroladores de más potencia con ADC dedicados.

2. Circuito



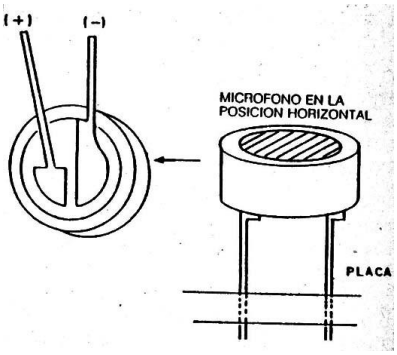
Como se pretende captar el sonido se empleará un micrófono electret. Este tipo son una variante de los micrófonos de condensador que no necesitan alimentación. Ellos con capaces de generar una tensión muy pequeña a partir

de nuestro sondo. Como esta señal es muy pequeña normalmente los electret vienen integrados con un transistor JFET y por ello es necesario emplear alimentación para polarizarlos. ¿Por qué se amplifica la señal? Pues porque la señal del electret es tan pequeña que se perdería a través de los cables que van al amplificador.

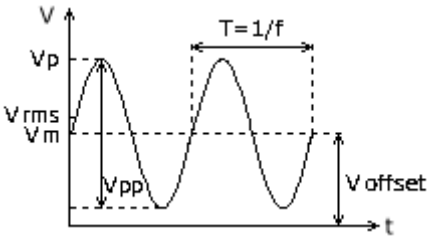


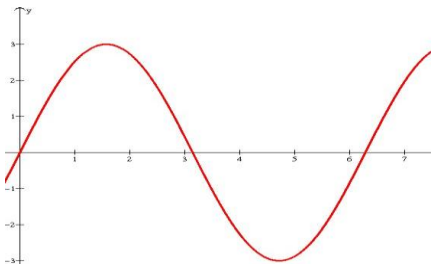
De este modo se añade una resistencia R5 de 2.2K para polarizar el micrófono electret. La resistencia R5 puede ir desde 1K hasta 10K y marcan la ganancia y saturación del micrófono. El valor de R5 no es vital ya que luego es amplificado por un operacional.

Recuerda: los micrófonos electret necesitan alimentación porque llevan integrado un JFET, por ello una cápsula electret lleva polaridad, es decir, un positivo y un negativo. Se identifica porque el negativo esta soldado a la cápsula o porque es la patilla mas corta (como con los leds).



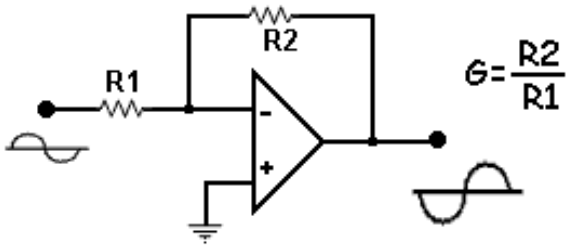
Tras el micrófono nos encontramos con un condensador de 100 nF que desacopla la componente de tensión continua generada por el micrófono. Como el micrófono lleva un transistor la salida es una señal que sube y baja entorno a una tensión, esa tensión se llama offset. El condensador elimina ese offset para que la señal suba y baje en los 0 voltios.





Después nos encontramos con un amplificador operacional. Este operacional es el encargado de amplificar las señales provenientes del micrófono y

amplificarlas para que puedan ser leídas por un microcontrolador. En teoría de circuitos, la configuración que estamos empleando se llama amplificador inversor. Existen otras configuraciones como el amplificador no inversor, sumador, restador, integrador, etc.



amplificador inversor

¿Por qué se llama amplificador inversor? Pues porque la señal de entrada se amplifica y a la vez se le cambia de signo, es decir, que si a la entrada hay 2 mV la salida será de -2mV considerando que la ganancia es unitaria. Los signos + y - no hacen referencia a la alimentación sino al lugar por donde entra nuestra señal.

