# CAPÍTULO 1

# INGENIERÍA INVERSA

El cuadricóptero adquirido resuelve la comunicación entre los controladores de los motores (**ESCs**) y el microprocesador mediante el protocolo **$i^2c$**.

Para el presente proyecto resulta de vital importancia conocer dicho protocolo, ya que se utilizará otro microprocesador que deberá comandar a esos mismos ESCs, supliendo el trabajo del anterior. Es entonces imprescindible conocer al detalle el funcionamiento de este protocolo, para luego poder reproducirlo.

Dado que no se cuenta con la mínima colaboración de los fabricantes y la información es completamente privativa, es necesario realizar un proceso de ingeniería inversa para poder analizar, decodificar, entender y reproducir el protocolo existente. Dicho proceso de ingeniería inversa se realiza utilizando el hardware existente del cuadricóptero comercial adquirido y un analizador lógico[1] que es capaz de leer e interpretar las líneas del bus $i^2c$ sin intervenir en las mismas.

Antes de presentar los resultados obtenidos en el proceso, se realiza una breve introducción al protocolo $i^2c$. En la figura 1.1 se presenta la definición de los ejes a utilizar, lo cual será de utilidad más adelante.


Figura 1.1: Definición de ejes

[1] ChronoVu - `www.chronovu.com` - Adquirido en `www.ebay.com` debido a las dificultades encontradas al intentar implementar uno.

## 1.1. Introducción al protocolo $i^2c$

El bus $i^2c$ es un bus de comunicaciones serie. Su nombre viene de *Inter-Integrated Circuit* (Circuitos Inter-Integrados).
Utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. Además será necesaria una tercera línea de tierra, como referencia.
En la imagen 1.2[2] se muestra un diagrama de un circuito equivalente simplificado de la conexión $i^2c$ entre 2 dispositivos.

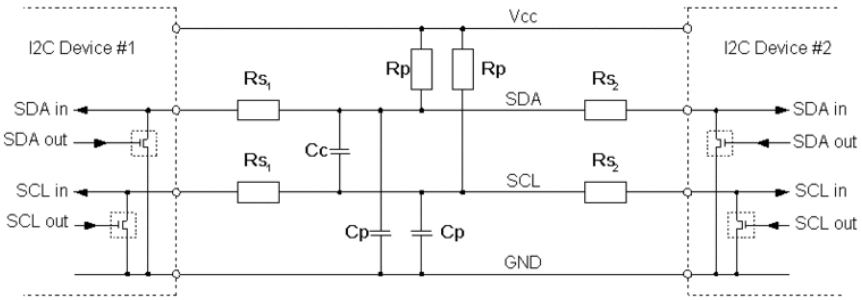



Figura 1.2: Conexión $i^2c$.

donde:

- `Vcc`: Voltaje de entrada, típicamente varía entre 1.2 V y 5.5 V
- `GND`: Tierra común
- `SDA`: Línea serial de datos
- `SCL`: Línea serial de reloj
- `Rp` : Resistencia de "Pull-up"
- `Rs` : Resistencia serie
- `Cp` : Capacitancia del cable
- `Cc` : Capacitancia de canal cruzado

Las líneas **SDA** y **SCL** son de drenador abierto, lo que significa que tanto el maestro como los esclavos solamente pueden conducir a nivel bajo estas líneas, o dejarlas abiertas. Si ningún dispositivo $i^2c$ está conduciendo hacia abajo la línea, la resistencia de *pull-up* $R_p$ se encarga de llevar la línea a $V_{cc}$.

El reloj siempre es generado por el maestro. El bus de datos debe mantenerse estable mientras el reloj está en nivel alto ("1" lógico) y se le permite cambiar de nivel mientras el reloj está en nivel bajo ("0" lógico). Se toma como dato válido el valor del bus de datos cuando el reloj está en "1".

