Informe Entrega Individual N°3

Utrera, Francisco

En el presente informe se detalla el proceso de implementación de un delay utilizando un bucle for y la utilización de este para un programa que, usando una barrera óptica, sensa si hay un obstáculo y en caso de haber uno se deja prendido un led pero si no detecta nada el led comienza a titilar con una frecuencia de 1 ms (aquí es donde se usa el bucle for implementado anteriormente).
En primer lugar se realizó una primera versión de la placa en SimulIDE donde para emular la barrera óptica se usa un pulsador (si está pulsado es que la barrera sensa "algo", en caso contrario no) conectado al pin 1. En el Arduino IDE se setearon en el void setup al pin del led (pin 13) y al pin 1 como entrada. Además se dejó encendido el led. En el loop se implementó un sencillo programa en el que si el pulsador está pulsado se apaga el led y si no lo está se vuelve a prender el led.
Al comprobar que esto funcionaba se pasó a la etapa de implementación del delay usando un bucle for.
En primer lugar se buscó la frecuencia de operación del Arduino Uno. Se encontró que este funciona a 16 MHz, por lo tanto cada iteración tarda 0,1875 µs (ya que en cada iteración se realizan 3 operaciones: primero se chequea si se sigue cumpliendo la condición, luego se incrementa el iterador y finalmente se hace un goto al principio, es decir, se reinicia el proceso). Sabiendo esto se trató de hacer un delay de 1 s (ya que es fácil de ver si efectivamente este anda), sabiendo que como cada iteración tarda 0,1875 µs había que hacer 5.333.333 iteraciones. Cuando se realizó este delay se vió que no funcionaba y para esto había consideré dos opciones: El compilador al ver que el bucle for no hace nada dentro lo optimiza o el SimulIDE no puede realizar estas operaciones ya que es solo un simulador y no está diseñado para manejar estos números tan grandes.
Contemplando esto último se cambió el bucle para que realice sólo 5300 operaciones (es decir que el delay sea de 1 ms). Al efectuar este cambio no se pudo apreciar que titile el led, pero esto es lógico ya que 1 ms es muy poco tiempo y no es visible para el ojo humano.
Para poder ver si efectivamente hay un delay debido al for se agregó a la placa una resistencia puesta a tierra que se conectó al pin 2. Lo que se hizo fue que luego de cada delay, se mande una señal que pase por la resistencia y que luego se vuelva a apagar esta señal. Con un osciloscopio se mide la frecuencia de la señal que pasa por la resistencia. En la figura 1 se puede apreciar la medida de frecuencia indicada por el osciloscopio:

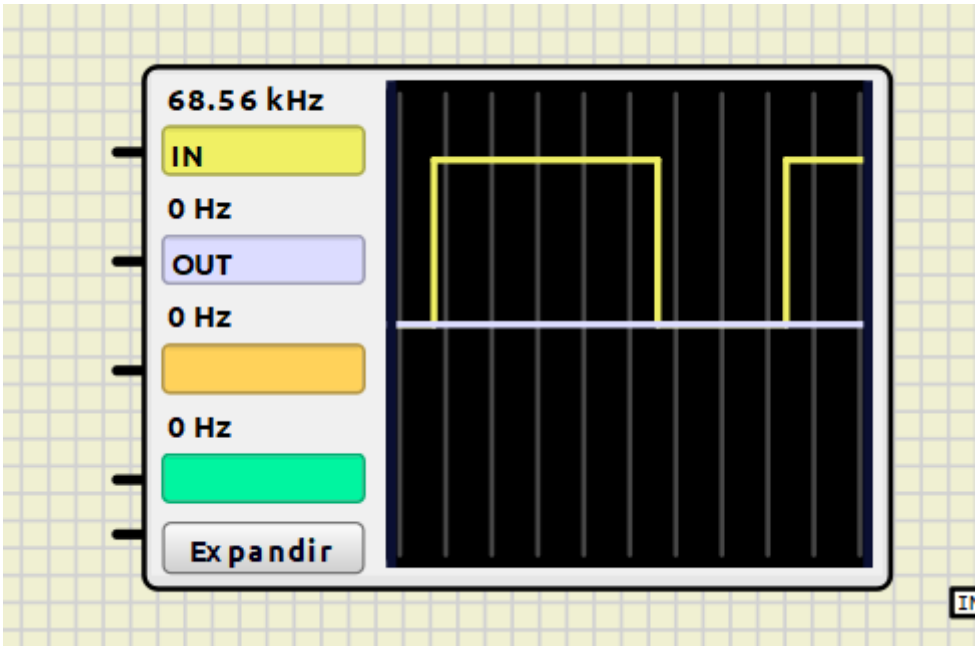

Figura 1: Señal de salida medida por el osciloscopio.

Como se ve en la figura, efectivamente hay un delay debido al bucle for. Pero hay un pequeño problema. Si la frecuencia es de 68,56 kHz, significa que el delay es de:

$$delay = (68,56\ kHz)^{-1}/2$$
$$delay = 7\mu s$$

Aclaración: divido por 2 ya que el delay afecta a la mitad de la onda, no el período entero

Entonces efectivamente se ve que hay una optimización realizada por el compilador, ya que de un delay de 1 ms originalmente sólo se tiene 7 µs de delay (casi 143 veces menos).
De todas formas se procede a usar delay para la versión final de la simulación. Para esta se cambió sólo el código. Si el sensor óptico sensa "algo" se deja prendido el led y en caso contrario se apaga el led, se aplica el delay, se prende el led y se vuelve a aplicar el delay. Lo mismo que se hace con el led se hace con el pin 2 que tiene la resistencia para poder medir con el osciloscopio las ondas de salida.
Ahora se observa que la frecuencia medida baja un poco (a 48,50 kHz), es decir que el delay aumenta a 10 µs, siendo ahora el delay 100 veces menor que lo teórico.
Finalmente como curiosidad se fue al github de Arduino[1] para ver cómo se implementó la función delay en Arduino.

Referencias:

[1] «Arduino AVR Boards». Arduino, 26 de septiembre de 2023. Accedido: 26 de septiembre de 2023. [En línea]. Disponible en: https://github.com/arduino/ArduinoCore-avr