INGENIERÍA MECATRÓNICA

DI_CERO

DIEGO CERVANTES RODRÍGUEZ

LENGUAJE C, PROGRAMACIÓN MICROCONTROLADORES: ATMEGA328P (ARDUINO)

MICROCHIP STUDIO

Interrupciones

Contenido

¿Qué es una interrupción?

Interrupción se refiere a una acción del microcontrolador donde cuando ocurre un evento se ejecuta una acción, no importando el valor del Program Counter, que es un puntero o registro especial que apunta a cada línea de código (que está almacenada en la memoria FLASH) para ejecutar cada parte del programa. **Las interrupciones son causadas por eventos externos y se ejecuta de forma asíncrona, a diferencia de las subrutinas que son síncronas y no se detonan por eventos externos**. Un ejemplo de esto puede ser el microondas, donde no importando que esté pasando en la ejecución de la máquina, cuando se abra la puerta, deja de hacer lo que sea que esté haciendo.

Subrutinas

Recordemos que por ejemplo cuando se ejecuta una función, al esta ser llamada, en el PC (Program Counter) se cargará el valor de parte del programa donde se encuentra la línea de código donde empieza la función (registro de la memoria FLASH) y a su vez se cargará el valor de la parte del código donde se encontraba el programa antes de brincar a la otra parte del código en una memoria llamada Stack Poiner o Pila de programa, esto es llamado una subrutina. Un ejemplo de esto puede ser un delay, que ejecuta una subrutina para manejo de tiempos.

Interrupciones

Las interrupciones se ejecutan a través del algo llamado RETI que es una línea de código que hace que el programa retorne a la línea de código en la que se quedó antes de ejecutar la interrupción (trabaja más o menos como la instrucción return), las interrupciones además poseen algo llamado **ISR o Interruption Service Routine**, este lo que hace es lo siguiente:

- **Interruption:** Cuenta con un bit de status llamado GI (General Interruption) o switch general de interrupciones, que indica si la interrupción va a estar activada o no para que se ejecute cuando ocurra el estímulo que la detona.
- El número de interrupciones que puede tener el microcontrolador depende del número de puertos donde se puedan conectar sus periféricos.

Las interrupciones se dividen en 2 tipos:

- **NMI (Non Maskable Interrupt):** Se les llama interrupciones no enmascarables, se refiere a cuando la interrupción no puede ser ejecutada, ya que el registro de STATUS que posee 8 bits y donde en su bit de índice 7 se encuentra el GI cuenta con la instrucción de no dejar que se ejecute ninguna interrupción, porque cuando tiene valor de 1 lógico no está activada ninguna interrupción y cuando tiene valor de 0 lógico si lo están, pero en el caso de las NMI, el GI tiene 1 lógico como valor.
    - **Registro de estado (STATUS) o chismoso:** Es un registro con banderas que dejan constancia de algunas condiciones que se dieron en la última operación realizada, son

los que toman en cuenta acarreos en la suma o resta binaria y podrán ser tomadas en cuenta en operaciones posteriores.
    - El registro de STATUS me permite pasar de un banco de registros a otro.
- **MI (Maskable Interrupt):** Tienen que ver con el GI, que es un switch que representa la S de ISR y se refiere a que, si la interrupción va a estar activada o no, para que reaccione cuando el estimulo externo llegue al programa, en el caso de las MI, el GI tiene 0 lógico como valor, por lo cual si puede ejecutar interrupciones.
    - Algo interesante a saber de cuando se ejecuta una interrupción, es que guarda la dirección de memoria FLASH a la que debe volver ya que se haya dejado de ejecutar la interrupción para poder seguir con la ejecución del programa, pero a su vez mientras ejecuta la interrupción, pone en 1 el valor lógico del GI, por lo que ya no se puede ejecutar otra interrupción mientras se corre la actual.

El microcontrolador cuenta con algo llamado vector de interrupciones, el cual

Una buena práctica de programación es utilizar las siguientes instrucciones:

- **CLI**: Cuando se activa una CLI interruption, automáticamente el bit 7 del status register que es GI (General Interruption) se convierte en 0 lógico, lo que convierte a las interrupciones en MI (Maskable Interrupt):
    - Este ***se activa antes de la Inicialización de máquina***, osea cuando se declara que puertos son entradas o salidas dentro del método main y solamente se hace cuando vaya a utilizar interrupciones en el programa.
        - X = Don't care, osea que no importa el valor de este bit.