Cada dispositivo tiene asignada una dirección que lo identifica. Típicamente para establecer una comunicación el maestro envía una secuencia de comienzo de conexión, seguida de la dirección del esclavo con el cual desea comunicarse. Seguidamente

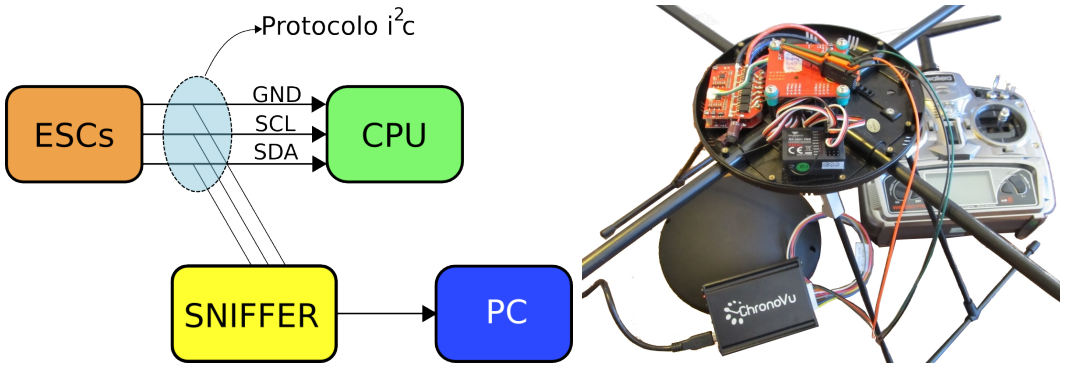
[2]Imagen tomada de www.i2c-bus.org

el maestro envía un bit que determina si la acción que desea realizar es escritura o lectura, a lo que el esclavo correspondiente responde con un bit de *acknowledge* (**Ack**). Luego el maestro envía la dirección de memoria interna del esclavo donde debe ser almacenada la información enviada, y por último envía los datos. Para finalizar la conexión, el maestro envía una secuencia de fin de conexión.

## 1.2. Pruebas en régimen

- Materiales
  - *Cuadricóptero:* transmisor, receptor, CPU y ESCs
  - *Analizador lógico ("sniffer"):* Chrono Vu
  - *PC*
- Procedimiento
  - La forma de operar es enviar comandos conocidos con el control remoto y analizar el flujo de datos en las líneas del bus.
  - En particular se analizará el funcionamiento en régimen, durante el arranque y durante el frenado.

La comunicación entre la CPU y los ESCs es mediante el protocolo $i^2c$, y es precisamente allí donde interviene el analizador lógico, "olfateando" todo lo que pasa por el bus. En la figura 1.3a se muestra un diagrama de bloques del proceso descripto, y en la figura 1.3b su implementación.




(a) Diagrama de interconexión (b) Proceso de "sniffeado"

Figura 1.3: Proceso de lectura del bus.

Al realizar este proceso se puede observar que se repiten, cada $2ms$, bloques similares al mostrado en la figura 1.4.

De acuerdo a la observación de dichos bloques, teniendo en cuenta la secuencia de comunicación típica entre maestro y esclavos mencionada anteriormente, se deduce que hay 4 esclavos correspondientes a los 4 ESC's de los 4 motores, cuyas direcciones en hexadecimal son **D0**, **D2**, **D4** y **D6**. La dirección de la memoria interna de los esclavos donde se almacenan los datos que envía el maestro es, en todos los casos **A2**. Además el maestro envía un tercer conjunto de datos que refiere, de alguna manera, a la velocidad con la que debe girar cada motor. Este conjunto de órdenes,

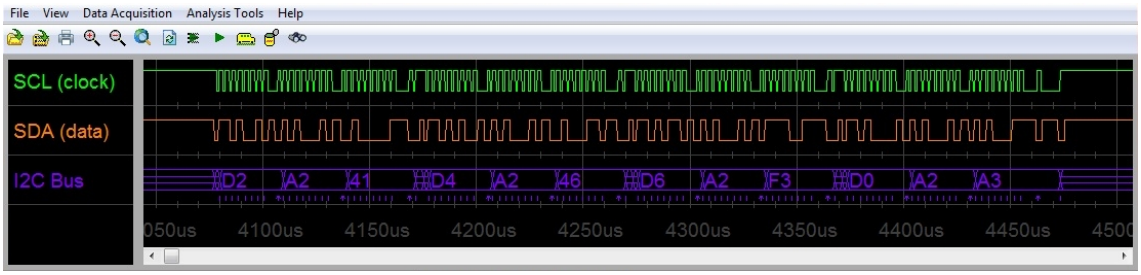



Figura 1.4: Bloque de transmisión $i^2c$

agrupadas por bloques como el que se muestra en la figura 1.4, se repite periódicamente, indicando la velocidad con la que debe girar cada motor.