Pero si te das cuenta en algún momento necesitaremos disponer de alimentación negativa en nuestro amplificador operacional. Se pueden generar tensiones negativos con los integrados TCP7660 o NE555 pero hoy no estamos de humor como para hacer eso. La solución es añadir un offset por la entrada positiva + del amplificador. Con ello la onda resultante se amplificará y variará entre la tensión de offset que hayamos puesto. Si nuestra señal varía entre +1V y -1V y aplicamos un offset de 1.65 nuestra señal ahora variará entre 2.65V y 0.65V. La ganancia del amplificador surge de dividir R2 entre R1.

En nuestro caso añadiremos un offset de 1.65V a partir de un divisor de tensión con dos resistencias de 1K alimentado desde los 3.3V de Arduino UNO. En cuanto la ganancia las resistencias R2 y R1 otorgan una amplificación de 580 veces la señal de entrada. El amplificador puede ser un LM358 o un OPA2344 ya que son los únicos de ser alimentados solo con tensión simple.

Además es necesario colocar dos condensadores C2 y C3 justo al lado de los pads de la alimentación del amplificador operacional y la tensión de 3.3V de Arduno UNO. Esto muy importante ya que se genera ruido por las altas conmutaciones de la pantalla táctil TFT. Este ruido si no se filtra da como resultado una pésima medición del sonido. Este fenómeno ocurre solo cuando lo alimentamos con una batería. Si lo estamos alimentando desde el ordenador no ocurre el problema.

Los dos pulsadores S3 y S4 sirven para cambiar de modo de funcionamiento. Aunque la pantalla TFT es táctil, cuando se configura el ADC por registros la librería TouchScreen que controla la pantalla deja de funcionar.

Finalmente la salida del amplificador se lleva a la entrada analógica número cinco. La señal que medirá Arduino será una señal con un offset de 1.65V que oscilará entre 3.3V y 0V.

2. Software

Convertidor Analógica a Digital (ADC)

La característica protagonista del proyecto es **la tasa de muestreo del ADC** de Arduino. Por defecto se emplea la función *analogRead()* que funciona a 9600Hz. Según Nyquist para que se pueda leer una señal sin perder informacion de esta es necesario leer o **muestrear** esa señal al doble de la frecuencia de la señal que pretendemos medir. Luego con 9.6 KHz solo podemos medir señales de hasta 4.8 Khz (la mitad). Para solucionar este problema podemos configurar el ADC de Arduino por registros en el mismo IDE de Arduino UNO mediante ADMUX y ADCSRA.

Con el registro ADMUX podemos seleccionar desde qué entrada analógica queremos leer con los bits 0,1,2 y 3.

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

ADC Multiplexer Selection Register

A cada bit le daremos un 1 o un 0 formando un numero binario de cuatro cifras. Atendiendo a la siguiente tabla podemos obtener la entrada analógica.

MUX 3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Temp Sensor (ATmega168/328 only)
1001 - 1101	(reserved)
1110	1.1V (ATmega168/328) 1.30V (ATmega8)
1111	0V (GND)

MUX Bits

Con los bits 7 y 6 (REFS1 y REFS0) podemos modificar la tensión de referencia. En este caso seleccionaremos la opción de AREF con Vref apagado (0 0). Más adelante veremos de lo que se trata.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVcc with external capacitor on AREF pin
1	0	Reserved
1	1	Internal 1.1V (ATmega168/328) or 2.56V on (ATmega8)

Con el bit 5 ADLAR se modifica la forma en la que la lectura es guardada. De este modo obtendremos para el registro ADMUX un valor de 00000101 en binario que en decimal representaría 0x5 (recordad que 0x es la forma de indicar a Arduino que estamos empleando código hexadecimal).

Por otro lado, con el registro ADCSRA seleccionamos el prescaler a partir de los bits 0, 1 t 2 .