Índice 7	Índice 6	Índice 5	Índice 4	Índice 3	Índice 2	Índice 1	Índice 0
GI = 0	Z	C	C	X	X	X	X

- **SEI**: Cuando se activa una SEI interruption, automáticamente el bit 7 del status register se convierte en 1 lógico, lo que convierte a las interrupciones en NMI (No Mask Interruption):
    - Este ***se activa después de la Inicialización de máquina y ejecución del programa y declaración de interrupciones***, osea cuando se declara que puertos son entradas o salidas dentro del método main y solamente se hace cuando vaya a utilizar interrupciones en el programa.
        - X = Don't care, osea que no importa el valor de este bit.

Índice 7	Índice 6	Índice 5	Índice 4	Índice 3	Índice 2	Índice 1	Índice 0
GI = 1	Z	C	C	X	X	X	X

Por lo tanto, la arquitectura del programa es la siguiente:

- **CLI: Activación de interrupciones con el bit GI con valor 0 lógico (MI).**
- Programación de máquina y configuración de interrupciones con los registros EIMSK y EICRA, además de declaraciones de PCI.
- **SEI: Desactivación de interrupciones con el bit GI con valor 1 lógico (NMI).**
- Ejecución del programa.

Esto se usa más que nada con contadores, incluidos dentro de un registro llamado TOV (Timer Overload), existen varios de estos dentro del microcontrolador y depende de la frecuencia del oscilador el tiempo de conteo.

Por lo tanto, la diferencia más grande entre subrutina e interrupción es que la subrutina empieza cuando sea y es síncrona y la interrupción es solamente desencadenada por un evento externo y puede ser ejecutada de forma asíncrona.

En específico en el mcirocontrolador ATMEGA328P, cada uno de sus 3 puertos B, C y D cuenta con un puerto especial para interrupciones llamado **PCI** (Pin Change Interruption) y este lo que hace es indicar cuales de los pines de cada puerto activan una interrupción.

- **PCIB**: Indica cuál de los **6 pines** del **puerto B** **activan interrupciones**.
    - Existe ***una sola función de interrupción*** para los pines del **puerto B** que activen interrupciones.
- **PCIC**: Indica cuál de los **6 pines** del **puerto C** **activan interrupciones**.
    - Existe ***una sola función de interrupción*** para los pines del **puerto C** que activen interrupciones.
- **PCID**: Indica cuál de los **8 pines** del **puerto D** **activan interrupciones**.
    - Existe ***una sola función de interrupción*** para los pines del **puerto D** que activen interrupciones.

A continuación, se describen las posiciones del vector de interrupciones y qué es lo que hace cada una, este material fue sacado de la página 49 del datasheet del microcontrolador.

11. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATmega328P. For a general explanation of the AVR® interrupt handling, refer to Section 6.7 "Reset and Interrupt Handling" on page 15.

- Each interrupt vector occupies two instruction words in Atmel ATmega328P.
- In Atmel ATmega328P, the reset vector is affected by the BOOTRST fuse, and the interrupt vector start address is affected by the IVSEL bit in MCUCR.

