Capítulo 2. Multiplexor de Sensores

2.1 Introducción al Multiplexado.

De acuerdo con Wayne Tomasi el Multiplexado es la transmisión de información (en cualquier forma) de más de una fuente a más de un destino a través del mismo medio (o instalación) de transmisión. Las transmisiones pueden suceder en la misma instalación pero en tiempos diferentes. De hecho, hay varios dominios en los que se puede hacer el multiplexado, incluyendo espacio, fase, tiempo, frecuencia y longitud de onda.[9]

Así pues el multiplexor de entradas que se piensa construir en esta tesis se debe entender como un dispositivo que trabaje conectando múltiples sensores a uno solo de los puertos de entrada. Dado que se habilita un sensor a la vez, se trata de un multiplexor que utiliza la división de tiempo para realizar la lectura de diferentes dispositivos. Michael Gasperi, uno de los muchos entusiastas de LEGO[10], afirma en su artículo de cómo crear un sensor de color que la mayoría de las computadoras emplean multiplexores en el hardware de conversión análoga a digital para ahorrar espacio y costo.

El RCX tiene un multiplexor, pero sólo tres de sus entradas están disponibles para el usuario, ya que el RCX utiliza uno de los otros canales internamente para leer su propio nivel de batería.

2.2 Proceso de sensado del RCX

Como primer paso de esta tesis se decidió que el multiplexor de sensores fuera el primer circuito en diseñarse, ya que éste sería utilizado por los demás circuitos que se construyeran, de tal modo que se buscaría después que los sensores se adecuaran al multiplexor construido, en lugar de que se tuvieran que hacer dos o más tipos de multiplexores para los diferentes sensores.

El RCX tiene tres entradas y tres salidas. En las entradas, las cuales están numeradas del 1 al 3, se conectan los sensores. Mientras que en las salidas nombradas como A, B y C pueden conectarse motores y luces que operen en un rango de 0V a 9V de voltaje y que no demanden más de 0.5A de corriente. Los puertos de entrada y salida, así como su nomenclatura se pueden apreciar en la Figura 1.5 del capítulo anterior.

Resulta obvio que para construir un multiplexor de sensores para el RCX primero se debía entender como se lleva a cabo el proceso de sensado a través de sus 3 puertos de entrada. De este modo fue que se repitió un experimento realizado previamente por Michael Gasperi.

Gasperi programó a través del Spirit.OCX[1] con ayuda de un programa de Visual Basic que le permitía seleccionar directamente el tipo de sensor y verificar las lecturas por “poleo” (*polling*). Posteriormente usando un osciloscopio midió directamente sobre los puertos el tiempo y los voltajes y observó que el RCX tiene dos modos de sensado:

[1] Control Spirit.ocx

Cuando se instala el software que acompaña a LEGO® MindStorms™ (versiones 1.0 ó 1.5) o LEGO Technic CyberMaster automáticamente se instala en el ordenador un control ActiveX llamado SPIRIT.OCX. Este control permite controlar el RCX desde diferentes entornos de programación: Visual Basic, Visual C++, Delphi, Visual Java++. Sin embargo, la versión 2.0 de LEGO MindStorms no contiene ya este control.

http://www.donosgune.net/2000/gazteler/prg_leng/VB.htm

pasivo y alimentado. Finalmente conectó diferentes valores de carga en los puertos de acuerdo con la Tabla 2.A y midiendo la caída de voltaje en las terminales determinó el valor de las resistencias mostradas en los circuitos equivalentes de la Figura 2.3 y la Figura 2.4.

Tabla 2.A. Equivalencias de los distintos modos de sensado.

Voltaje (Volts)	Valor Raw	Resistencia (Ohms)	Modo Luz	Modo Temperatura (°C)	Modo Contacto
0.0	0	0	---	---	1
1.1	225	2816	---	70.0	1
1.6	322	4587	100	57.9	1
2.2	450	7840	82	41.9	1
2.8	565	12309	65	27.5	0
3.8	785	32845	34	0.0	0
4.6	945	119620	11	-20.0	0
5.0	1023	Inf.	0	---	0

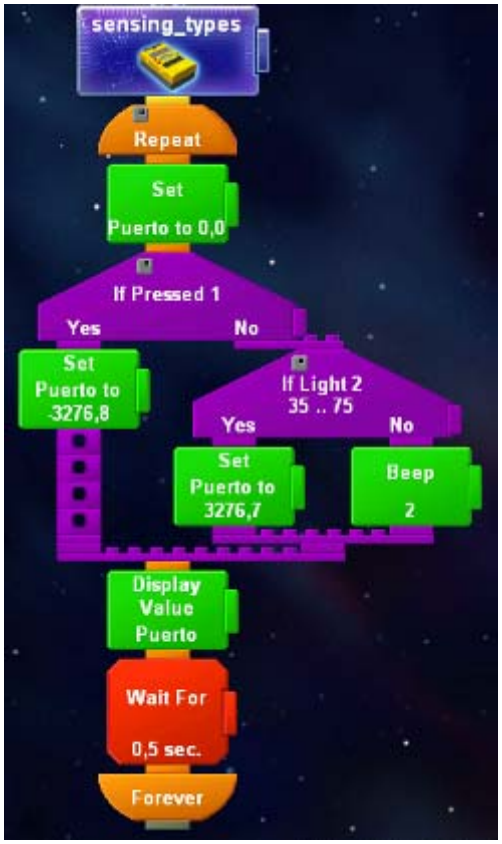


Figura 2.1. Programa utilizado para verificar los 2 tipos de sensado.