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
ADCSRA	ADEN	ADSC	ADFR*	ADIF	ADIE	ADPS2	ADPS1	ADPS0

El prescaler es un número que determina la frecuencia de muestreo. El ADC funciona con el reloj que lleva Arduino UNO. En este caso es un cristal de cuarzo de 16Mhz. Como el ADC necesita 25 ciclos de reloj para hacer la primera lectura y 13 ciclos para las restantes, la frecuencia efectiva de funcionamiento será de 16Mhz/ 13 = 1.23Mhz. El ADC funcionará entonces a 1.23Mhz pero el prescaler divide esa frecuencia entre los números 2, 4, 8, 16, 32, 64 y 128. Eligiendo un prescaler de 32 la frecuencia con la que leeremos la señal será de 1.23Mhz/ 32 o lo que es igual a 38 Khz. Como vimos anteriormente según Nyquist debemos muestrear al doble de la señal a medir. En este caso como se muestrea a 38KHz podemos leer señales de hasta 19 Khz (la mitad). En la siguiente tabla con el numero binario 101 elegimos el prescaler de 32.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADPS Bits

El resto de bits del registro ADCSRA no son de interés. De este modo obtendremos un valor binario de 11100101 que en binario representa 0xE5

Finalmente se configurará la tensión de referencia del ADC. Por defecto Arduino lee hasta 5 voltios y a la hora de digitalizar esa lectura lo divide entre 1024 valores. El número de esa división surge porque el ADC tiene una resolución de 10 bits. Por lo tanto 2 elevado a 10 da 1024 valores. Este número no se puede modificar porque depende de las características de ADC pero sí que podemos modificar la tensión de referencia máxima que podemos leer. Si le decimos al

ADC que en vez de leer hasta 5V lea hasta 3.3V conseguiremos más resolución. Esto es porque 3.3 entre 1024 da valoren en tensión más pequeños que con 5V.

En el registro ADMUX configuramos REFS1 y REFS0 para que la tensión de referencia sea Aref. Si te fijas en tu Arduino, justo al lado del pin 13 hay un pin llamado Aref. La tensión que introduzcamos en ese pin marcará la tensión de referencia del ADC. En este caso usaremos 3.3V aprovechando que Arduino tiene una salida de 3.3V. ¡Pero ojo! La tensión que introduzcas no puede ser negativa ni mayor de 5V, además deberás declarar en el código que vas a emplear tensión externa con *analogReference(EXTERNAL).*

Pantalla TFT

Una vez resuelta la lectura del ADC se añade una pantalla TFT. La información de esta pantallas abunda en internet. Aquí hay un tutorial muy bueno donde explican cómo ponerlas en marcha: Tutorial Pantalla TFT. Lo que hay que tener en cuenta es que los colores están en formato RGB565. Si quieres modificar los colores te dejo este enlace: Color RGB565 Además, si por alguna razón los colores te aparecen invertidos elimina el signo "~" que aparece delante de los #define de los colores.

3. Algoritmos

El analizador de frecuencias posee varias funciones: medidor de amplitud, medidor de RMS y medidor de decibelios por parte del sonómetro. Por parte del analizador de frecuencias podemos visualizar el ancho de banda hasta 4Khz y hasta 19Khz.

Para tener una lectura fiable del valor de la señal del sonido en un instante no se hace una única lectura sino que se realizan 256 lecturas y luego se hace la media. Con esto eliminamos el error aleatoria de la lectura. El valor obtenido de

esa media será el valor de la señal mientras que la amplitud será el valor máximo conseguido en esas 256 lecturas realizadas.

Para calcular los decibelios basta con aplicar la siguiente ecuación.

$$dB = 20 \cdot \log\left(\frac{A_{sonido}}{A_{referencia}}\right)$$

Los decibelios es una unidad que mide el nivel de sonido. Como el oído del ser humano tiene una respuesta logarítmica ante el volumen de un sonido se aplica el logaritmo en base diez respecto a la ganancia de la señal (matemáticamente también se puede justificar el porqué del logaritmo). A_{sonido} representa el valor de la señal que valga la amplitud en el instante en el que estemos aplicando la ecuación y $A_{referencia}$ representa el valor respecto al cual la señal crece y decrece. En este caso como ya hemos visto es 1.65, es decir, el offset que añadimos en la etapa de amplificación. El resultado de esta ecuación es un número comprendido entre menos infinito y 0 ya que la escala es negativa. El infinito representará el silencio absoluto y el 0 el máximo valor que el micrófono puede ofrecer sin saturar el ADC (es decir pasar de 3.3 voltios).

El RMS significa Root Mean Square y en teoría de circuitos es la corriente contínua equivalente en una resistencia cuando se introduce una señal de corriente alterna. Para calcular tan solo hay que dividir la amplitud entre $\sqrt{2}$.