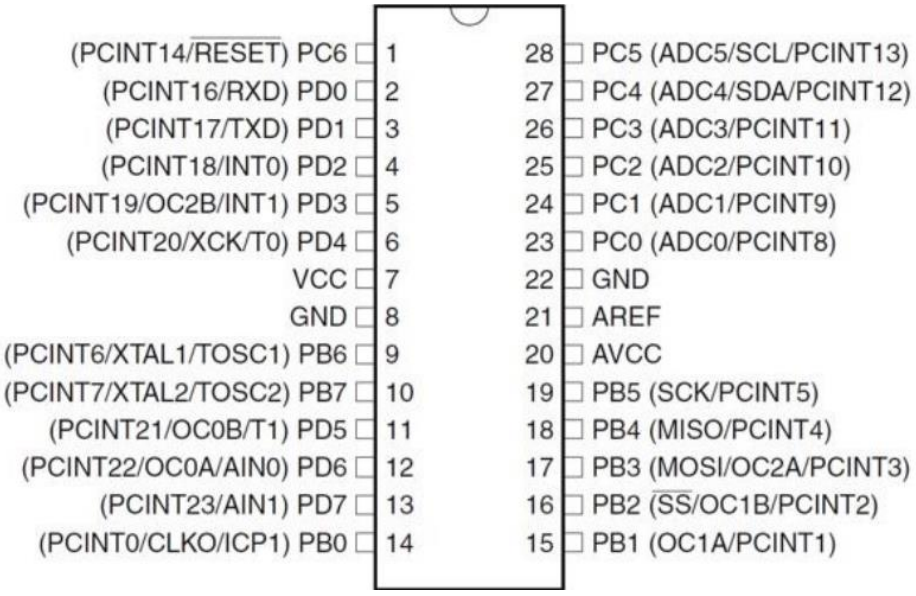
11.1 Interrupt Vectors in ATmega328P

Table 11-1. Reset and Interrupt Vectors in ATmega328P

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

- Reset: Este lo que hace es reiniciar todo el programa.
- INT0: Interrupción externa 0, se activa con los registros **PCI**.
- INT1: Interrupción externa 1, se activa con los registros **PCI**.
- PCINT0: Pin change interruption 0.
- PCINT1: Pin change interruption 1.
- PCINT2: Pin change interruption 2.
- WDT: Watchdog time out interrupt, es una interrupción que se activa cuando se ha acabado el conteo del watchdog.
    a. **Watchdog:** Es un temporizador que irá continuamente incrementando un contador para evitar que ocurran errores inesperados cuando el micro esté en marchas forzadas o entornos dañinos.
- Timers: Con contadores de comparación, de desbordeo, etc.
- SPI: Módulo de comunicación Serial Program Interface que comunica distintos dispositivos electrónicos.
- UART: Recepción y transmisión de datos entre dispositivos electrónicos.
- ACD: Ejecuta una interrupción cuando hay una conversión completa de datos analógicos a digitales.
- TWI: Se refiere a la comunicación I^2C.

Por lo tanto, se cuenta con 26 vectores de interrupción en el microcontrolador ATMEGA328P, con los cuales se pueden realizar un gran número de interrupciones por distintos factores.



Las interrupciones de los puertos están alojadas en los siguientes pines:

- **PORTB**: Indica cuál de los **6 pines** del **puerto B** **activan interrupciones**.
    - Si se quiere una interrupción externa en el Puerto B se puede usar el registro **PCIB o PCINT** (Pin Change), que se encuentra en todos los pines del puerto.
- **PORTC**: Indica cuál de los **6 pines** del **puerto C** **activan interrupciones**.
    - Si se quiere una interrupción externa en el Puerto C se puede usar el registro **PCIC o PCINT** (Pin Change), que se encuentra en todos los pines del puerto.

- **PORTD**: Indica cuál de los **8 pines** del **puerto D** **activan interrupciones**.
    - **PD2**: Contiene la interrupción externa **INT0**, para activar la interrupción 0 se le debe asignar un 1 lógico y para desactivarla un 0 lógico.
    - **PD3**: Contiene la interrupción externa **INT1**, para activar la interrupción 1 se le debe asignar un 1 lógico y para desactivarla un 0 lógico.
    - También i se quiere una interrupción externa en el Puerto C se puede usar el registro **PCID o PCINT** (Pin Change), que se encuentra en todos los pines del puerto.

Las interrupciones se pueden ejecutar por distintos factores externos que ocasionan cambios en la señal que entra a los pines de interrupciones externas, configurados con los 4 bits ISC11, ISC10, ISC01 e ISC00 pertenecientes al registro EICRA o registro de control de interrupciones descrito a continuación, aunque creo que la tabla 1.4 está al revés, es mejor checar la documentación del código:

BIT 7	6	5	4	3	2	1	0
-	-	-	-	ISC11	ISC10	ISC01	ISC00

Registro de control de la Interrupciones Externa A (EICRA)

Los bits 0, 1, 2 y 3 de este registro se usan para elegir alguno de los cuatro formas de disparo de las interrupciones INT0 e INT1, estos se describen en la tabla 4.1, los bits para INT0 son ISC01 (bit 1) e ISC00 (bit 0) y para INT1 son ISC11 (bit 3) e ISC10 (bit 2).

Tabla 4.1 formas de disparo de las interrupciones INT0 e INT1.

ISC11 / ISC01	ISC10 / ISC00	DESCRIPCION
0	0	Se genera una llama a interrupción (INT0 / INT1) cuando hay un nivel bajo.
0	1	Se genera una llamada a interrupción (INT0 / INT1) con cualquier cambio de estado lógico.
1	0	Se genera una llamada a interrupción (INT0 / INT1) con un flanco de bajada.
1	1	Se genera una llamada a interrupción (INT0 / INT1) con un flanco de subida.

