Práctica 12

Uso del Convertidor Analógico Digital del ATmega1280

Objetivo: Mediante esta práctica el alumno aprenderá la programación y uso básico del convertidor analógico digital del microcontrolador ATmega1280.

Material: 1 – Tarjeta T-Juino
1 – Cable USB
1 – LED

Equipo: Computadora Personal con USB, AVRStudio y WinAVR

Teoría: – Programación y uso del ADC (Diagrama, Funcionamiento, regs. de conf. y operación).
– Código Morse.

Desarrollo:
Implementar un decodificador de código Morse que se transmitirá de forma inalámbrica a través de cambios de intensidad en el "flash" de un teléfono celular (app de ejemplo).

Para poder percibir estos cambios de intensidad se podrían utilizar dos componentes comunes como se muestra en la siguiente figura:

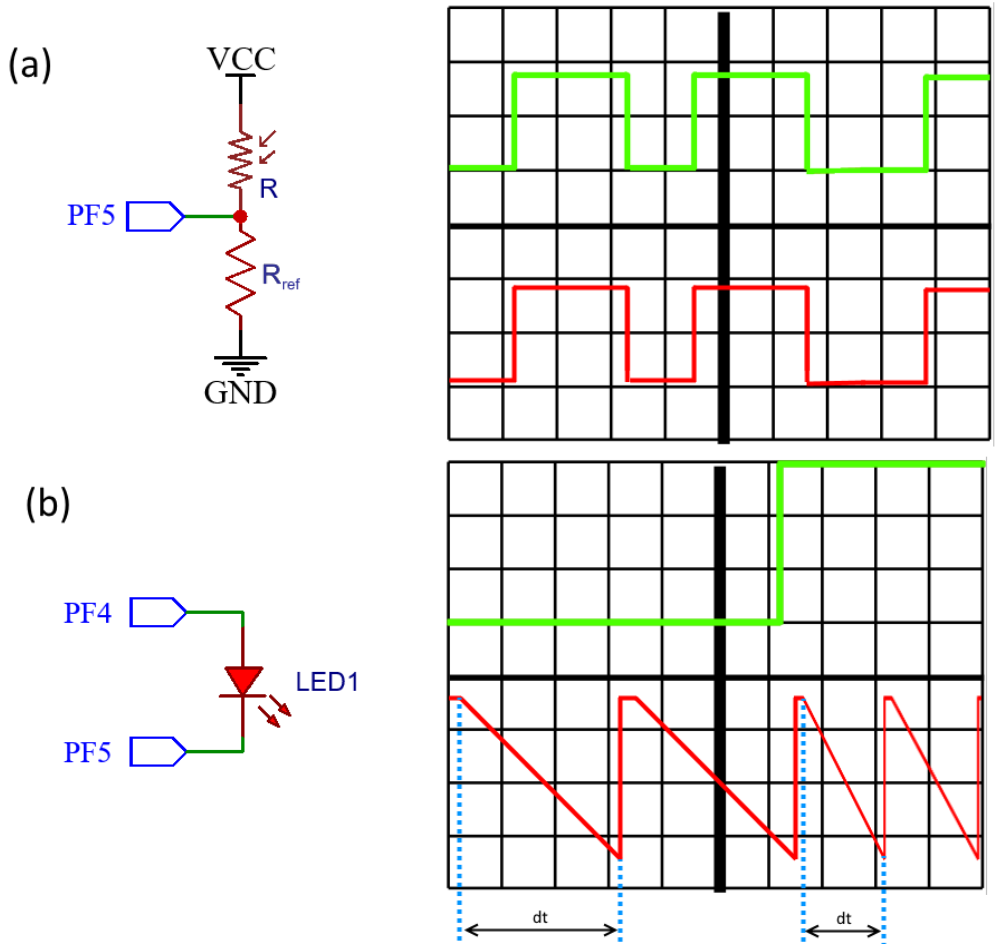

Figura 1. Circuito sensor de cambios de luminosidad mediante componentes comunes.

De la Fig. 1 se muestra ejemplifica el comportamiento teórico de ambos circuitos. En las graficas se aprecia en línea verde el estado lógico del "Flash", y la línea roja es el voltaje sobre PF5. Vemos que para el caso (a) al usar una foto-resistencia observamos una relación directa con respecto al estado del "flash". En cambio, si utilizamos un LED para percibir estos cambios de intensidad luminosa, debemos hacerlo de forma indirecta midiendo el tiempo de descarga *dt*.

Es importante remarcar que la curva de descarga del LED es casi lineal, así que el voltaje que se decida usar como el límite inferior podría ser casi cualquiera, ya que la relación entre los tiempos de descarga se sigue manteniendo. La limitante es el tiempo de conversión del ADC, así que se debe cuidar que el periodo de descarga sea mayor al periodo de conversión del ADC al momento de elegir el voltaje del límite inferior.

Cualquiera de las dos formas mostradas en Fig. 1 es válido para esta práctica.

Para esto se deberá diseñar e implementar las siguientes funciones de configuración y operación:

1) `void` **`ADC_Ini`** `(void)`
   Esta función inicializa para 16 bits de resolución y habilita el ADC del microcontrolador de forma genérica. Encontrar el desplazamiento (offset) de la medición y almacenarla.

2) `uint16_t` **`ADC_Read`**`( uint8_t channel )`
   Está función lo que realiza es una lectura del ADC usando el canal correcto y retornando el valor de 16 bits acorde a la aplicación, compensando el desplazamiento de la medición.

3) `uint16_t` **`measureDischargeTime`**`(void)` *`[opcional]`*
   En el caso de usar la opción (b) de la Fig. 1, es recomendable implementar la función que retorna el tiempo de descarga del LED. Esta función deberá polarizar de forma inversa el LED y medir el tiempo de descarga hasta que llegue al voltaje del límite inferior que elijan, retornando el numero de milisegundos que transcurrieron. El propósito de la función es para discernir si el "Flash" este encendido o no.

Una vez teniendo estas funciones solo es cuestión de medir el tiempo que se mantiene encendido y apagado el "Flash" para distinguir los siguientes eventos:

● dot	es un periodo pequeño (~300ms) con el flash encendido.
▬ dash	es un periodo largo (~1s) con el flash encendido.
O rest	es un descanso de duración pequeña (~300ms) entre cada símbolo (*dot* y *dash*).
▭ gap	es un descanso de duración larga (~1s) entre cada carácter.

Los periodos son una aproximación ya que no está estandarizado y podría ser más rápido o más lento, pero se mantiene la relación en base a la duración de *dot*.

Ejemplo de una captura utilizando la configuración de tiempo por defecto de la aplicación de la lampara Morse del celular:

Estado percibido del Flash	Duración (ms)	Evento	Significado
1	318	**dot**	
0	290	rest	
1	362	**dot**	S
0	270	rest	
1	351	**dot**	
0	909	gap	
1	988	**dash**	
0	266	rest	
1	998	**dash**	O
0	263	rest	
1	989	**dash**	
0	883	gap	
1	368	**dot**	
0	265	rest	
1	319	**dot**	S
0	314	rest	
1	334	**dot**	

Como evidencia, enviar cualquier mensaje a través del “Flash” en codificación Morse, capturarlo con el Microcontrolador y desplegarlo en la terminal.

Comentarios y Conclusiones.

Bibliografía.