En nuestro caso sólo programamos el RCX mediante el RIS 2.0 para que supusiera conectado un sensor de contacto en el PUERTO_1 y un sensor de luz en el PUERTO_2. El programa se muestra en la Figura 2.1. Posteriormente se conectaron las terminales al osciloscopio con ayuda de cables caimán-caimán y BNC-pin como se ve en la Figura 2.2 y se comprobaron los dos tipos de sensado del RCX.

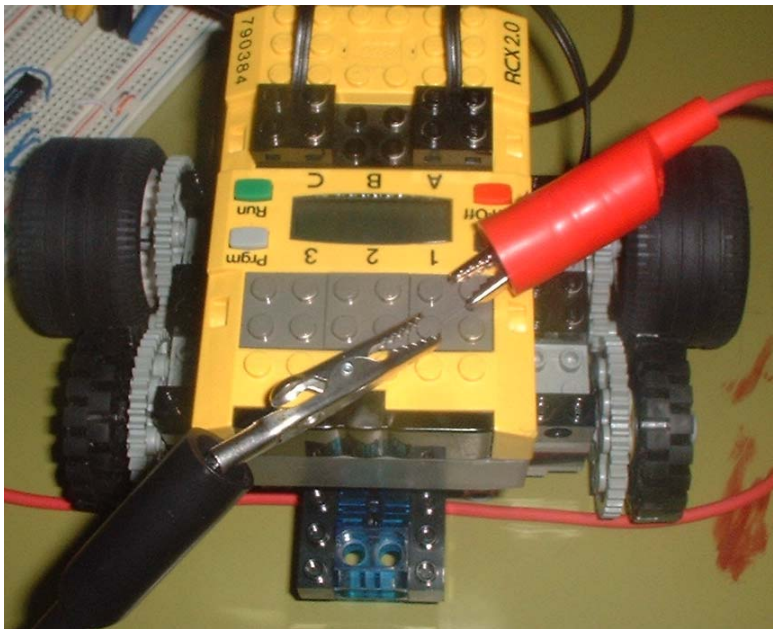


Figura 2.2. Conexión del PUERTO_1 mediante cables Caimán-Caimán

2.2.1 Sensado pasivo.

En el modo pasivo, el RCX mide un voltaje en la entrada a través de un convertidor análogo digital con resolución de 10 bits. Para evitar un corto circuito hay una resistencia de 10k. El valor de la resistencia de 10k se determinó a partir de los resultados de la Tabla 2.A.

Ejemplos de sensores de tipo pasivo son los sensores de contacto y los sensores de temperatura. La Figura 2.3 muestra el circuito equivalente del RCX cuando se programa para leer este tipo de sensores, así como la polaridad de las terminales de Entrada. (Puertos 1, 2 y 3).

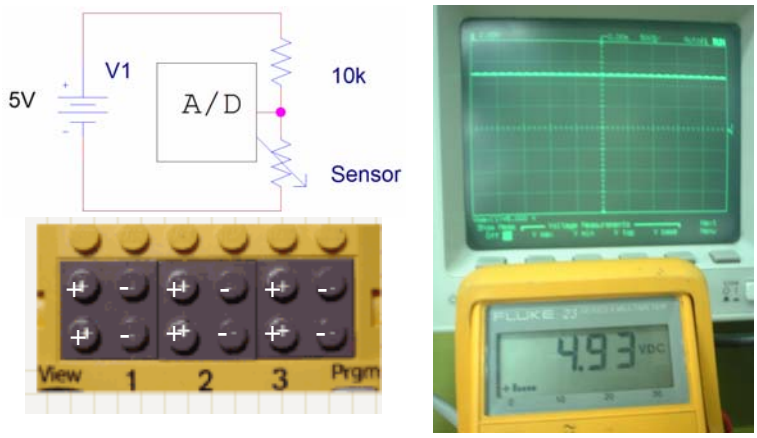
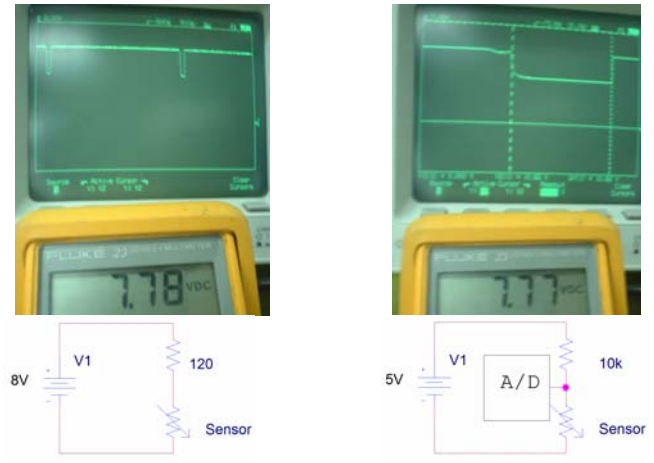


Figura 2.3. Circuito equivalente y polaridad de los puertos de entrada.