En conjunto con esto se usa la EIMSK o máscara de interrupciones, que sirve para indicar qué interrupción es la que quiero emplear.

BIT 7	6	5	4	3	2	1	0
-	-	-	-	-	-	INT1	INT0

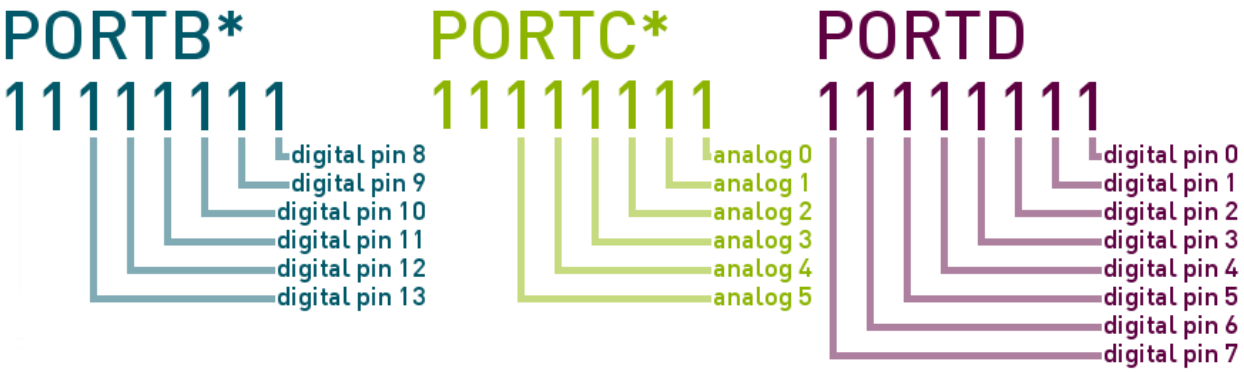
Registro de Mascara de Interrupción Externa (EIMSK)

Por último, el registro EIFR es un registro de banderas que responde al **ISR o Interruption Service Routine** que cuenta con un bit de status llamado GI (General Interruption) o switch general de interrupciones, que indica si la interrupción va a estar activada o no para que se ejecute cuando ocurra el estímulo que la detona.

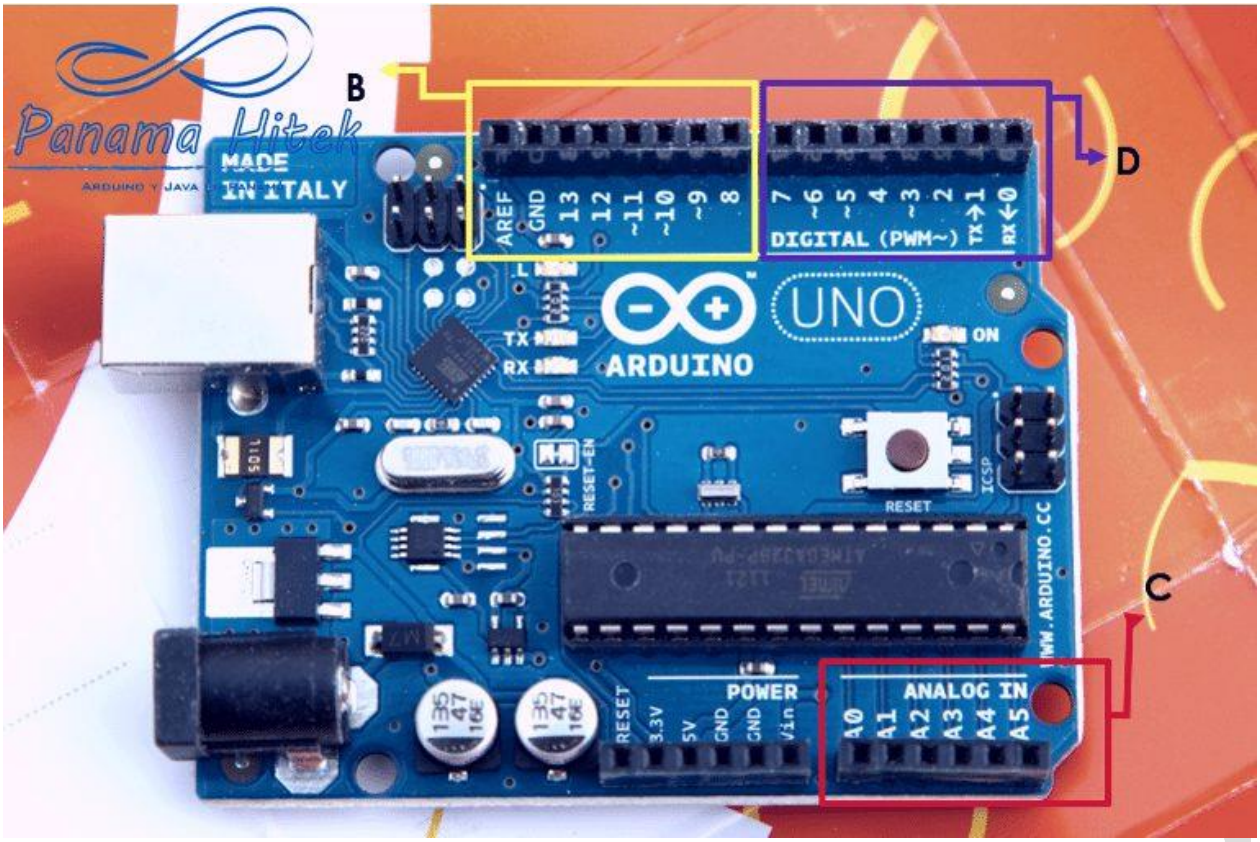
BIT 7	6	5	4	3	2	1	0
-	-	-	-	-	-	INTF1	INTF0