Para comprender las pruebas realizadas es importante dejar en claro algunos aspectos previos. En la figura 1.5 se muestra el transmisor utilizado para enviar comandos al cuadricóptero, un **Walkera WK-2801** y se indican los nombres de los mandos más importantes del mismo.


Figura 1.5: Transmisor

Al mover el mando de la izquierda (**Elev/Rudder**) en la dirección vertical (**Elev**) se logra que el cuadricóptero se eleve verticalmente, dando igual potencia a todos los motores, mientras que al moverlo en la dirección horizontal, el cuadricóptero presenta un movimiento de rotación según su eje vertical.

Al mover el mando de la derecha (**Throttle/Aile**) en la dirección horizontal (**Aile**) y vertical (**Throttle**), se logran movimientos de rotación según los ejes horizontales $x$ e $y$ del cuadricóptero. Las definiciones de los ejes se pueden ver en la figura 1.1.

Se analizan las siguientes situaciones:

| Id | Elev | Rudder | Aile | Throttle | Movimiento |
|---|---|---|---|---|---|
| 0 | atrás | medio | medio | medio | idle |
| 1 | medio | medio | medio | medio | vertical hacia arriba con aceleración constante |
| 2 | adelante | medio | medio | medio | vertical hacia arriba con aceleración constante |
| 3 | medio | izquierda | medio | medio | giro según eje $z$ |
| 4 | medio | derecha | medio | medio | giro según eje $-z$ |
| 5 | medio | medio | izquierda | medio | giro según eje $-x$ |
| 6 | medio | medio | derecha | medio | giro según eje $x$ |
| 7 | medio | medio | medio | atrás | giro según eje $-y$ |
| 8 | medio | medio | medio | adelante | giro según eje $y$ |

Tabla 1.1: Pruebas realizadas

Se realiza un análisis de los resultados obtenidos para cada motor, graficando el contenido del byte de datos que se le transmite a cada motor. Se obtienen representaciones como la mostrada en la figura 1.6. En dicha figura puede observarse que los 4 motores reciben un byte con el valor promedio en 50, el cual corresponde a la mínima potencia entregada a los motores para encenderlos.

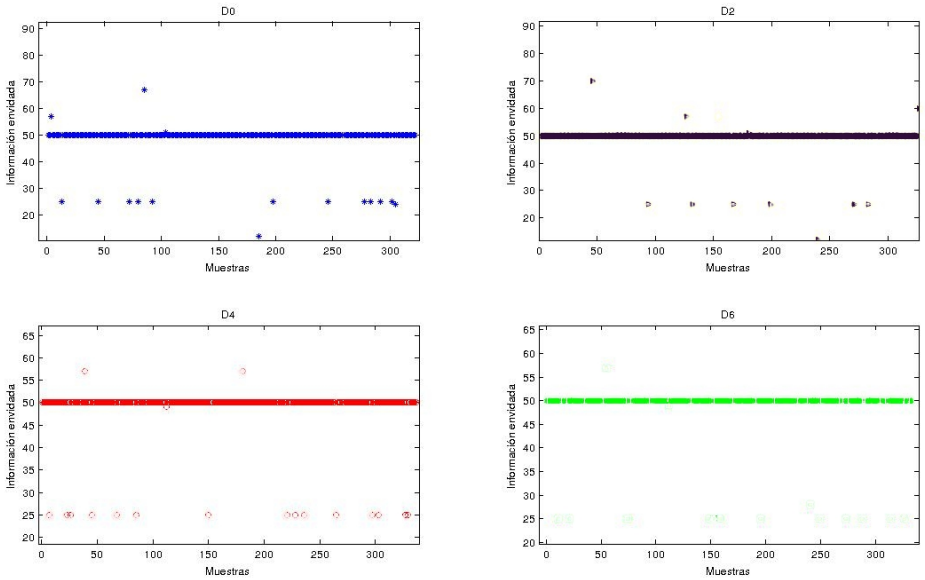



Figura 1.6: Prueba N$^o$ 0

Haciendo un análisis similar con el resto de las pruebas detalladas en la tabla 1.1 se construye la tabla 1.2 donde se muestran los valores enviados a cada motor en promedio en todas las pruebas.

