‘KNOW HOW’ DEL MONTAJE EXPERIMENTAL ‘HYDROPORE’

Autor: Arnau Quintana Llorens

El presente documento pretende explicar el funcionamiento de cada una de las partes del sistema, tanto a nivel de circuito, el código y los sensores.

CONTROL DE CAMBIOS

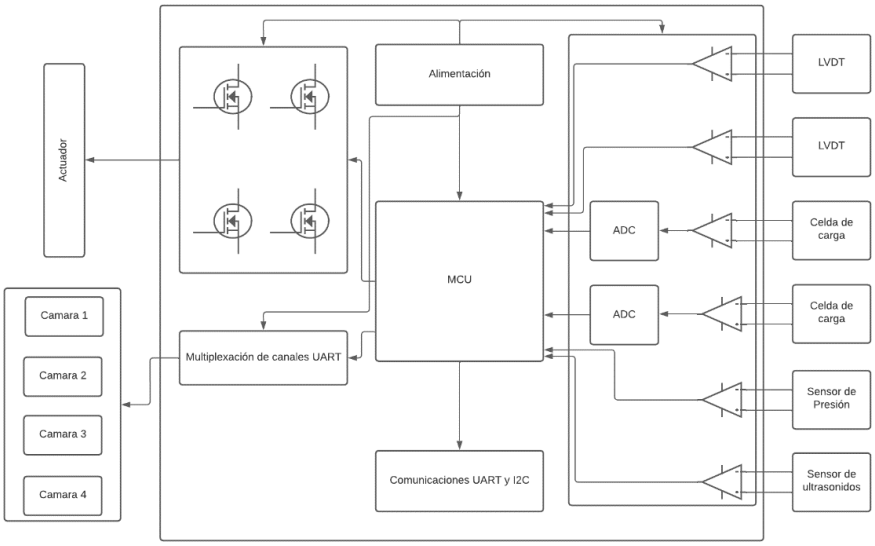
Autor	Apartado	Fecha	Versión
Arnau Quintana	TODOS	18-09-2022	0

ÍNDICE

PCB 3
LOAD CELLS 3
LVDTs 4
SENSOR DE PRESIÓN 4
ULTRASONIDOS 4
CÁMARAS 4
ACTUADOR 8
ALIMENTACIONES 9
POSICIÓN DE LOS CIRCUITOS EN EL PCB 10
ERRORES EN PCB 11
CÓDIGO EN C 12
CÓDIGO EN PYTHON 12
CÓDIGO EN ARDUINO 13
SENSORES 14
LVDT 14
SENSOR DE PRESIÓN DEL AGUA 14
LOAD CELLS 14
CÁMARAS 15

PCB

En este apartado se puede encontrar la explicación de todos los circuitos que forman el PCB.



LOAD CELLS

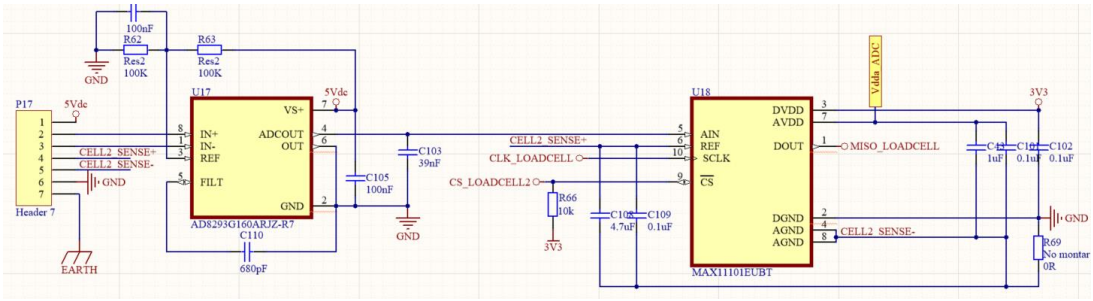
Las celdas elegidas para el proyecto disponen de 6 cables: Data+, Data-, Alimentación+, Alimentación-, Sensado de alimentación + y sensado de alimentación -.

El circuito para poder recoger datos de las celdas se compone básicamente de 2 partes: un amplificador de instrumentación y un ADC de 14 bits.

El amplificador de instrumentación nos ayuda a amplificar la señal y rechazar el modo común, que en este contexto es muy crítico al trabajar con señales tan pequeñas. Usamos un ADC externo porque el microcontrolador dispone de uno interno de 12 bits, el cual se no queda un poco corto para las señales que medimos de este sensor.

El amplificador tiene una ganancia fija de 160, es decir, la señal que entra saldrá amplificada 160 veces. El ADC dispone de 14 bits, es decir, de hasta 16384 valores digitales. Como lo alimentamos a 5V significa que por cada voltio disponemos de 3276 valores aproximadamente. Esto significa que 1V a la entrada del ADC, se traduce en un valor digital de 3276 a la salida. El cambio es lineal, es decir, siempre que añadamos peso, la salida cambiará de manera proporcional.