Registro de Banderas de Interrupción Externa (EIFR)

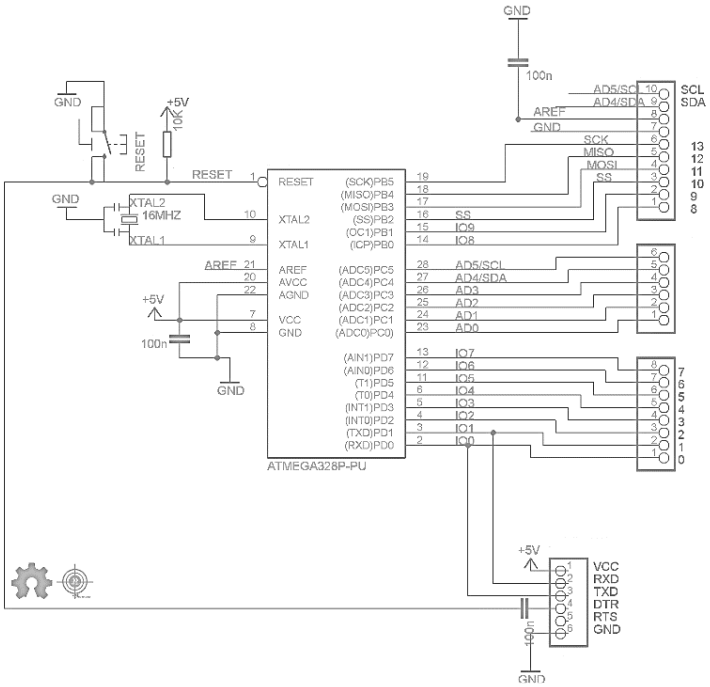
Puertos B, C y D Físicos en el Arduino



Los pines de mayor peso de los puertos **B** y **C** no se usan, por lo cual en los puertos **B** y **C** existen **6 pines** y en el puerto **D** existen **8 pines**, los pines de cada una de las 3 familias de puertos de la placa Arduino UNO se muestran en la siguiente imagen.



Conexión Física de las Interrupciones



Para activar las interrupciones se puede usar botones, cuando esto se realice así, se debe usar resistencias pull up o down para elegir la forma en la que las interrupciones funcionarán:

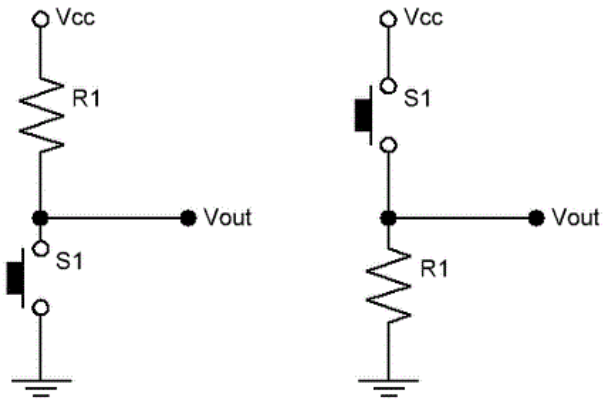
La configuración de resistencias Pull-up o Pull down son utilizadas para fijar un 0 o 1 lógico en el nodo de algún circuito, es muy utilizado en puentes H por ejemplo y el valor de la resistencia depende de la corriente que se busque para que circule a través del circuito final.