| | Valor promedio | | | | |
|---|---|---|---|---|---|
| Prueba | D0 | D2 | D4 | D6 | Movimiento |
| 0 | 50 | 50 | 50 | 50 | Idle |
| 1 | 80 | 140 | 180 | 140 | $a_z = cte \neq 0$ |
| 2 | 180 | 220 | 250 | 240 | $a_z = cte \neq 0$ |
| 3 | 160 | 70 | 60 | 250 | giro según eje $z$ |
| 4 | 50 | 200 | 200 | 100 | giro según eje $-z$ |
| 5 | 250 | 160 | 140 | 50 | giro según eje $-x$ |
| 6 | 50 | 160 | 150 | 250 | giro según eje $x$ |
| 7 | 90 | 50 | 250 | 160 | giro según eje $-y$ |
| 8 | 110 | 250 | 50 | 100 | giro según eje $y$ |

Tabla 1.2: Resumen de los resultados obtenidos

En la tabla 1.3 se muestran los registros a los que se mandaron datos para cada dirección.

| | D0 | D2 | D4 | D6 |
|---|---|---|---|---|
| Registro | A2 | A2 | A2 | A2 |

Tabla 1.3: Resumen de las direcciones obtenidas

De las anteriores 2 tablas se pueden sacar algunas conclusiones que se analizarán en la sección 1.5.

## 1.3. Prueba de arranque

En esta sección se analiza la secuencia de arranque del cuadricóptero. Se parte con la palanca de elevación al mínimo y se procede a moverla para arrancar los motores.

Mientras la palanca de elevación se mantiene al mínimo, se le envía el valor 0 a los cuatro motores. Al mover la palanca, luego de un *tiempo muerto* donde las líneas quedan inactivas, se les empieza a enviar el valor correspondiente a cada motor, como se puede ver en la figura 1.7.

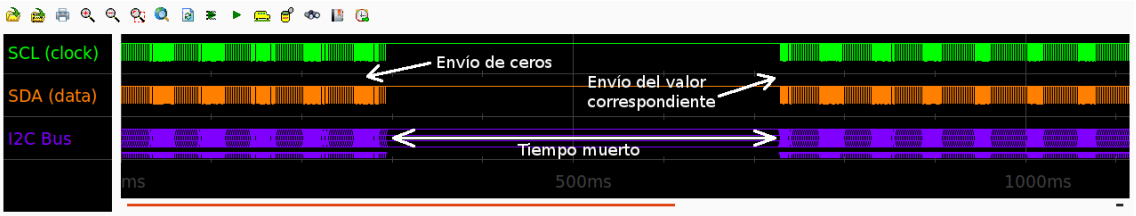
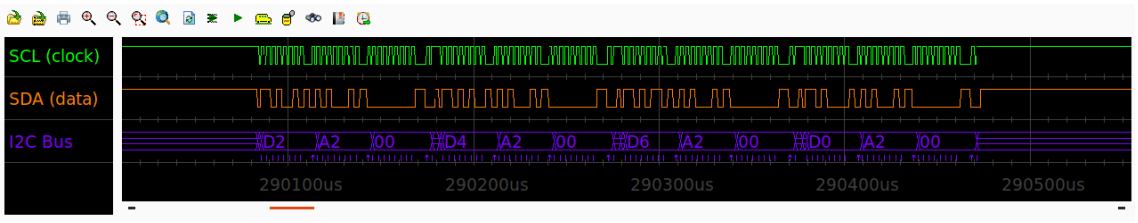


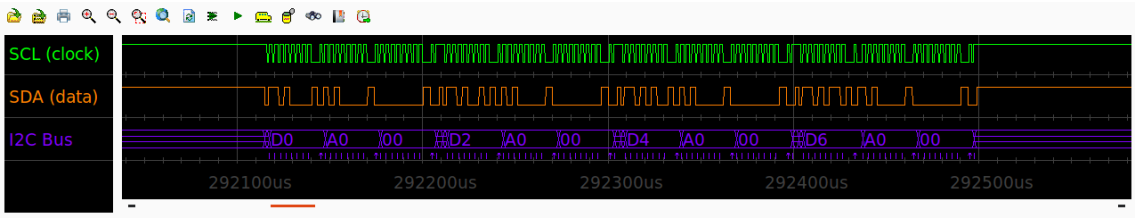

Figura 1.7: Arranque

Observando más detenidamente los comandos enviados se pueden obtener algunas conclusiones importantes. En la gran mayoría de los casos se envían los comandos en el orden $D2 \rightarrow D4 \rightarrow D6 \rightarrow D0$ y se guardan en el registro 0xA2 del esclavo.