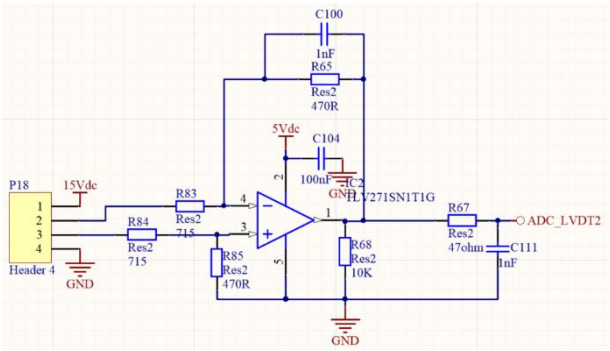
Nuestro rango medible es de 0N a 300N. Si quisiéramos aumentarlo, por ejemplo, hasta 600N, deberíamos, como mínimo, cambiar el amplificador y poner uno de ganancia fija menor; existe un modelo igual que en lugar de 160 de ganancia, tiene 80 (AD8293G80ARJZ-R7). El amplificador se alimenta a 5V, si intentamos sacar más de esa tensión, el componente se satura y a su salida veremos 5V por mucho que aumentemos por encima de 300N la fuerza aplicada en la celda.



LVDTs

Este sensor dispone de 4 líneas: +15Vdc, GND, Datos+ y Datos-. Nos devuelve una señal de 0V a 5V dependiendo de la extensión del núcleo. Nuestro sensor es 3.3V tolerante, por lo que aparte de acondicionar la señal analógica debemos reducir su amplitud. Para ello usamos un amplificador diferencial, que básicamente es un amplificador operacional con una cierta topología capaz de amplificar la diferencia entre sus entradas. Si en la entrada positiva tenemos 6V y en la otra 1V, la salida será de 5V, siempre y cuando no la amplifiquemos jugando con el valor de las resistencias. Nosotros hemos dimensionado el circuito teniendo en cuenta que la señal máxima que nos puede devolver el LVDT es de 6V (haciendo pruebas en el laboratorio vimos que esa era la tolerancia), por lo que cuando el LVDT devuelva 6V, el microcontrolador recibirá 3.3V.

$$V_o = \left(\frac{R_{65}}{R_{83}}\right) \cdot (V_{PIN2} - V_{PIN3})$$



SENSOR DE PRESIÓN

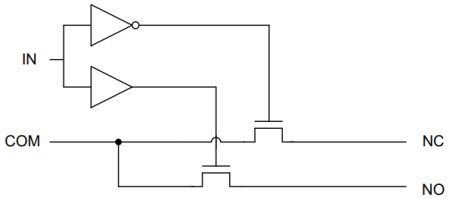
Este sensor es muy parecido en cuanto a funcionamiento al LVDT. Se alimenta a 15V y nos devuelve una señal de 0V a 5V dependiendo de la presión leída. Usamos el mismo circuito que en los LVDT para acondicionar la señal.

ULTRASONIDOS

Lo único que hemos tenido que añadir ha sido un pequeño divisor resistivo, ya que el sensor nos devuelve una señal digital de 0V a 5V, mientras que nuestro microcontrolador soporta de 0V a 3.3V. Este sensor sólo tiene parte digital, por lo que no hemos tenido que acondicionar nada.

CÁMARAS

En el PCB tenemos un circuito dedicado a la comunicación con las cámaras, pero parte de él no lo usamos por cuestiones de eficiencia. Lo compone 2 partes: un multiplexor digital y un driver RS232. El primero nos permite usar 2 canales UART para transformarlos en 4, tal como muestra la siguiente imagen:

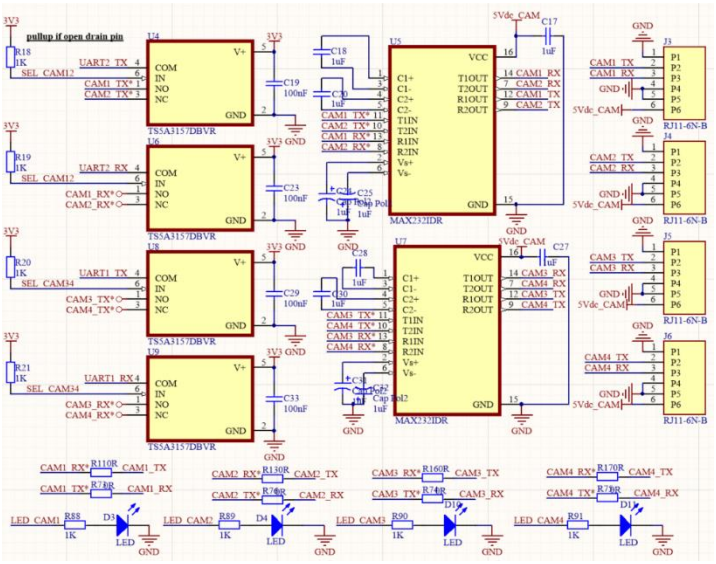


IN es el selector de canal, COM es la señal que queremos multiplexar, NC y NO son las dos salidas. Es un circuito bidireccional, por lo que podemos ir de COM a NC o NO, o podemos ir de NC o NO a COM; esto nos permite tanto enviar datos como recibirlos a través de la misma línea.