Pull-up:

Es una configuración de resistencias que cuando el switch está abierto, entrega un 1 lógico y cuando está cerrado entrega un 0 lógico en el nodo Vout.

Pull-down:

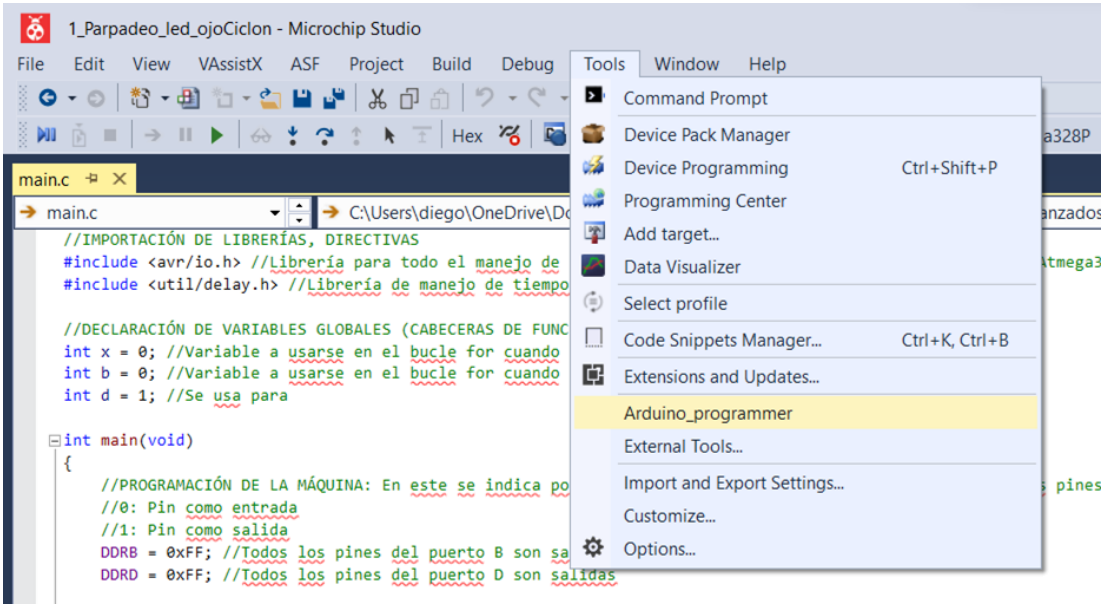
Es una configuración de resistencias que cuando el switch está abierto, entrega un 0 lógico y cuando está cerrado entrega un 1 lógico en el nodo Vout.



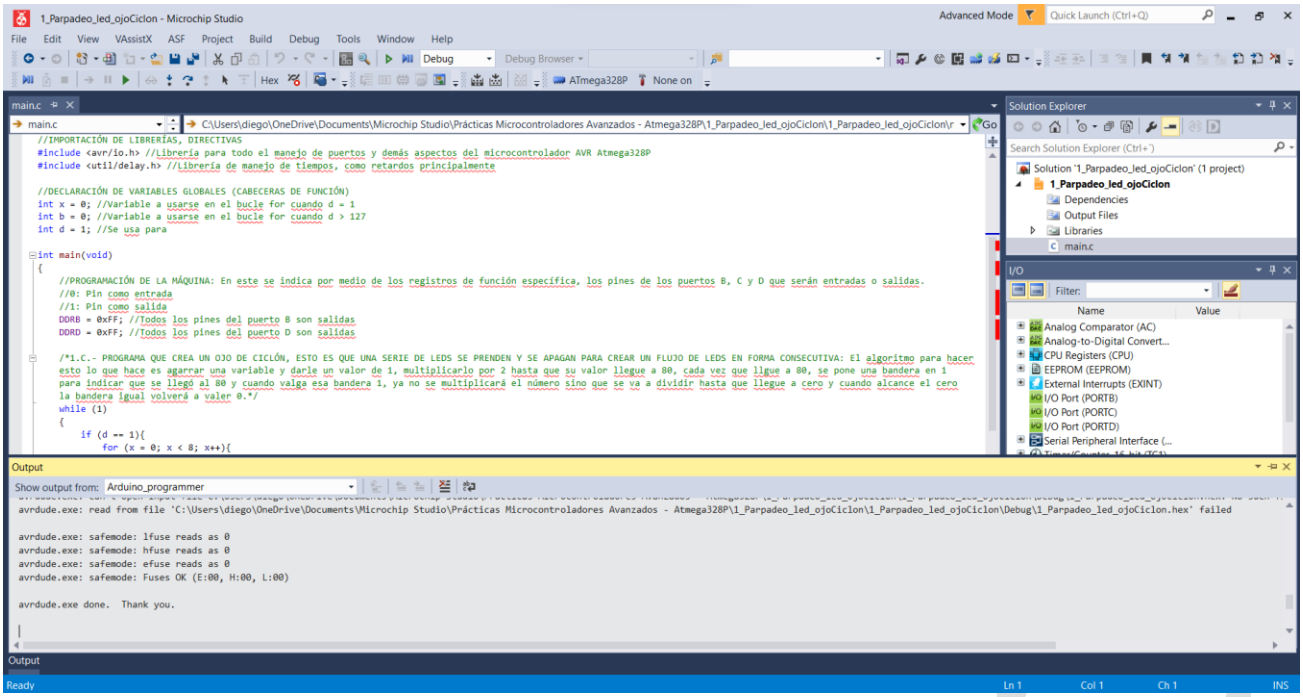
Programar Físicamente la Placa Arduino UNO con el Programa de Microchip Studio