2.2.2 Sensado alimentado.

En el modo alimentado, el RCX aplica aproximadamente 8V a la entrada durante 3ms y después lee el valor de la misma forma que lo hace en el modo pasivo durante un pequeño periodo de 0.1ms. Qué tanto se aproxime al valor de 8V depende del estado de las baterías. Ejemplos de sensores que necesitan alimentación incluyen al sensor de luz y al sensor de rotación. La Figura 2.4 muestra los circuitos equivalentes del RCX durante el modo de sensado alimentado.



a) Alimentado por 3ms **b)** Sensado por 0.1ms

Figura 2.4. Señal en las terminales de entrada del RCX en modo activo

En la Figura 2.4.a se observa el pequeño tiempo de 0.1ms, en que el voltaje es de 5V, como un pequeño impulso invertido, mientras que la Figura 2.4.b muestra una

ampliación en la escala de tiempo de este pequeño instante y por eso se aprecia como un escalón. Es importante notar que la lectura del voltímetro no es de 5V en la Figura 2.4.b, debido a que ni este instrumento ni nosotros a simple vista somos capaces de apreciar el pequeño instante (100µs) en que el voltaje es de 5V.

2.3 Construyendo el Multiplexor.

Se debe mencionar que muchos autores entusiastas del RCX MINDSTORMS™ de LEGO® han ideado algunas alternativas para solucionar el problema de tener un número reducido de puertos, tanto de entrada como de salida. Muchos de ellos publican sus ideas en sus páginas personales o en páginas de asociaciones dedicadas a promover el desarrollo de nuevas creaciones utilizando el kit.

Algunas de estas alternativas incluyen multiplexores basados en relevadores que utilizan uno de los puertos de salida para cambiar entre las diferentes entradas o salidas. Sin embargo, esto implica el uso de dos puertos: uno de entrada y otro de salida para poder realizar el multiplexado y el sensado.

Para el caso de sensores de contacto hay unos arreglos de conexión bastante ingeniosos que permiten conectar 2 o más de ellos a un mismo puerto y saber cual ha sido oprimido. Otra opción más socorrida, es conectar los interruptores a través de resistencias de diferente valor para generar 4, 8 o 16 diferentes valores de voltaje dependiendo de si se requieren multiplexar 2, 3 o 4 sensores de este tipo, respectivamente. Lamentablemente, con esta solución no se podrían multiplexar sensores de luz u otros sensores que utilicen el sensado activo, además de que el número de resistencias utilizadas aumenta muy rápidamente conforme se requieren más entradas.

El siguiente circuito mostrado en la Figura 2.5 se basó en el diseño de Gasperi para su sensor de color.

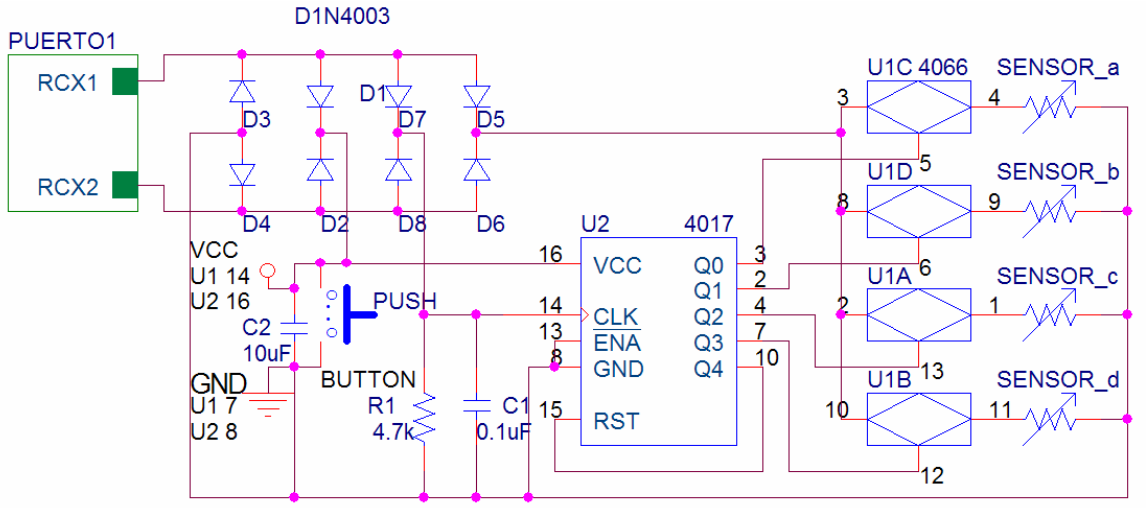


Figura 2.5. Circuito Multiplexor

Dicho circuito aprovecha el cambio de voltaje debido a los dos modos de sensado para cerrar un interruptor analógico a la vez, permitiendo que cuatro sensores resistivos compartan una misma entrada del RCX. Algunos buenos ejemplos de sensores resistivos son foto-celdas, termistores y potenciómetros. Los interruptores también pueden ser usados ya que son casos extremos de sensores resistivos de acuerdo con la Tabla 2.B.