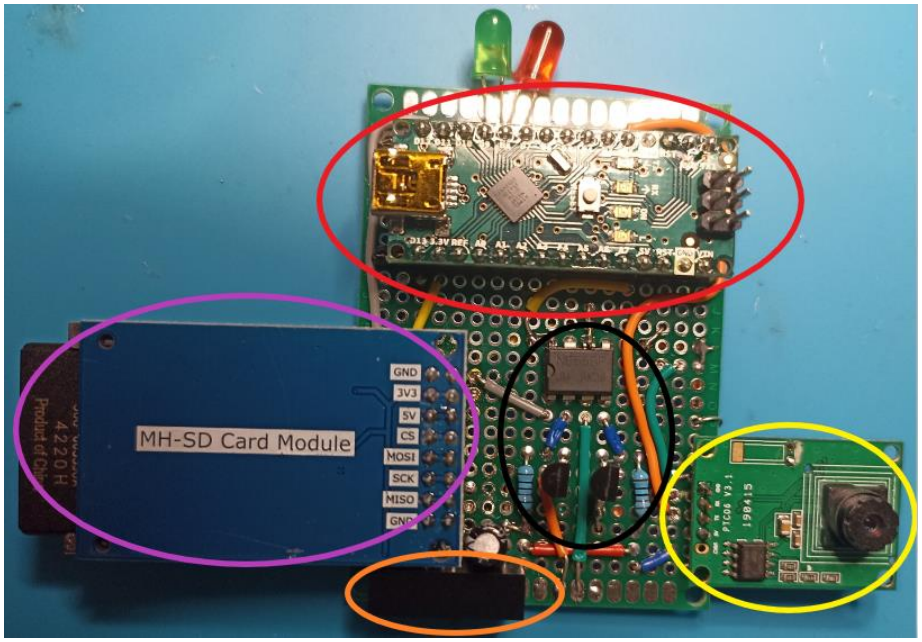
El driver RS232 nos ayuda a pasar la señal de niveles UART a niveles RS232, el cual es más seguro para comunicaciones en las que haya cierta distancia.

Los puentes de la parte inferior de la imagen son los que usamos en nuestro circuito final, ya que sirven para poder saltar el driver RS232.

En el montaje final únicamente se usa el multiplexor digital y los puentes para saltar el driver RS232 como comentamos. Esto es así porque no nos comunicamos a través de UART con las cámaras ya que cada una de ellas tiene un Arduino dedicado. Las líneas que van a las cámaras son la de 'trigger' y la de 'acknowledge' del 'trigger'.



La solución final para tomar las imágenes con las cámaras es la siguiente:

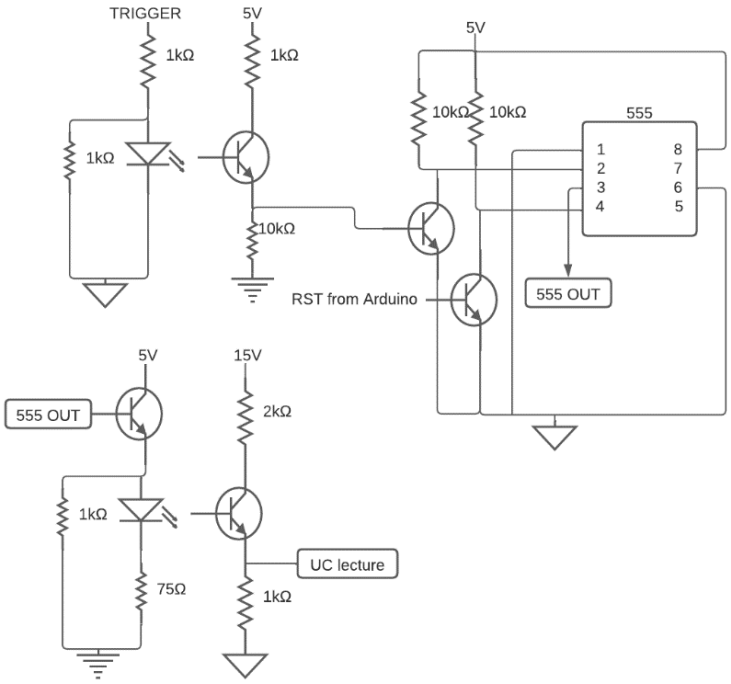


COLOR	PARTE DEL SISTEMA
	Arduino
	Circuito 'toggle' ON/OFF
	'Shield' tarjeta SD
	Convertidor DCDC aislado 15V a 5V
	Cámara serie

Alimentamos el módulo con el convertidor DCDC MPF1505S-1W, que le entran 15V desde la PCB principal a través del cable y los reduce a 5V, que es la tensión con la que funciona todos los componentes del módulo.

En esta PCB se sitúa: el 'SHIELD' donde se conecta la tarjeta SD en la que guardaremos las imágenes, la cámara que toma las imágenes, el Arduino que controla la faena, un convertidor que alimenta toda la placa y un pequeño circuito para detectar el 'trigger' que nos envía el microcontrolador principal para saber cuándo debemos empezar o parar de tomar imágenes, ya que una vez empezamos, cada 20 segundos aproximadamente se tomará una imagen.

