Toma el control de tu ladrillo MINDSTORMS (5)

por Oton Ribić

Finalmente, el conocimiento combinado que hemos reunido a lo largo de los cuatro artículos anteriores debería permitirnos ahora alcanzar el objetivo final por el que nos esforzamos: enviar comandos a nuestros ladrillos inteligentes EV3 por Bluetooth. Por supuesto, el tipo más práctico y más frecuentemente utilizado es el de realizar movimientos motores controlados, y eso es exactamente lo que vamos a hacer.

Ensamblar los componentes del mensaje

Sin intentar explicar de nuevo cada uno de los componentes del mensaje de comando, sigamos adelante y trabajemos en un ejemplo. Supongamos que queremos girar el motor conectado al puerto nº 2 durante una vuelta y media (540°) en dirección positiva a tres cuartos de la velocidad máxima, es decir, a la velocidad de 75%. Y construir el mensaje poco a poco.

1) Cabecera. En primer lugar tenemos que establecer un encabezado del mensaje, como se explicó en los artículos anteriores. Como esta vez no vamos a utilizarlo para nada "elegante", comenzaremos con cinco ceros, es decir, cinco bytes con valores de cero.

2) Dirección. Ahora queremos establecer la dirección del motor para este comando. Esto se hace añadiendo los bytes 167, 0, luego el número del motor que es 2 en nuestro caso, y finalmente 1 para avanzar, o 63 para la dirección inversa. Así que tenemos 167, 0, 2, 1 aquí.

3) Valores de movimiento. Luego viene la parte principal, la instrucción de movimiento en sí misma. Comienza con 174, 0, luego continúa con el número del motor, otra vez 2. Luego está la velocidad: es el valor 129 seguido de la velocidad deseada que da 129, 75. Luego está el valor de rampa de aceleración, para el cual podemos usar el cero codificado en cinco bytes, que es 131, 0, 0, 0, 0. Luego, finalmente el ángulo que pretendemos girar, 540 codificado en una estructura de cinco bytes, que es 131, 28, 2, 0, 0. Luego el valor de rampa de aceleración, que es nuevamente cero, resultando en 131, 0, 0, 0, 0. Finalmente, el parámetro que dice frenar una vez completado, que es un simple byte final 1 para esta sección.

4) Instrucción para comenzar. Habiendo establecido todos los parámetros, ahora añadiremos la instrucción para que el motor empiece realmente este trabajo meticulosamente preparado. Es bastante más simple: 166, 0, seguido del número del motor, que es 2.

5) Espere a que termine. Si queremos que el EV3 realice el movimiento completamente antes de pasar al siguiente, añadiremos ahora 170, 0, 15, que es esencialmente "esperar a que termine". Sin él, la siguiente instrucción comenzará mientras el motor gira, lo cual puedes querer, o quizás evitar.

6) Longitud. Contemos finalmente la longitud del mensaje que hemos reunido: contiene 36 bytes. Así que ponemos 36,0 delante de él.

Si todo salió bien, obtuvimos el siguiente mensaje y estructuras.

36, 0, 0, 0, 0, 0, 0, 167, 0, 2, 1, 174, 0, 2, 129, 75, 131, 0, 0, 0, 0, 131, 28, 2, 0, 0, 131, 0, 0, 0, 0, 1, 166, 0, 2, 170, 0, 15

Y cada parte corresponde a su propio propósito en la combinación de colores: Longitud, Cabecera, Dirección, Especificar el motor, Velocidad, Rampa de aceleración, Cantidad a girar, Rampa de aceleración, Frenar cuando se haya completado, Comenzar a girar, Esperar a que se complete.

Envío del mensaje

En este punto estamos listos para "disparar" estos últimos 38 bytes al ladrillo del EV3 a través del puerto serie virtual encapsulado en el protocolo Bluetooth. Como se ha explicado en los artículos anteriores, el ladrillo del EV3 acusará recibo de este mensaje 0 cuando termine. Podemos entonces enviar más mensajes si lo deseamos.

Si desea ir a las longitudes completas e implementar la codificación de los valores en estructuras de cinco bytes, estos son los valores para cada byte. Esto supone que el ampersand (&) se utiliza como un operador AND binario, y >> un desplazamiento binario hacia la derecha por un número determinado de lugares. (Esto funciona "tal cual" en Python).

```
byte1 = 131
byte2 = value & 255
byte3 = (value >> 8) & 255
byte4 = (value >> 16) & 255
byte5 = (value >> 24) & 255
```

Por supuesto, esto funciona solo con números enteros, pero de todos modos esa es la suposición subyacente para todo este tutorial. Si va a trabajar con divisiones de números, siempre es una precaución prudente redondear los números que ingresan este cálculo en números enteros.

La rotación de varios motores a la vez se hace simplemente construyendo y disparando mensajes independientes, uno para cada motor, y no habilitando en ellos el parámetro de "espera de finalización". De esa manera todas las rotaciones se iniciarán y procederán simultáneamente.

Sin embargo, en este nivel no hay una manera simple y a prueba de tontos de controlar la posición de cada motor en cada momento particular. Es decir, si arrancamos un motor a toda velocidad y el otro a media velocidad, es sólo una suposición optimista que, en cualquier momento posterior, el último habrá hecho exactamente la mitad del movimiento del primero. Si buscas movimientos simultáneos muy precisos, por ejemplo, para dibujar una línea diagonal con un plotter X-Y, considera la posibilidad de dividir los movimientos en segmentos más pequeños, y utilizar mucha reducción para limar aún más las diferencias entre los motores. Por supuesto, el precio a pagar en este caso es una ejecución más lenta, así que tendrás que encontrar la fórmula que te funcione mejor.
#