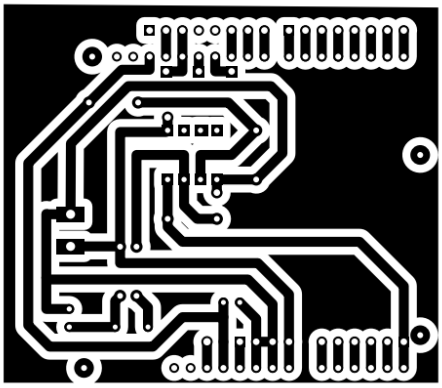
$$RMS = \frac{A_{sonido}}{\sqrt{2}}$$

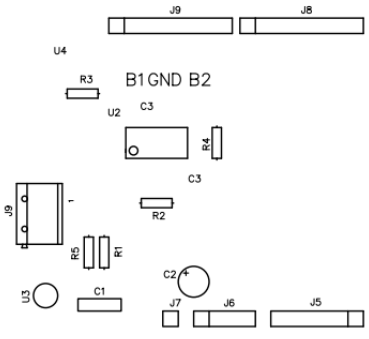
Finalmente para calcular la amplitud no hay que realizar nada adicional puesto que en las 256 lecturas que realizamos ya lo obtenemos. Tan solo hay que pasarlo a tanto por ciento.

En cuanto al analizador de audio se emplea la transformada rápida de Hartley (FHT) para representar en una gráfica las frecuencias en el eje X y el volumen en decibelios en el eje Y. Este algoritmo es igual que la transformada rápida de Fourier (FFT) salvo que tarda la mitad de tiempo de procesarse en un microcontrolador. La FHT tarda 3.18 mientras que la FFT 6.32 ms. Esto es debido a que la FFT trabaja con números complejos mientras que la FHT lo hace con números reales. De este modo empelaremos la FHT en nuestro Arduino UNO para quitarle carga matemática. Un detalle es **que los #defines LOG_OUT y FHT_256 deben estar declarados antes de incluir la librería.** Se puede encontrar más información al respecto en la página web de Open Music Labs