Existe una cuestión muy importante a tener en cuenta en este circuito: la referencia de la parte del Arduino y la referencia del microcontrolador principal no son la misma, ya que usamos un DCDC aislado para alimentar todo el módulo. Debemos separar el 'trigger' que enviamos desde el microcontrolador principal al Arduino y la señal de 'trigger correcto' que envía el Arduino al microcontrolador principal. Para esta misión hemos usado optoacopladores, son dispositivos dedicados específicamente a lo que necesitamos.



Las siguientes tablas resumen las conexiones entre la cámara, la tarjeta SD y el Arduino:

Pin Arduino	Pin adaptador tarjeta SD
GND	GND
3.3V	3V3*
5V	5V*
D10	CS
D11	MOSI
D13	SCK
D12	MISO
GND	GND

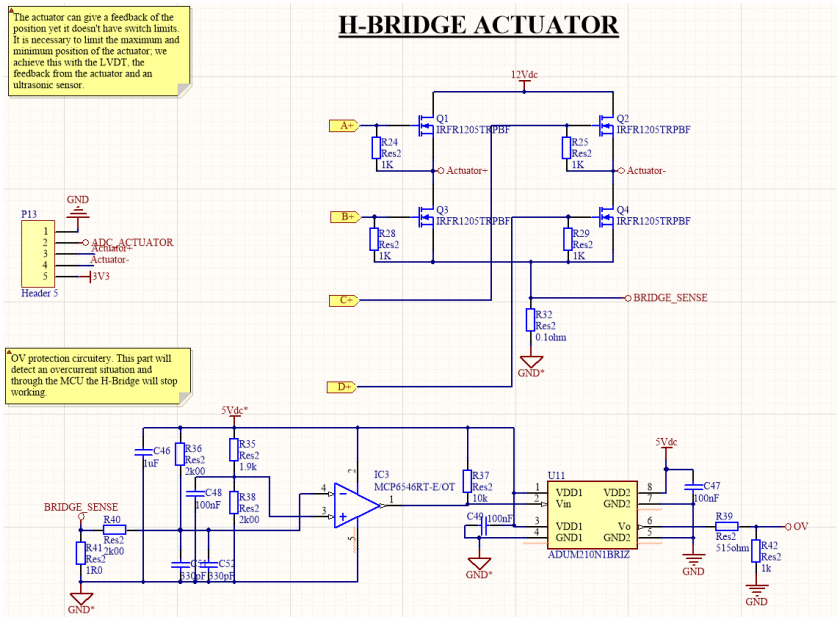
*Si conectamos 5V, no conectamos 3V3. Si conectamos 3V3, no conectamos 5V.

Pin Arduino	Pin Cámara
GND	GND
D3	RX (divisor resistivo 5V a 3V3)
D2	TX
5V	5V

Pin Arduino	Circuito 555
D5	555 OUT
D6	RST from Arduino

Cable RJ11	Módulo Cámaras
Cable rojo	TRIGGER
Cable negro	UC lecture
Cable blanco	15V
Cable gris	GND

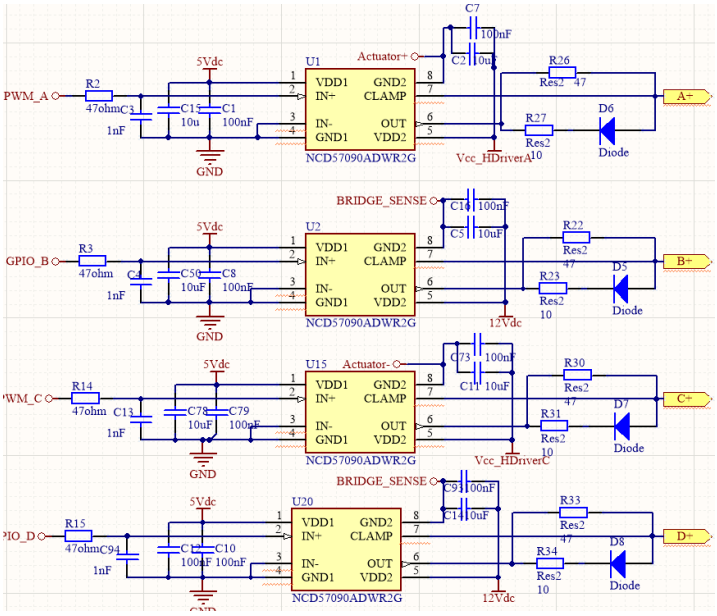
ACTUADOR



En esta primera imagen vemos el puente H con el circuito detector de sobre corriente. El actuador se moverá hacia adelante o hacia atrás dependiendo de cómo lo polaricemos. La manera más sencilla y efectiva de usar esa capacidad es con un puente en H, si cerramos los transistores en diagonal haremos fluir la corriente a través de la carga, que es el actuador, en la dirección que nos interese.