Pero se pueden observar dos importantes diferencias en el último comando que se manda a cada motor antes de arrancar, es decir, el último comando en la tanda de ceros mostrada en la figura 1.7:




(a) Tanda normal de ceros




(b) Última tanda de ceros

Figura 1.8: Envío de ceros

- **El orden en el que se envían datos a los motores no es mismo** que el anterior. En este caso el orden es: $D0 \rightarrow D2 \rightarrow D4 \rightarrow D6$.
- **El registro en el que se escriben los datos es 0xA0**

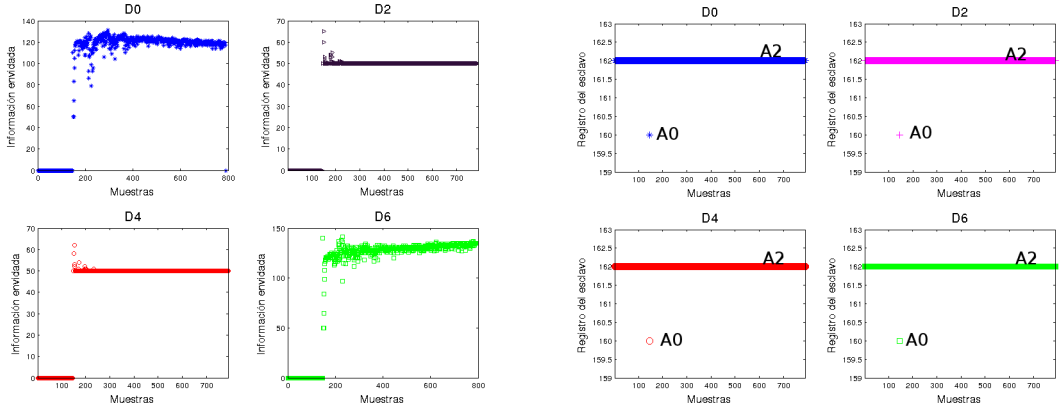
En la figura 1.8a se muestra una tanda regular de ceros y en la figura 1.8b se muestra la última tanda de ceros mencionada.

Dichas diferencias se pueden observar claramente al graficar todos los datos obtenidos, como se muestra en la figura 1.9. Se grafica para cada motor el valor que le llega (figura 1.9a) y el registro donde el valor es escrito (figura 1.9b). Al analizar los resultados gráficamente resulta evidente el cambio en el registro al terminar de mandar los ceros. En la figura 1.9a se puede observar que el último cero que se manda a los motores, se manda alrededor de la muestra 150. Analizando la figura 1.9b es clarísimo que alrededor de la muestra 150, el registro al que se envían los comandos cambia al valor 160 (0xA0), tal como se había observado anteriormente.

## 1.4. Prueba de frenado

Se procede del mismo modo que en la prueba anterior (sección 1.3), analizando los comandos enviados a los motores a la hora de apagarlos. Se realiza una prueba análoga a la anterior partiendo de los motores funcionando y bajando al mínimo la palanca de elevación, causando que estos se detengan.

Una vez que se lleva la palanca de elevación al mínimo, los motores se apagan y se permanece enviando ceros a los motores en el orden $D2 \rightarrow D4 \rightarrow D6 \rightarrow D0$. Se pueden obtener algunas conclusiones importantes de la última tanda de




(a) Valores enviados a los motores (b) Registro al que se envía el valor

Figura 1.9: Arranque

valores distintos de cero enviados (comandos que provocan el frenado de los motores), mostrada en la figura 1.10:

- **El orden en el que se envían datos es** $D0 \rightarrow D2 \rightarrow D4 \rightarrow D6$
- **El registro en el que se escriben los datos es 0xA1**