Recordemos que la forma de programar la placa del Arduino UNO, ya habiendo realizado la configuración de su programador en el programa de Microchip Studio es seleccionando la opción de:

Tools → Arduino programmer (opción creada y nombrada cuando se configuró el Arduino).

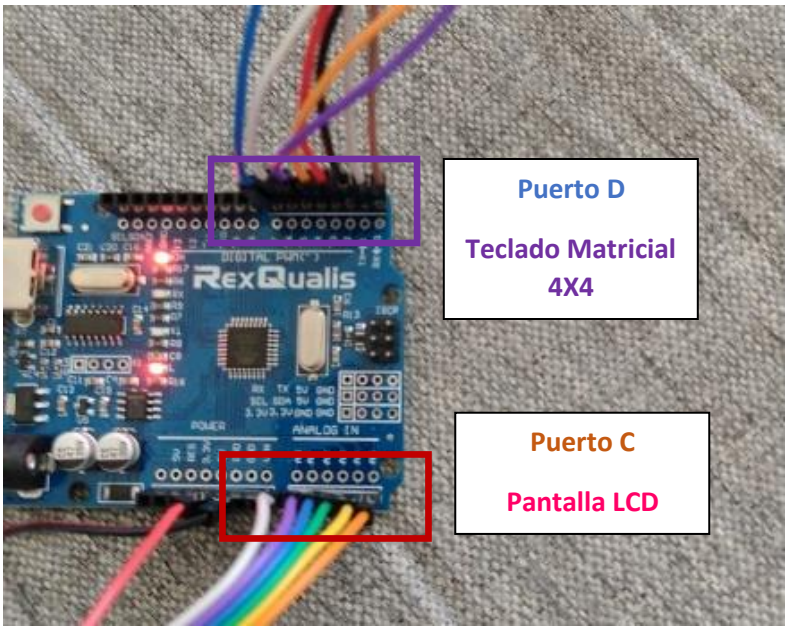


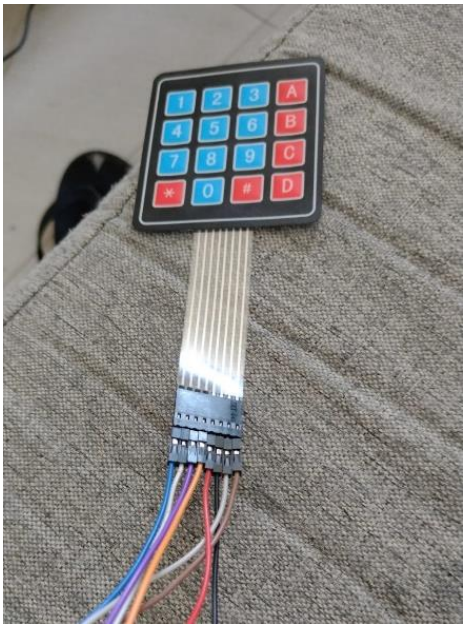
Es muy importante mencionar que cuando se carga un código a una placa Arduino, debemos tener cuidado que no haya nada conectado a los pines **TX** y **RX** del **Puerto D** porque si no Microchip Studio indicará un error cuando tratemos de subir el programa, ya habiéndolo subido podemos utilizar esos pines de nuevo y no habrá problema.