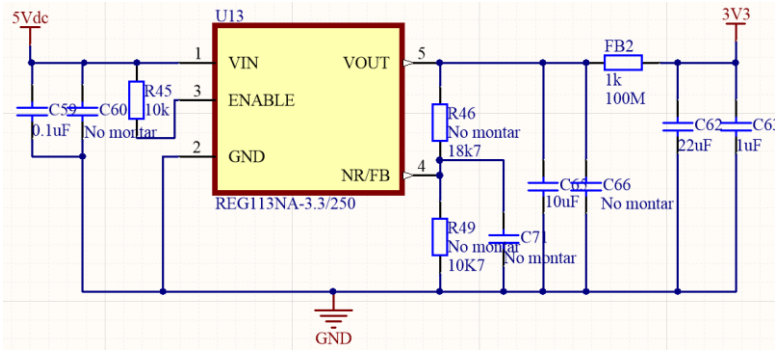
Por otro lado, tenemos el circuito de la parte inferior de la imagen. Mediante una SHUNT (una resistencia de baja impedancia y tolerancia) sensaremos la corriente que atraviesa el puente para poder proteger a los transistores de un cortocircuito o de un consumo excesivo. Se basa en la ley de ohm, por ejemplo, si la corriente que atraviesa la SHUNT es de 1A y la resistencia de es 1mOhm, la señal 'BridgeSense' será de 1mV ($V = I \cdot R$). Lo que hacemos con esa señal es llevarla directamente a la entrada negativa de un comparador. En la entrada positiva tenemos un nivel de continua constante. Si la señal de la SHUNT no supera esa señal de continua, todo va bien, la salida será 1 y no tendremos que hacer nada. En cuando la salida sea 0, deberemos actuar para parar el movimiento del actuador, ya que algo no está funcionando como debería.

En la segunda imagen, que encontramos más abajo, tenemos los drivers de los transistores. Estos se dedican a adaptar la señal del microcontrolador a una que sea correcta para la puerta de los MOS, ya que son muy delicados. Además, esos drivers son aislados, lo que significa que la referencia de la parte izquierda y la de la derecha no son iguales. Alimentamos el puente H con la fuente de 12V y el microcontrolador se alimenta indirectamente de la fuente de 15V, sus referencias son distintas y hemos de tenerlo en cuenta. Los componentes que rodean al driver son: un filtro RC en la entrada (para eliminar componentes de alta frecuencia de la señal), capacidades de desacoplo en ambos lados del driver (para mantener estable la tensión de alimentación), una resistencia de puerta (para limitar la corriente que llega a la gate del MOSFET y una resistencia y un diodo de descarga de la puerta (para descargar la carga de la puerta mucho más rápido cuando debamos abrir el transistor).

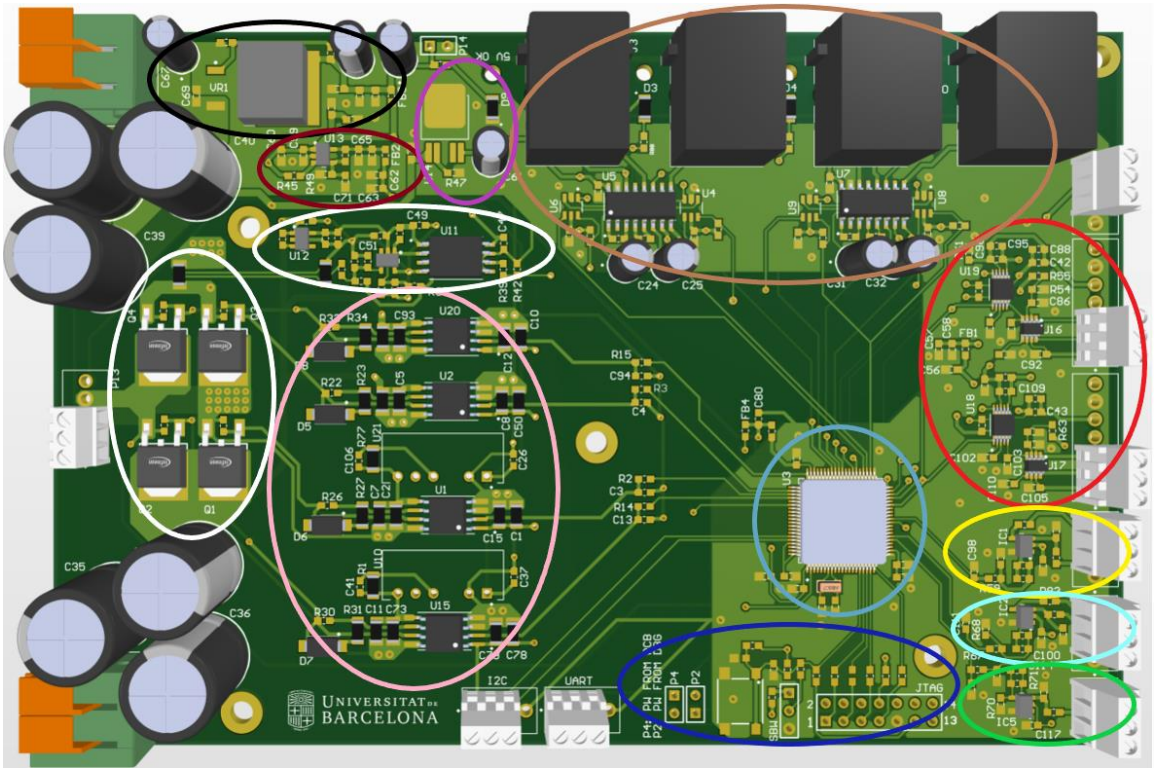


ALIMENTACIONES

Todas las tensiones que obtienen a partir de 2 fuentes, una de 12V para la parte del puente H, otra de 15V para toda la parte de control y sensores. Para obtener todas las alimentaciones que necesitamos, ya que, por ejemplo, el microcontrolador necesita 3.3V y algunos sensores 5V, usamos LDOs (reguladores lineales). Más abajo tenemos una imagen del circuito, todos son parecidos: condensadores en la entrada y salida para mantener estable la tensión y reducir el rizado.