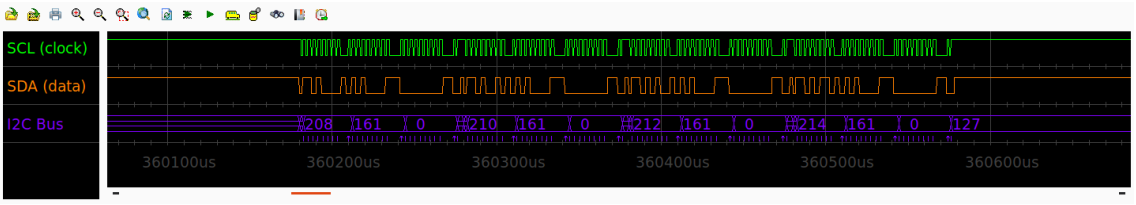


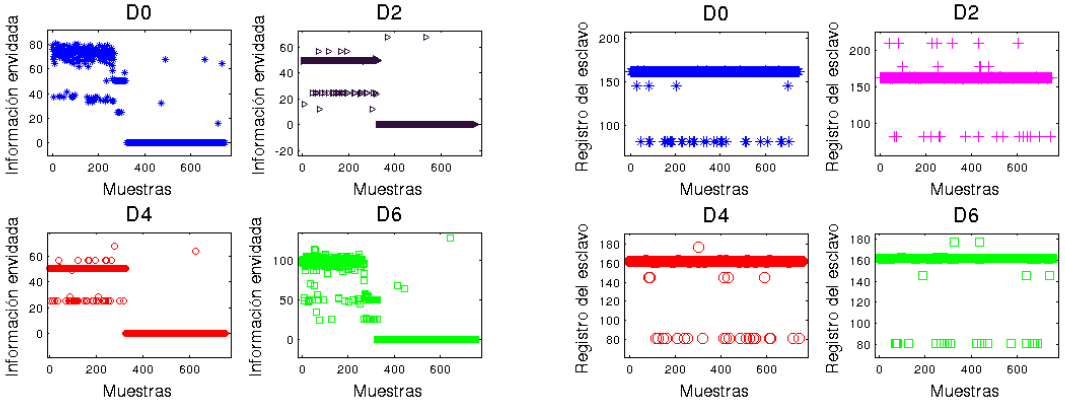

Figura 1.10: Frenado

Se presenta también un análisis gráfico en la figura 1.11. En este caso, a diferencia del de la sección 1.3, no es posible divisar con claridad el registro 0xA1 (161) en la figura 1.11b, ya que se escriben comandos a una cantidad importante de registros diferentes.

Avalados por la prueba realizada en la sección 1.6 que se verá luego, se puede afirmar que la aparición de otros registros (por ejemplo los números 81, 145, etc) son causados por el propio proceso de intervención en los buses de comunicación y no son el comando transmitido a los motores. Analicemos por ejemplo el valor 81. Es claro que el resultado de dividir el valor 162 (correspondiente al conocido registro 0xA2) entre 2 es 81, por lo cual parece probable que haya sido generado por un error del analizador lógico, ya que un corrimiento hacia la derecha de un bit en una palabra binaria es equivalente a realizar una división entre 2. Es probable entonces que el sniffer haya incluido un bit en 0 antes de la verdadera palabra y haya descartado el último bit, causando una aparente división entre 2. Más gráficamente:

$$162 = 10100010 \rightarrow 0\,1010001\,\cancel{0} = 81$$

En verde se muestra el bit agregado y en rojo el eliminado.




(a) Valores enviados a los motores

(b) Registro al que se envía el valor

Figura 1.11: Frenado

## 1.5. Conclusiones

- La comunicación entre el amo y los esclavos por medio del protocolo $i^2c$ se lleva a cabo mediante un formato del tipo:

  *Dirección esclavo - Lugar de memoria donde guardar dato - dato*

- Dicho formato se repite para todos los esclavos
- Cada esclavo recibe una actualización de estado (un dato nuevo) cada $2ms$
- La velocidad mínima de funcionamiento se logra enviando el valor 50
- La velocidad máxima de funcionamiento se logra enviando el valor 250
- Cuando el mando de Elevación del control se encuentra al mínimo, se les envía el valor 0 a los cuatro motores periódicamente con el formato mencionado.
- La dirección de memoria interna de todos los esclavos donde se guardan los datos recibidos por el maestro es siempre 0x$A2$ (ó 162), a excepción del arranque y el frenado.
- El último comando enviado antes de arrancar se guarda en el registro 0xA0, comando que se interpreta como una orden de arranque.
- El último comando enviado antes de frenar se guarda en el registro 0xA1, comando que se interpreta como una orden de frenado.
- El orden normal en el que se envían los comandos es $D2 \rightarrow D4 \rightarrow D6 \rightarrow D0$, menos en los casos descriptos en los 2 puntos anteriores donde el orden es $D0 \rightarrow D2 \rightarrow D4 \rightarrow D6$. Se comprobó, experimentalmente, que no es necesario alterar el orden en el que se envían los comandos para lograr arrancar y frenar los motores.
- La correspondencia entre las direcciones y los motores se muestra en la figura 1.12. El motor con dirección 0xD0 es el de adelante, el motor con dirección 0xD4 es el de atrás y mirándolo de frente el de la izquierda se corresponde con 0xD0, y el de la derecha con 0xD6

- Las direcciones 0xD0, 0xD2, 0xD4 y 0xD6 (de 8 bits) en realidad no refieren únicamente a direccionamiento, sino que incluyen además el bit de lectura/escritura, por lo cual cada esclavo tendrá una única dirección de 7 bits. Notar que las direcciones 0xD0 a 0xD6 al pasarlas a binario todas terminan en 0, lo cual implica que se tratan de comandos de escritura. Al omitir el último bit se obtienen las direcciones (sin incluir el bit de lectura/escritura) como se puede ver a continuación:

$$0xD0(11010000) \longrightarrow 0x68(1101000) \quad (1.1)$$
$$0xD2(11010010) \longrightarrow 0x69(1101001) \quad (1.2)$$
$$0xD4(11010100) \longrightarrow 0x6A(1101010) \quad (1.3)$$
$$0xD6(11010110) \longrightarrow 0x6B(1101011) \quad (1.4)$$

  Las direcciones de los motores son entonces: **0x68**, **0x69**, **0x6A**, **0x6B**.

- En las pruebas 0, 1, 2, 5, 6 y 7 se observa que la suma del valor promedio enviado al motor D0 más el enviado a la dirección D2 es similar a la suma de los valores entregados a los motores con dirección D4 y D6, lo cual implica un equilibrio en los pares realizados en ambos sentidos.

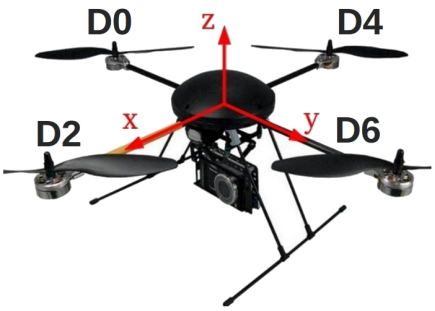



Figura 1.12: correspondencias

## 1.6. Verificaciones

Se presentan diversas verificaciones realizadas para corroborar la veracidad de las conclusiones extraídas. Además de la corroboración experimental utilizando el puerto $i^2c$ de la Beagleboard y logrando hacer funcionar los motores, previamente se realizaron 2 verificaciones adicionales:

- **i2cdetect**: comando enviado desde la *BeagleBoard* por el puerto *i2c-2*.
- **MSP430F5438**: microcontrolador que se programa como esclavo con una de las direcciones de un ESC.

### i2cdetect

Utilizando la librería *i2c Tools*[3] con la Beagleboard es posible enviar el comando **i2cdetect**, el cual "pregunta" a todos los esclavos presentes en el bus por su dirección. El resultado en terminal es el siguiente

[3]http://www.lm-sensors.org/wiki/i2cToolsDocumentation

```
root@beagleboard:~# i2cdetect -r 2
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-2 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] y

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 69 6a 6b -- -- -- --
70: -- -- -- -- -- -- -- --
```

Puede verse claramente que las direcciones de los esclavos obtenidas por la función mencionada son: **0x68**, **0x69**, **0x6A**, **0x6B**.