Tabla 2.B. Interruptor como caso extremo de sensor resistivo.

Caso	Valor de R
Interruptor Cerrado	0
Interruptor Abierto	∞

Los diodos D_1 a D_4 en conjunto con el capacitor C_2 constituyen la fuente de alimentación. El arreglo de rectificador de onda completa permite que el conector del RCX tenga cualquier polaridad. Esto sirve como medida de seguridad para que no ocurra ningún corto circuito ni se invierta la polaridad en caso de que por error se conecte de forma incorrecta.

Los cables de conexión usados por los dispositivos eléctricos (sensores y motores) parecen tener cuatro terminales diferentes, sin embargo, vale la pena aclarar que están conectadas internamente en pares como se muestra en la Figura 2.6, de modo que sólo se tienen dos terminales una positiva y otra de tierra. En otras palabras, sólo se tienen dos hilos, y por ellos se debe proveer de energía a dichos dispositivos y también en el caso de los dispositivos sensores es por estos mismos hilos que se debe recibir la señal correspondiente a la variable física de interés.

Figura 2.6. Conector LEGO.

En la Figura 2.6 también se aprecia que la parte metálica en la parte superior no está presente alrededor de todo el borde, de hecho, el corte metálico tiene un ingenioso diseño el cual impide que las terminales positiva y tierra estén en corto circuito entre sí, no importando que se haya cometido un error al realizar la conexión no hay posibilidad de corto circuito. De forma que sólo queda el problema de invertir la polaridad; es para evitar esto que se conecta el puente de diodos.

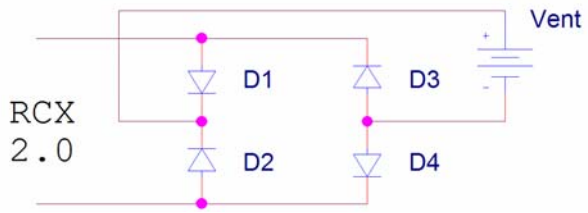


Figura 2.7. Circuito para evitar el problema de polaridad.

El capacitor C_2 por su parte, tiene ese valor de 10µF con el fin de mantener los dispositivos U_1 y U_2 energizados con una corriente constante durante el periodo de 0.1ms en que el RCX alterna a 5V para realizar la medición mediante el convertidor A/D.

El circuito integrado U_2 es un contador digital de década que va desde Q_0 hasta Q_3 y luego se reinicia al interconectar el pin Q_4 y el pin de reinicio o *"reset"*. (Ver Apéndice B para mayores detalles de este integrado).

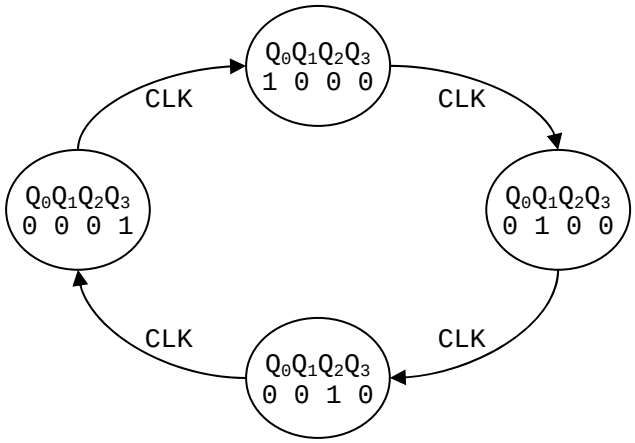


Figura 2.8. Diagrama de Estados del integrado U2.

La resistencia R_1 y el capacitor C_1 forman un circuito de descarga con una constante de tiempo de 470µs (filtra los pulsos del sensado pasivo) y se conectan al pin de reloj (CLK) de U_2. El cambio de estado del contador U_2 en este circuito se realiza precisamente al aprovechar las transiciones de voltaje que el RCX puede hacer entre 2 niveles de voltaje, derivados de los dos tipos de sensado explicados previamente. El voltaje en la entrada de reloj permanece en alto durante el modo de sensado alimentado, pero cambia a bajo durante el periodo de tiempo más largo en el que el modo de sensado cambia a modo pasivo. Cuando el sensor vuelve a cambiar al modo alimentado, el circuito crea la transición positiva necesaria para cambiar al siguiente estado en el integrado U_2.

El circuito integrado U_1 (Apéndice C) contiene 4 interruptores analógicos usados para conectar un sensor al RCX a la vez. En el esquema original, U_{1A} proveía de

información al RCX para sincronización, sin la cual el RCX no puede saber cual sensor es el que está conectado ya que el RCX no puede saber el estado presente de U_2.

Vale la pena aclarar este punto con un ejemplo: supongamos que recién encendimos el RCX después de varios días de no ocuparlo, por lo tanto los capacitores deberán estar descargados y dado que no ha habido energía, el circuito U_2 se ha reiniciado automáticamente al estado Q_0. De modo que el interruptor correspondiente al SENSOR_a será el único activo. Ahora procedemos a presionar el botón verde (RUN) para iniciar la ejecución de un programa de ejemplo cuyo código se muestra a continuación.