POSICIÓN DE LOS CIRCUITOS EN EL PCB



COLOR	PARTE DEL SISTEMA
	LDO 15Vdc a 5Vdc para sensores, drivers del puente H y uControlador
	LDO 5Vdc a 3.3Vdc para el uControlador
	Regulador lineal de 15Vdc a 5Vdc para Arduinos y cámaras
	Driver a RS232 y multiplexores digitales para las cámaras
	Puente H y circuito de detección de sobre corriente
	Drivers del puente H
	Conector de programación del uControlador
	uControlador
	Acondicionamiento de la señal del sensor de presión
	Acondicionamiento de la señal del LVDT1
	Acondicionamiento de la señal del LVDT2
	Acondicionamiento de la señal de las celdas de carga

ERRORES EN PCB

Este apartado pretende resumir los errores de diseño que se han cometido y que deberían corregirse de cara a una versión B de la PCB.

- Referencia de los convertidores de los drivers del 'high side' del puente H.

La referencia del secundario del convertidor que alimenta el driver del transistor 'A' debe ir conectada a la 'net' 'Actuator+', puesto que es esta la que está conectada al 'source' del transistor.

La referencia del secundario del convertidor que alimenta el driver del transistor 'C', debe ir conectada a la 'net' 'Actuator-', puesto que es esta la que va conectada al 'source' del transistor.

Para arreglar este problema en la versión A del PCB, se ha cortado la pista que une GND con la referencia de la parte del secundario del convertidor y se ha unido la referencia del convertidor a donde corresponde con un cable.

- Pin 'OUT' del amplificador de instrumentación cortocircuitado con GND.

Al realizar el esquema y pasar una línea por encima de la otra, el pin 'OUT' se cortocircuitó con la GND. El pin 'OUT' debe ir conectado al condensador que tiene el otro pin conectado a 'FILT'. En caso de duda, en el datasheet del componente aparece un esquema de cómo debe ir conectado el amplificador.

Para corregir el error en la versión A del PCB, se ha levantado el pin 'OUT' del driver para que no toque con el pad y se ha levantado el condensador que debe unir 'OUT' y 'FILT' para que tampoco toque con GND. Mediante un cable se ha unido el pin del driver con el condensador.

- LDO de la alimentación de las cámaras y Arduino no usado.

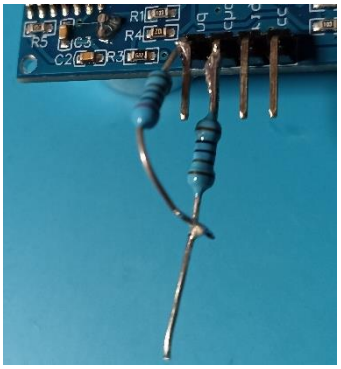
La primera idea del proyecto para la parte de las cámaras es que se comunicaran con el microcontrolador principal, por lo que únicamente deberíamos hacer llegar los 5V a las cámaras. Conforme avanzaba el proyecto y nos dimos cuenta de lo poco eficiente que era ese método, ingeniamos un módulo para poder resolver el problema, pero ese módulo consumía mucho más que la cámara sola. Al realizar las pruebas funciones del PCB nos dimos cuenta de que usar un LDO para alimentar los 4 módulos no era viable, ya que el consumo de cada uno de ellos es de aproximadamente 1.5W, lo que hacía calentarse en exceso al regulador.

Para solucionar el problema en la versión A de circuito hemos cortocircuitado los 5V de las cámaras con los 15V de la fuente, es decir, hemos quitado todos los componentes de la parte de alimentación de las cámaras y hemos cortocircuitado la entrada y la salida del regulador. Esto hace que la línea 5Vdc_CAM, sea en realidad, 15V. Esos 15V luego los pasamos a 5V en cada módulo con un convertidor DCDC (podemos revisar la explicación de cómo funciona el módulo un poco más arriba, en este mismo documento).

- Serigrafia en PCB y pines del ultrasonidos.

En la capa bottom del PCB tenemos el nombre de cada pin para poderlos identificar. Por motivos de software, los pines ECHO y TRIGGER están girados, es decir, donde en la placa pone 'trigger' conectamos el 'echo' y donde pone 'echo' conectamos el 'tigger'.

Además, hemos incorporado un divisor de tensión en el pin 'ECHO' del ultrasonidos tal como vemos en la imagen. Esa señal es de 5V mientras que el micro permite 3.3V.



