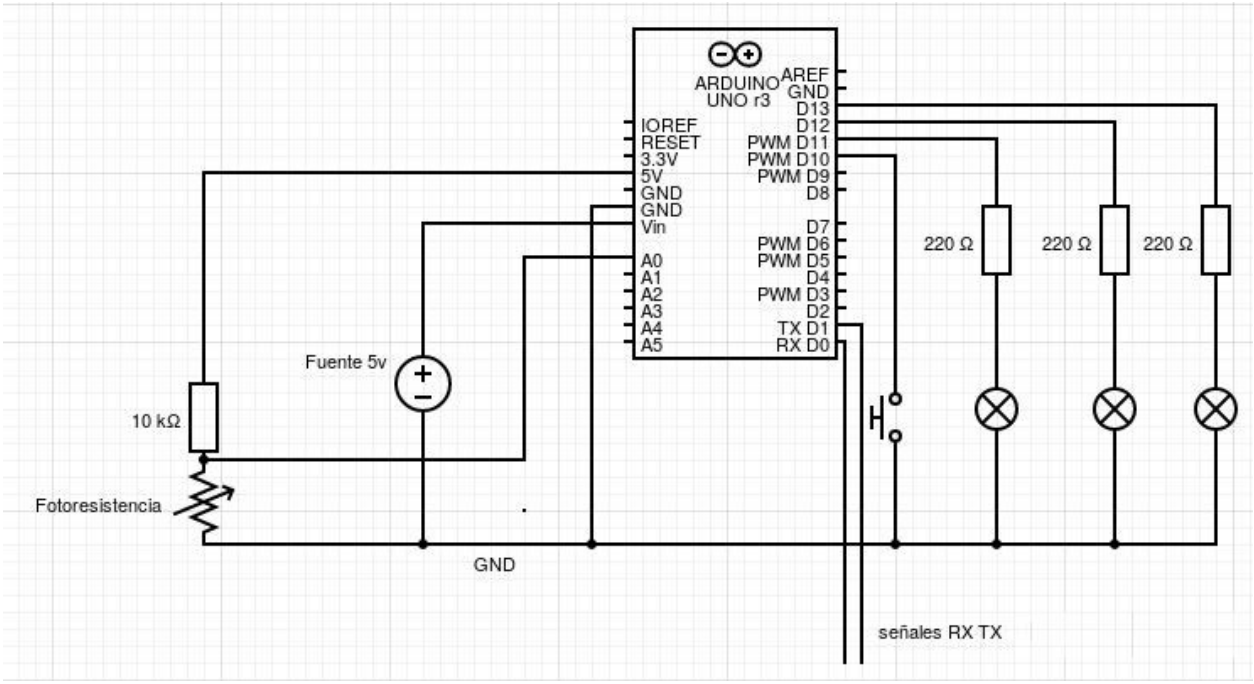
Programación de Sistemas Embebidos 2020
Facultad de Informática
Universidad Nacional del Comahue
Trabajo Práctico 4

ADC (Analog-to-Digital Converter) - UART asincrónico – GPIO paralelo: aplicación de adquisición de datos

El objetivo es lograr el control de algun software en la PC mediante la señal de luz.
Para eso, se debe desarrollar una aplicación que comunique el microcontrolador con la PC. El sistema embebido debe adquirir la señal de luz y enviarla de manera digital a la PC. Se debe utilizar interfaz serial UART para la comunicación, y el sensor de luz (fotoresistencia) a través del periférico ADC.

Hardware: configure el hardware en la protoboard como se indica en la figura:

- Se debe conectar los pines del serial UART (rx/tx) a los pines del conversor UST-TTL.
- La configuracióin de la fotoresistencia y el divisor de tensión debe estar como en el esquemático, conectando la salida del divisor de tensión a un pin de entrada analógico.