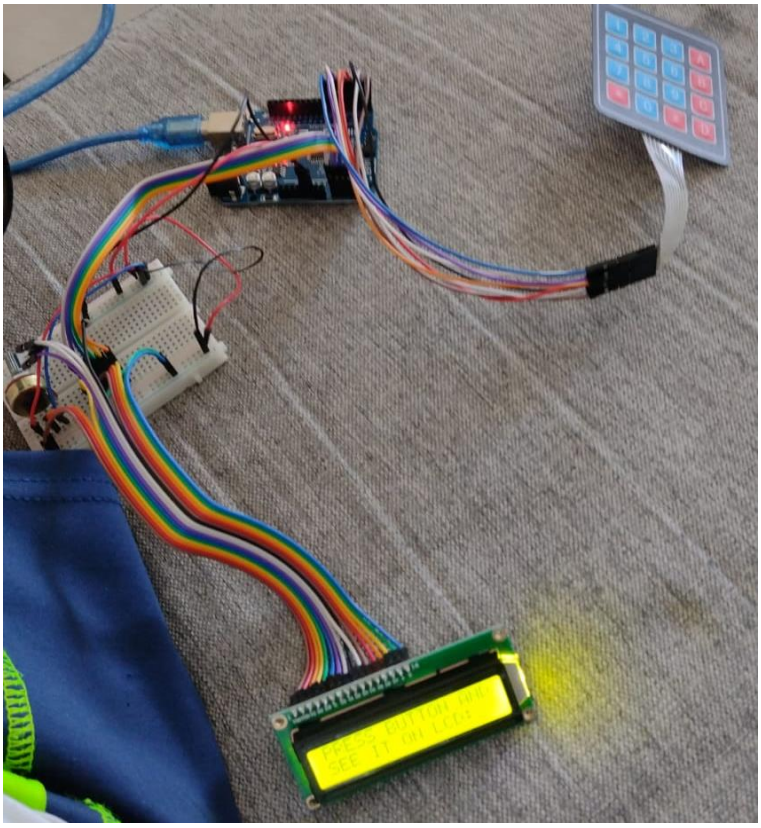


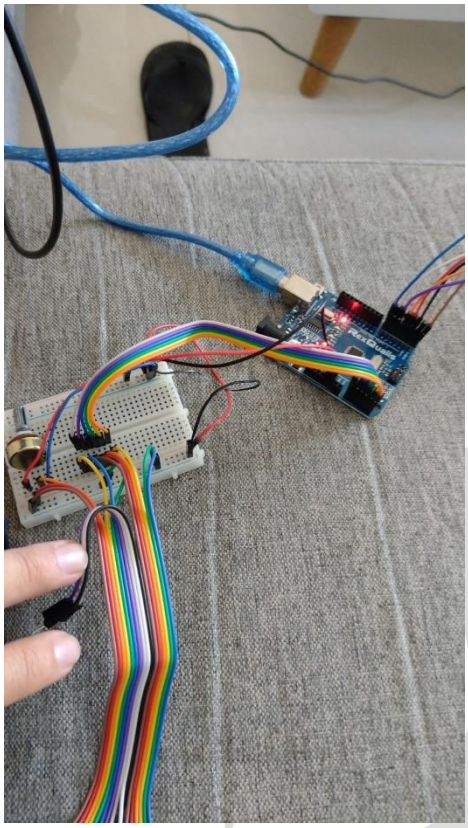
Para la creación de proyectos controlados con la placa Arduino UNO que además controlan más de un dispositivo periférico como lo es el teclado matricial 4X4, una pantalla LCD, etc. Normalmente se utilizarán los siguientes puertos para programar los dispositivos enlistados a continuación:

- **Puerto B**: Periféricos adicionales, como motores, sensores, etc.
- **Puerto C:** Pantalla LCD.
- **Puerto D**: Teclado matricial 4X4.









Utilización de Librerías en Microchip Studio: Control de Pantalla LCD

Cuando se quiera utilizar una librería propia para usarse al ejecutar el código de control de un LCD se deben seguir los siguientes pasos para crear un archivo `#include` con extensión .h y que pueda ser importado: `#include "..forma/de/llegar/al/archivo/nombre librería personalizada.h"`