CÓDIGO EN C

Usamos un MSP430 de Texas Instruments, concretamente el MSP430FR6043. Para programarlo, usamos el Code Composer Studio, la versión 10.4.0. El programador es el MSP-FET430UIF V1.

Tenemos el código separado por archivos intentando tener en cada uno funciones relacionadas entre sí.

Main.c	Archivo principal. El micro ejecuta la función 'main' situada en este archivo.
ADC.c	Funciones relacionadas con el ADC, desde configuración hasta recogida de datos.
ADC.h	Header del ADC.c. Encontramos las funciones que usamos en otra parte del código.
Control.c	Funciones principales, en este archivo usamos todas las librerías.
Control.h	Header del Control.c.
COMs.c	Funciones relacionadas con las comunicaciones UART y SPI.
COMs.h	Header de las funciones del archivo COM.c.
Clocks.c	Contiene la función de inicialización de los relojes.
Clocks.h	Header de la función de inicialización de los relojes.
DrivingH.c	Funciones relacionadas con el ultrasonidos y el movimiento del actuador.
DrivingH.h	Header con las funciones del archivo DrivingH.c.

En el código se han dejado comentada una parte del código, es decir, en la última 'release' del código no se incluye, su función es la siguiente: si el microcontrolador detecta medidas muy bajas en el LVDT (rondando los 0.01mm) que lo dé como error y no mueva el actuador, es decir, es una protección que no permitiría si quiera mover el motor sin el LVDT conectado. Se ha dejado sin usar porque puede darse el caso que el LVDT realmente mida 0.01mm y nos devuelva un error cuando no sería cierto.

Actualmente lo que pasaría que es el que el actuador se movería aunque el LVDT no estuviera conectado, pero cuando el microcontrolador detectara que las medidas del LVDT no varían, se pararía y nos devolvería un error. Si queremos usar esa protección simplemente habría que descomentar esa parte y reprogramar el microcontrolador.

CÓDIGO EN PYTHON

La aplicación tiene la misión de enviar comandos al microcontrolador, recibir los datos de los sensores y grabarlos en un documento de texto.

Usamos Jupyter Notebook como IDE para programar en Python. Hemos usado la versión Python3. Recomiendo instalar el Anaconda Navigator para poder trabajar cómodamente.

El código está organizado de la siguiente manera:

1. Función para cada botón.
2. Funciones de envío y recogida de datos.
3. Inicializaciones.

Usamos diversas librerías para poder comunicarnos y realizar las tareas:

1. Pyserial: para enviar y recibir comandos UART.
2. Time: para poder realizar timestamps de las medidas recibidas.

Para instalarlas deberemos abrir el 'Anaconda Prompt' (parecido al cmd de Windows, básicamente un intérprete de comandos) y escribir: 'pip install xxx', donde xxx es la librería por instalar.

CÓDIGO EN ARDUINO

Debemos comentar que Arduino está trabajando a las velocidades que recomienda el fabricante, si necesitamos más velocidad en la toma de imágenes podríamos intentar subir la velocidad de reloj de la comunicación SPI con la tarjeta SD y/o incrementar el baudrate de la comunicación serie con las cámaras.

Pasando a comentar el código, lo primero que hacemos es inicializar todos los recursos que usaremos, desde los puertos digitales a los módulos de comunicaciones. Durante todo el programa usamos las librerías que nos proporciona el fabricante, por lo que veremos funciones que no están definidas en nuestro 'main'.

El programa inicia comprobando que tengamos la tarjeta SD y la cámara conectada, de no ser así, el LED rojo parpadeará indefinidamente y dejamos el programa colgado.

Para poder volver a comprobar que está todo conectado, deberemos reiniciar el Arduino y que inicie el programa de nuevo. Si la comprobación ha sido correcta, entraremos en un bucle de espera en que estaremos alerta de un trigger enviado desde el microcontrolador principal. Lo sabremos porque el LED verde estará encendido mientras en rojo permanece apagado. Cuando se reciba el trigger, el LED rojo se encenderá y se apagará el verde, indicando que el sistema está ocupado.

En este momento tomaremos la imagen y se guardará en la tarjeta SD, debemos tener en cuenta que es una cámara serial, es decir, envía la información de la imagen por UART, lo que conlleva que el proceso demore unos segundos por imagen. Una vez Arduino la tenga, debe guardarla en la SD a través de SPI, que es un protocolo más rápido que el serie, pero a su vez que requiere de cierto tiempo. Una vez finalice el proceso (en unos 25 segundo por imagen), el LED verde se encenderá por un segundo y medio mientras el rojo está apagado.

Si no recibimos otro trigger, no entraremos en el modo espera y pasaremos directamente a tomar otra imagen.

Al final de cada secuencia se guarda el tiempo que ha pasado desde que Arduino ha arrancado hasta que se ha guardado la imagen en la SD, para poder tener una referencia del tiempo transcurrido y poder relacionarlo con los datos de los sensores.

SENSORES

En este apartado encontramos la explicación del funcionamiento de los sensores escogidos para el proyecto a nivel teórico.

LVDT

Usamos el modelo DC-SE 0-12.5MM de TE Connectivity.