Esquemático creado con : https://www.circuit-diagram.org/

Software: Desarrollar una aplicación que contenga un driver serial, un driver ADC, y un control GPIO para los LEDs y pulsador. El objetivo es visualizar el nivel de luz presente en el ambiente, a través de una escala de 0 a 7. El "display" de esta escala serán los LEDs. También, se deberá controlar una aplicación en la PC utilizando el sistema de adquisición de la señal de luz. La aplicación debe funcionar de la siguiente manera:

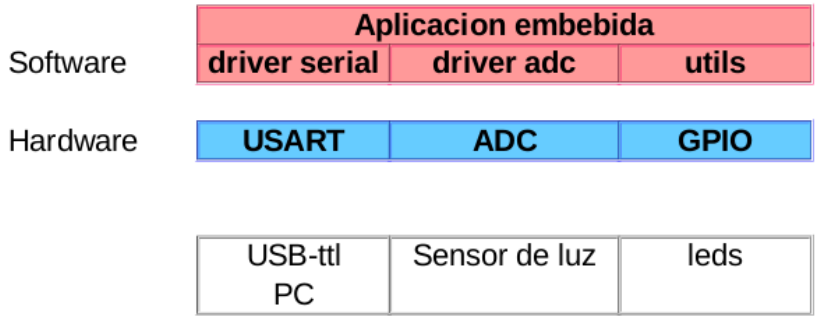
1- Desarrollar el driver ADC. Utilice la metodología de desarrollo de drivers visto en teoría, en el TP anterior y en la clase del video. Complete las funciones del driver ADC. Agregue la compilación al Makefile.

2- Leer la señal proveniente del sensor de luz y convertir el valor de 10bits devuelto por el driver ADC (0-1023) a una escala de 0 a 7. Enviar ese valor a los leds, para obtener una señal visual de la cantidad de luz capturada por el sensor y el sistema.

Programación de Sistemas Embebidos 2020
Facultad de Informática
Universidad Nacional del Comahue
Trabajo Práctico 4

3- Además, se debe utilizar la señal de luz percibida para controlar algun software en la PC. Para esto, se debe enviar el valor de 10bits obtenido desde el ADC a la PC, utilizando el driver serial ya desarrollado. Como el valor de 10bits será un entero (int de 16bits en AVR) se debe convertir el número entero a una cadena ASCII. Es decir, lo que se envía a la PC es la representación ASCII del número. Ejemplo, si el número es 251 el sistema enviará a la PC los valores 0x32, 0x35, y 0x31 y un caracter de fin de línea.

4- El software a desarrollar tendrá los siguientes archivos fuentes: main.c serial.c serial.h adc.c adc.h utils.c utils.h. Un diagrama de bloques de la arquitectura del sistema:



5- De esta manera el main con un while infinito podría parecerse simplemente al siguiente pseudo-código:

```
int v;
while (1) {
                    v = get_adc(3);
                    leds(convertir_a3bits(v));
                    serial_put_int(v);  /* serial_put_int() probablemente debe utilizar serial_put_str() */
}
```

6- Un script para Linux PC que captura los datos del serial y los transforma en pulsaciones del teclado es provisto de ejemplo.

```
int v;
while (1) {
            v = get_adc(3);
            leds(convertir_a3bits(v));
            serial_put_int(v);  /* serial_put_int() probablemente debe utilizar serial_put_str() */
}
```

(Puede agregar a la propuesta su propia “pincelada”. Ejemplo: el botón podria ser utilizado para enviar un código a la PC diferente al ASCII numérico, y agregar un control más para un videojuego. Prosiguiendo el ejemplo, podría controlar los movimientos izquierda y derecha con el sensor de luz en un juego como el Galaxy, y el boton para disparar.

Verificación: compilar, vincular y enviar el firmware AVR. Verificar el funcionamiento de la aplicación de manera progresiva.

Entrega: subir (push) el trabajo práctico resuelto (o sus versiones intermedias) al repositorio git compartido (http://github.com/zrafa/pse2020.git)