Dicho código fue creado utilizando el software Bricx Command Center que permite hacer el cambio en el modo de sensado de un puerto de entrada mientras ocurre la ejecución del programa (*on the fly*), cosa que no es permitida usando el RIS 2.0.[10]

```
task main()
{
// Variables requeridas para los puertos del MUX
     int SENSOR_a=8888;
     int SENSOR_b=8888;
     int SENSOR_c=8888;
     int SENSOR_d=8888;

     SetSensor(SENSOR,SENSOR_LIGHT); //se inicializa el MUX en el canal a
     while(true)                  //Repeat forever
     {
          multiplexor(SENSOR_a, SENSOR_b, SENSOR_c, SENSOR_d);
          Wait(5);
      }    //Fin Repeat forever
}     //Fin task main()
```

Figura 2.9. Programa de ejemplo para el uso del Multiplexor

Las primeras 4 instrucciones definen e inicializan las variables que almacenarán el resultado de los 4 sensores multiplexados, dichas variables se nombraron de acuerdo con el esquemático de la Figura 2.5. Posteriormente se activa el Multiplexor al definir el SENSOR, como un sensor de luz y se está listo para leer el canal correspondiente al

SENSOR_a. Lo cual es correcto ya que el estado presente en U_2 es Q_0. En el código anterior la palabra SENSOR se debe cambiar por las opciones de la Tabla 2.C dependiendo de si el multiplexor está conectado al PUERTO 1, 2 o 3.

Tabla 2.C. Palabras usadas para referirse a los 3 puertos del RCX.

Multiplexor conectado a	Usar la palabra
PUERTO_1	SENSOR_1
PUERTO_2	SENSOR_2
PUERTO_3	SENSOR_3

Después se entra a un ciclo que siempre se repite (el equivalente a un *"repeat forever"* del RIS 2.0) e inmediatamente después se llama a una función `multiplexor` cuyo código se muestra a continuación.

```
#define SENSOR SENSOR_1 //El MUX se supone conectado al puerto 1
#define ESPERA 1        //10ms de espera para estabilizar la lectura

void multiplexor(int &SENSOR_a, int &SENSOR_b, int &SENSOR_c, int &SENSOR_d)
{
     Wait(ESPERA);              //Espera a que la lectura se estabilice
     SENSOR_a=SENSOR;           //Sensor conectaodo al puerto 'a'

     SetSensor(SENSOR,SENSOR_TOUCH);
     SetSensor(SENSOR,SENSOR_LIGHT);//     cambio al PUERTO_b

     Wait(ESPERA);              //Espera a que la lectura se estabilice
     SENSOR_b = SENSOR;         //Sensor conectado al puerto 'b'

     SetSensor(SENSOR,SENSOR_TOUCH);
     SetSensor(SENSOR,SENSOR_LIGHT);//     cambio al PUERTO_c

     Wait(ESPERA);              //Espera a que la lectura se estabilice
     SENSOR_c = SENSOR;         //Sensor conectado al puerto 'c'

     SetSensor(SENSOR,SENSOR_TOUCH);
     SetSensor(SENSOR,SENSOR_LIGHT);//     cambio al PUERTO_D

     Wait(ESPERA);              //Espera a que la lectura se estabilice
     SENSOR_d = SENSOR;         //Sensor conectado al puerto 'd'

     SetSensor(SENSOR,SENSOR_TOUCH);
     SetSensor(SENSOR,SENSOR_LIGHT);//     reinicio al PUERTO_a
}
```

Figura 2.10. Código de la función multiplexor.

Lo primero que hace la función es esperar durante 10ms para que el circuito alcance el valor de voltaje deseado (8V) y la lectura del sensor sea la correcta, cuando termina la espera, almacena el valor presente en la variable correspondiente, define el SENSOR como un sensor de contacto con lo que el voltaje cae a 5V pero inmediatamente lo vuelve a activar como un sensor de luz, con lo que se crea la transición necesaria para que U_2 cambie al estado Q_1 nuevamente se espera 10ms para asegurar que la lectura sea correcta y almacena el valor correspondiente al SENSOR_b. Estas instrucciones se repiten en la función multiplexor y es mediante los cambios de voltaje que el estado de U_2 cambia a Q_2, Q_3, Q_0, Q_1, Q_2… y así sucesivamente.

Ahora supongamos que está corriendo el programa y se presiona el botón de STOP cuando Q_2 era el estado presente en U_2 con lo que U_{1A} de la Figura 2.5 sería el único circuito cerrado y la lectura en ese momento se correspondería con la variable del SENSOR_c, lo cual es correcto. Si se aprieta nuevamente el botón de RUN el RCX volverá a correr el programa desde el principio, por lo que al almacenar la lectura en una variable almacenará el valor del SENSOR_c en la variable SENSOR_a, lo cual es incorrecto.