La salida depende directamente de la extensión del núcleo: más dentro del cuerpo del sensor, mayor salida tendremos. En concreto para nuestro modelo, habrá una variación de 0.394V por milímetro. Esto no significa que la resolución sea esa, es una referencia que nos da el fabricante para poder hacer los correspondientes cálculos ya que el cambio en la salida entre movimientos es lineal.

El sensor permite alimentarse desde 8.5V a 28V, en nuestro diseño disponemos de 15V, y como está en el rango permitido, es la tensión que usamos.

SENSOR DE PRESIÓN DEL AGUA

Usamos el modelo 797-5037 de RS PRO, aunque es muy posible que cambie en un futuro.

El funcionamiento es muy parecido al del LVDT, pero en este caso debemos sacar nuestra conclusión con respecto a la variación de la salida. En este caso, el rango de la salida es de 0 a 5V, y el sensor mide hasta 10 bares, lo que significa que, cuando este midiendo 10 bares, la salida es de 5V. Con esta referencia podemos hacer los cálculos pertinentes para todas las demás medidas.

El sensor permite alimentarse desde 9V a 32V, en nuestro diseño disponemos de 15V, y como está en el rango permitido, es la tensión que usamos.

LOAD CELLS

La celda que usamos en el proyecto es la S2M de HBM, en concreto el modelo de 200N.

Las celdas de carga tienen una salida del tipo 2mV/V, es decir, cuando estén midiendo su carga nominal (en nuestro caso 200N), la salida será de 2mV por cada voltio aplicado de alimentación. Si nosotros alimentamos la celda con 5V, la salida será de 10mV cuando carguemos un peso de 200N.

Para saber mejor la salida teórica del sensor, hemos de seguir la siguiente fórmula:

$$U = U_0 * C * \frac{F}{F_{nom}}$$

U = salida en V, U_0 = alimentación (5V en nuestro caso), C = tipo de salida (nuestro caso 2mV/V), F = carga que aplicamos y F_{nom} = carga nominal de la celda (en nuestro caso 200N).

El circuito de la PCB está dimensionado para medir hasta 300N (en las especificaciones del proyecto se pidió medir hasta 200N, pero siempre es conveniente dejar un margen). Cuando en este contexto decimos 'dimensionar' nos referimos a nivel de componentes; en el apartado de la PCB podemos ver en detalle cómo funciona el circuito.

Cuando la celda mide 0N, en la salida del amplificador tendremos 2.5V. Esto es debido al funcionamiento de la propia celda, ya que se basa en un puente de Wheatstone. Esto se ha tenido en cuenta en el código Python, al convertir el valor digital a analógico.

El sensor permite alimentarse desde 0.5V a 12V, pero en nuestro diseño disponemos de 5V, y como está en el rango permitido, es la tensión que usamos.

CÁMARAS

Para poder controlar las cámaras hemos optado por dedicar un Arduino Nano a cada una de ellas. Son de tipo serie, por lo que necesitamos únicamente dos cables para obtener la información de ellas (RX y TX). Si no lo hiciéramos así, sólo podríamos tomar una imagen (ya que no podríamos hacer otra tarea en paralelo), y tardaríamos aproximadamente 2 minutos en hacer todo el proceso (tomar la foto, recibir los 'arrays' de datos y enviarlos al ordenador).

Con los Arduino tenemos la posibilidad de, independientemente de lo que haga el sistema, ir tomando imágenes y guardarlas en una tarjeta SD.

La cámara, la tarjeta SD y el Arduino van conectados entre sí, pero aparte tenemos un pequeño circuito en el centro que es necesario explicar. Básicamente es un circuito para mantener el 'trigger' que nos envía el microcontrolador principal hasta que el Arduino pueda leerlo. Imaginemos que el microcontrolador nos envía un 'trigger' pero el Arduino no está preparado para leerlo ya que está acabando de grabar una imagen en la SD; el microcontrolador sólo mantendrá el trigger durante unos milisegundos ya que debe hacer otras tareas. Podríamos llegar a perder el 'trigger'. Hemos montado un pequeño circuito ON/OFF controlado por 2 señales: SET y RESET.

El funcionamiento es el siguiente:

El pin 'SET' del BJT va conectado el microcontrolador, es el 'trigger' que recibiremos; se encarga de poner 'OUT' a 1. El pin RST del BJT va conectado al Arduino, una vez hayamos leído el 'trigger' deberemos volver a poner el 'OUT' a 0 a la espera de otro 'trigger'.

El microcontrolador principal envía el 'trigger' y debe leer el 'OUT' para ver si se pone a 1, si es así, lo da por bueno y sigue con otras tareas. Cuando el Arduino lea un 1 en el 'OUT', ejecutará un 'trigger' que implica empezar a tomar fotos o parar según lo que esté haciendo en ese momento y volverá a poner el 'OUT' a 0 a la espera del siguiente 'trigger' desde el microcontrolador.

Tanto el Arduino como el 'SHIELD' de la tarjeta SD se pueden extraer, no van soldados a la placa. Si lo hacemos, es muy importante volver a conectarlo como estaba, no funcionará de no hacerlo así. Es muy importante que todo esté conectado correctamente cuando conectamos la alimentación.