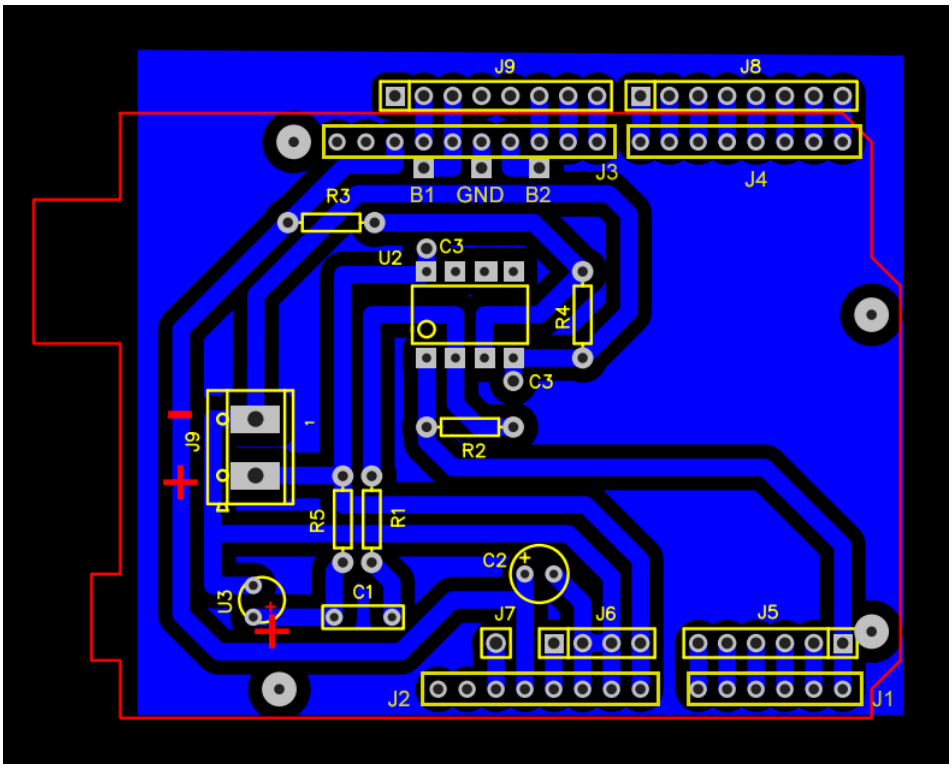
4. Esquema y materiales

Llegado a este punto no hace falta que comprendas toda la parte teórica. Este proyecto, aunque sea necesario muchos conocimientos para entenderlo, es muy fácil de construirlo. Además me he asegurado de que se empleen materiales comunes para un aficionado a la electrónica. Estas son las dos caras de la PCB que vamos a emplear. **Esto son solo imágenes representativas, las PCBs a tamaño real están al final del documento.**





En la siguiente página se encuentran la disposición de todos los componentes así como todos los materiales necesarios. Recuerda que se puede alimentar con una pila desde J9 o desde PC con el puerto USB.



Si quieres modificar el circuito aquí tienes el enlace a easyEDA: Analizador de Audio

- **R1 = 470R**
- **R2 = 270 K**
- **R3 = 1K**
- **R4 = 1K**
- **R5 = 2.2K**
- **C1 = 100nF (baja tensión de poliéster)**
- **C2, C3 = 470uF o superior (16V/25V bipolar)**
- **U2 = LM358 / OPA2344**
- **U3 = Micrófono electret**
- **J1,J2,J3,J4: tira de pines macho**
- **J5, J6, J7, J8, J9: tira de pines hembra**
- **J9: entrada de alimentación (opcional si lo alimentas desde el ordenador por UBS)**
- **B1, B2: pulsador momentáneo**
- **Arduino UNO**
- **Pantalla TFT 2.4 pulgadas (debe ser esta pantalla TFT)**

5. Bibliografía

Por si deseas profundizar en alguno de los temas expuesto te dejo varios enlaces a librerías y documentación.

Algoritmos:

https://blog.yavilevich.com/2016/08/arduino-sound-level-meter-and-spectrum-analyzer/

Datasheet Atmega328:

http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

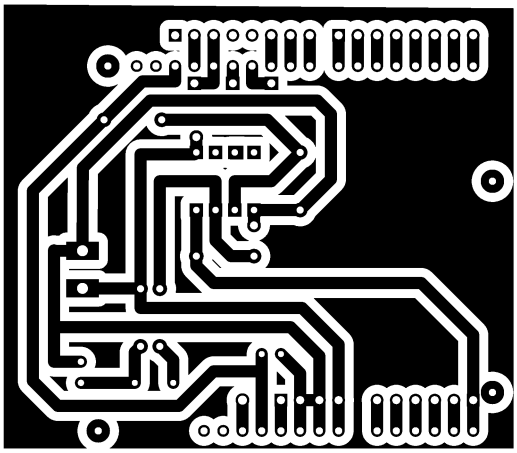
Configuración de registros ADC:
https://sites.google.com/site/qeewiki/books/avr-guide/analog-input

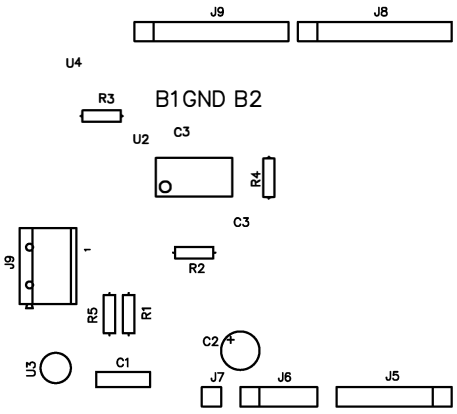
FHT Arduino UNO:

http://wiki.openmusiclabs.com/wiki/ArduinoFHT

Arduino con pantalla TFT:

https://electronicavm.wordpress.com/2015/03/05/tft-lcd-touch-2-4-shield-para-arduino-uno/




J9
J8
U4
R3
B1GND B2
U2
C3
R4
C3
J9
1
R2
R5
R1
C2
U3
C1
J7
J6
J5