## MSP430F5438

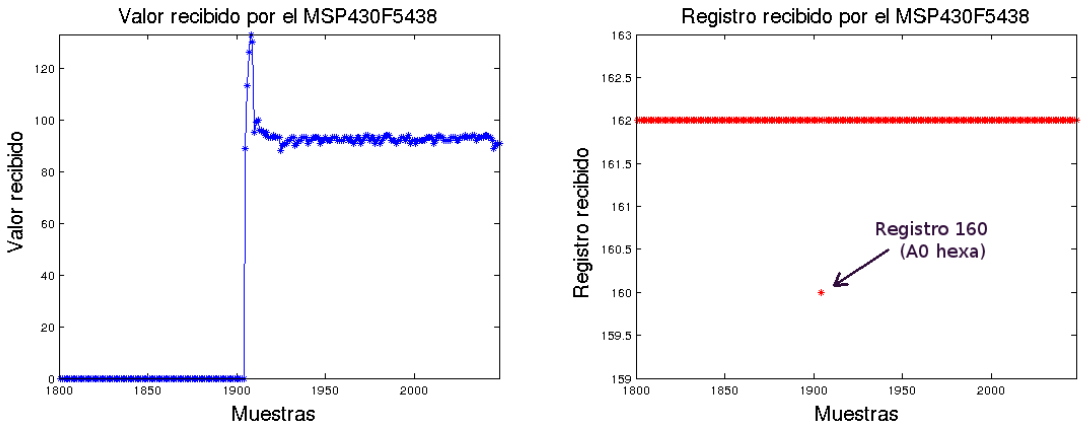
El MSP430F5438 (ver figura 1.13) posee un puerto $i^2c$ habilitado para su utilización. Se configurara como esclavo con la dirección 0x68 y se guardan en una variable los comandos recibidos en el bus. De este modo se logra independizarse de los problemas que pueda introducir el sniffer y se obtiene exactamente el comando transmitido al esclavo presente en dicha dirección.

Figura 1.13: MSP430F5438

Luego, analizando la variable donde se guardan los comandos, es posible obtener información más concluyente. En la figura 1.14 se muestran los resultados obtenidos al adquirir datos en un arranque, mientras que en la figura 1.15 se muestran los resultados obtenidos al adquirir datos al frenar.

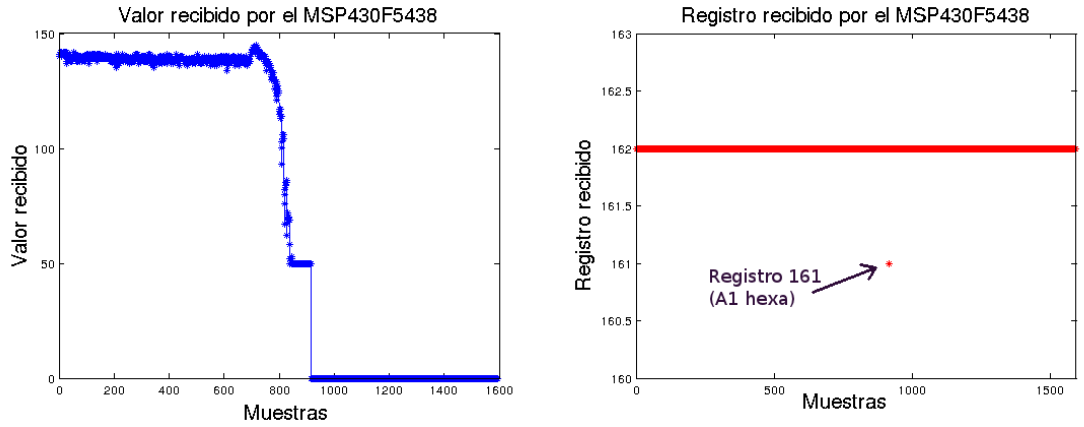
Estas últimas dos figuras son las corroboraciones más concluyentes que se obtuvieron. Ratifica y deja bien en claro lo antes dicho sobre los registros: **para el arranque se escribe en la dirección 0xA0 del esclavo y para el frenado se escribe en la dirección 0xA1.**

A su vez sirve de confirmación para los errores del sniffeado detectados en la sección 1.4.




(a) Valores enviados a los motores (b) Registro al que se envía el valor

Figura 1.14: Arranque detectado por el MSP430F5438




(a) Valores enviados a los motores (b) Registro al que se envía el valor

Figura 1.15: Frenado