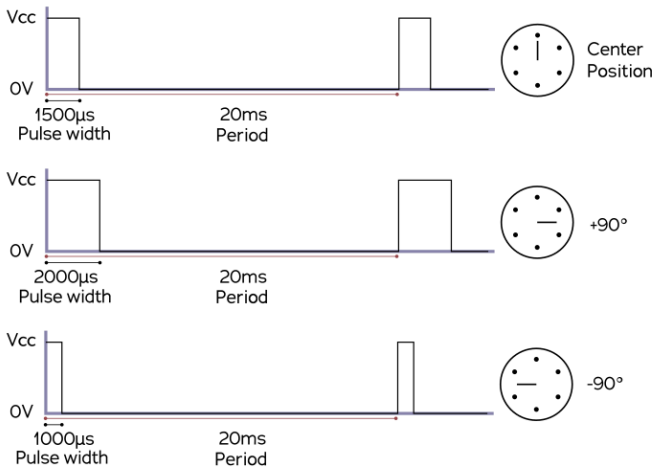
PSE. Trabajo práctico contadores/timers

1-Desarrollar un driver para el timer1 del atmega328p y un cronómetro con minutos:segundos:centésimas.

a. Definir la estructura de datos overlay, definir funciones init con nombres apropiados, etc.
b. Configurar el timer para que genere una interrupción cada una centésima de segundo (probablemente el modo CTC es indicado). Recuerde los 3 pasos para que sucedan y se procesen interrupciones: definir la rutina de atención de interrupciones, activar las interrupciones para el timer, y activar las interrupciones de manera global.
c. En la rutina de atención de interrupciones, actualizar una variable global que mantenga los "ticks" transcurridos (las centésimas de segundo transcurridas).
d. En main, mostrar via serial el cronómetro con una salida similar a
   01:23:78
   (debe verse siempre en la misma línea de texto cuando se utilice un programa de comunicación como minicom).

2- Modificar el driver del timer1 para controlar un servo.

a. Para controlar el angulo de rotación mecánica de un servomotor se debe generar una señal pwm, de 50hz. El ciclo de trabajo debe variar entre [1ms, 2ms]. Tipicamente:



(Probablemente un modo útil del timer1 es el modo 14, fast pwm con el registro icr1 como tope, para generar la frecuencia requerida. Y el timer configurado para que realice un "clear on match". Luego, será posible utilizar uno de los dos registros comparadores (OCR1A u OCR1B) para establecer el ciclo de trabajo de la señal pwm. Aquí no necesitamos interrupciones.)

b. Agregue un potenciómetro leído por ADC para que sea la interfaz con el usuario, y pueda utilizar el control del servo via el knob.

3. Crear un sistema embebido que se comporte como una placa de sonido rudimentaria para la PC. La idea es poder reproducir archivos .wav que estén en la PC. Los archivos serán .wav de 8bit sin signo por muestra, y de 11025hz. El archivo será enviado desde la PC via serial, a la placa rudimentaria de sonido, mientras se reproduce el archivo.

PSE. Trabajo práctico contadores/timers

Este ejercicio es en grupo. Conformaremos los mismos grupos que tuvimos para mayday. El unico grupo que se modifica es que Simon, Alejo y Diego serán “un grupo”.

Una posible implementación es que para cada muestra del archivo de audio original (para cada byte), el timer1 debería poder generar un ciclo de trabajo acorde a la muestra para ese período de la señal pwm (ver application note de st en pdf aparte). Para lograrlo, aquí va una idea :

a. Configurar el timer1 para generar interrupciones con una frecuencia los mas aproximadamente posible a 11025hz.
b. Configurar el driver serial para trabajar a 115200bps. (esto permitirá recibir via serial aproximadamente 11000 bytes por segundo del archivo wav).
c. Utilizar algún modo del timer1 para que la señal de salida del generador de señal haga un “clear on match” cuando coincida el contador con alguno de los registros comparador. El registro comparador es el que se utilizará para variar “la tensión media” que corresponda a la muestra procesada actual (probablemente el mismo modo utilizado con el servomotor).
d. Luego, la aplicación main puede ser implementada como sigue:

```
unsigned char dato;
while (1) {
  dato = recibir_dato_del_serial()
  Escalar dato para que “la tensión media a generar en la señal de salida” se corresponda con el valor de dato. Como la señal de audio utiliza menos de 5v (aproximadamente 1v), el tope de la “tensión media” debería ser 1v. (ejemplo: si el dato recibido por serial del archivo de audio original es 255, deberíamos generar una tensión media de 1v).
}
```

En la ISR del timer, actualizar el registro comparador con el valor escalado de la muestra actual (el dato recibido por serial ya escalado por main).

La señal de salida debe ir a un parlantito o equipo de audio (que amplifique el sonido). Si se escucha muy mal Dario, Emiliano, Simon y Alejo nos pueden ayudar a confeccionar en la protoboard el low-pass filter que aparece en el application note.

En Linux, se puede utilizar el programa ffmpeg para convertir cualquier archivo de sonido (ejemplo, un tema en mp3) a .wav con el formato requerido.