Gasperi había propuesto utilizar dos valores de resistencias diferentes como se muestra en la Figura 2.11. El interruptor U_{1A} se cierra cuando la cuenta de U_2 activa Q_0, conectando los diodos D_5 y D_6 mediante R_1 a tierra. La lectura creada de esta manera será siempre 100. Como los otros 3 sensores están conectados a través de R_2 que es mayor que R_1, la lectura máxima de éstos sería un valor cercano a 100, que en el caso de las pruebas fue de alrededor de 97.

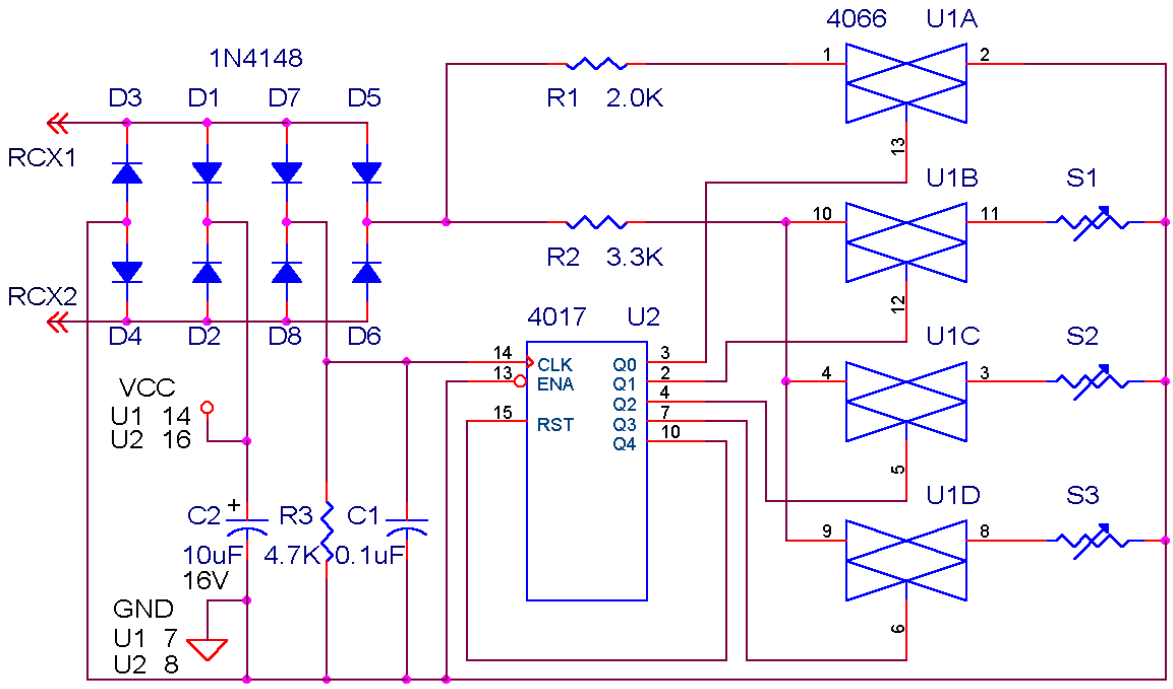


Figura 2.11. Circuito Multiplexor ideado por Michael Gasperi.

Lamentablemente con esta solución se impedía por un lado que los sensores S_1, S_2 y S_3 variaran en el rango completo (0 a 100) y por otro lado desperdiciaba uno de los interruptores analógicos del circuito integrado U_1 con lo que el multiplexor realizaba multiplexado de 3 a 1 en lugar de 4 a 1 como se logró en nuestro diseño.

La solución fue agregar el PUSH BUTTON en paralelo con el capacitor C_2 el cual se debe presionar preferentemente cuando el RCX está apagado. Al presionar el botón se forma un corto circuito que descarga el capacitor, como el RCX también está apagado el circuito U_2 se queda sin energía y se reinicia al estado Q_0.

Una forma muy fácil de verificar que U_2 ha regresado al estado Q_0 es conectar un sensor de luz al PUERTO_a del multiplexor antes de presionar el botón verde, de este modo el foco rojo del sensor de luz debe encender como se muestra en la Figura 2.12.

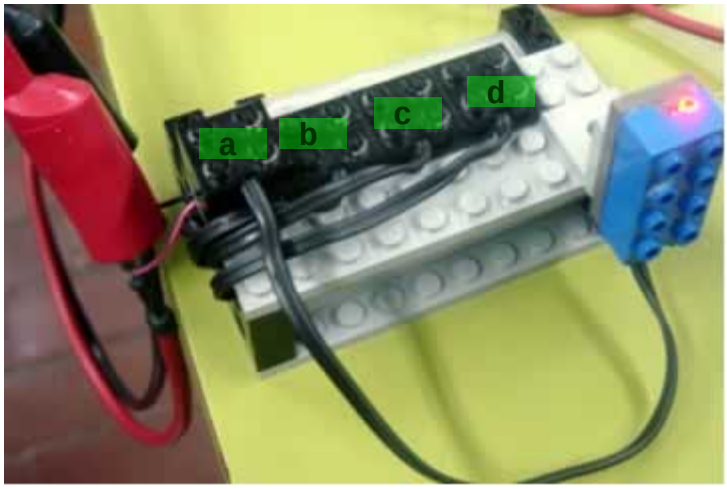


Figura 2.12. Imagen del Multiplexor realizado.

Una vez que se presiona el botón verde si se conectan los canales 1 y 2 del osciloscopio con ayuda de los cables caimán-caimán se puede observar en la pantalla la gráfica de la Figura 2.13.

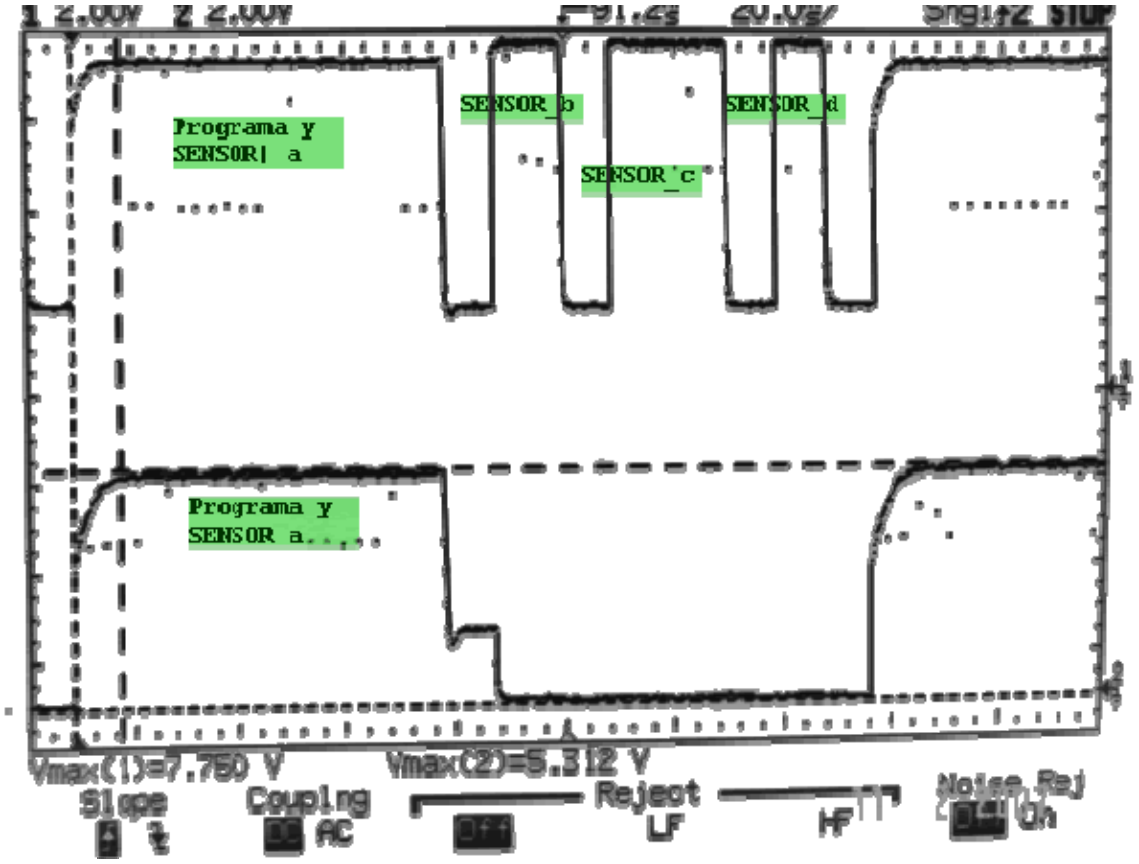


Figura 2.13. Funcionamiento del Multiplexor.
1) PUERTO_1 del RCX 2) PUERTO_a del MUX.

La conexión se ha realizado de la siguiente forma: el canal 1 se conecta directamente al PUERTO_1 del RCX, mientras que el canal 2 se conecta al PUERTO_a del multiplexor construido (MUX) esto se muestra mejor en la Figura 2.14.

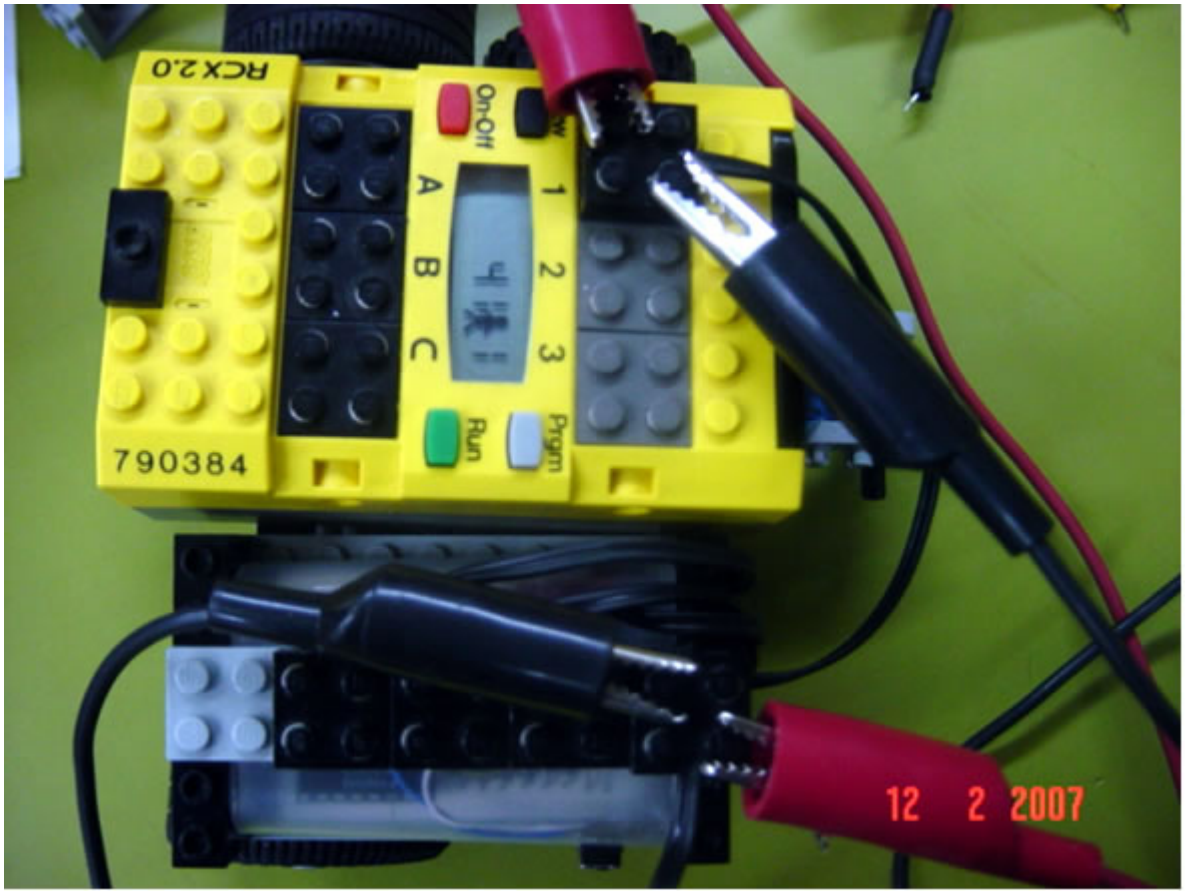


Figura 2.14. Conexión del multiplexor.

En el canal 1 se pueden observar los niveles de voltaje derivados de alternar entre los dos diferentes tipos de sensado y se puede indicar que ranura de tiempo corresponde a cada uno de los puertos del multiplexor, la ranura más ancha corresponde a la ejecución de las instrucciones de programa y durante este tiempo está activo el SENSOR_a, en el caso del programa de ejemplo simplemente se trata de una espera de 50ms (`Wait(5);`).

Después se aprecian otros tres pulsos, de aproximadamente 10ms cada uno, que corresponden a los periodos de tiempo en que están activos los otros puertos del

multiplexor. Lo anterior se hace más evidente al revisar el canal 2 en la parte inferior de la Figura 2.13 aquí se observa que sólo existe voltaje durante la ranura de tiempo indicada como SENSOR_a, con lo que se verifica la correcta acción de multiplexado por división de tiempo.

Debido a que el almacenamiento de las variables se lleva a cabo justo después de que el modo de sensor se ha definido como SENSOR_LIGHT, las variables almacenarán la lectura del sensor en valores de 0 a 100 de acuerdo a la Tabla 2.A no importando cual sea el verdadero tipo de sensor que se conecte en cada puerto del multiplexor, esto es importante mencionarlo ya que un sensor de contacto conectado al multiplexor dará una lectura de 0 si el botón no está siendo presionado mientras que almacenará 100 en la variable correspondiente si está siendo presionado.

Otra cuestión a considerar es que la lectura se hace en un instante dado, a esto se le conoce como lectura por “poleo” (*polling*), y durante otro tiempo el sensor permanece apagado o ejecutando otras instrucciones, de modo que si ocurre un cambio en la señal correspondiente a un sensor conectado al multiplexor durante un instante en que el sensor está apagado, dicho cambio no será almacenado. Por esto último es que se ha buscado que el periodo de tiempo en que cada uno de los puertos del multiplexor este apagado sea tan breve como sea posible. En el ejemplo los PUERTOS b, c y d están activos durante 10ms y apagados aproximadamente 80ms, mientras que el PUERTO_a está activo durante 50ms y apagado 40ms. En todos los casos un cambio en la señal que tenga una vigencia mayor a 80ms siempre será detectado, mientras que tendría que tener una vigencia mucho menor para que no pudiera ser apreciado el cambio.

Esto no es tan crítico en el caso del RCX ya que se pretende su uso para medir variables que difícilmente cambian tan rápidamente como es la intensidad de luz y

temperatura en un cuarto o la presencia de un objeto delante del robot en el caso de los sensores de proximidad y contacto. Sin embargo vale la pena recomendar que si las instrucciones a ejecutar después de que se ha hecho el sensado son muchas o si se pretende sensar un ambiente rápidamente cambiante la función multiplexor sea llamada varias veces a lo largo de la rutina principal (`task (main)`).