Sistemas Embebidos
Trabajo práctico Nº1 - Año 2024
Conversores AD y DA. Protocolos de comunicación serial cableados

Objetivos

- Comprender el uso y la utilidad de conversores A/D y D/A.
- Implementar una comunicación serial simple entre un sistema embebido simple y una computadora.
- Integrar los componentes de un sistema de IoT completo.

Metodología

Trabajo individual o grupal. 2 estudiantes por grupo máximo.
Tiempo de realización estimado: 2 a 4 clases.

Aprobación

- Mostrar en clases la aplicación funcionando correctamente.
- Enviar los programas de computación implementados a través de la plataforma Moodle.
- Escribir un breve informe y enviarlo a través de la plataforma Moodle.
- Responder el cuestionario Trabajo Práctico Nº1 - Cuestionario.

Materiales necesarios

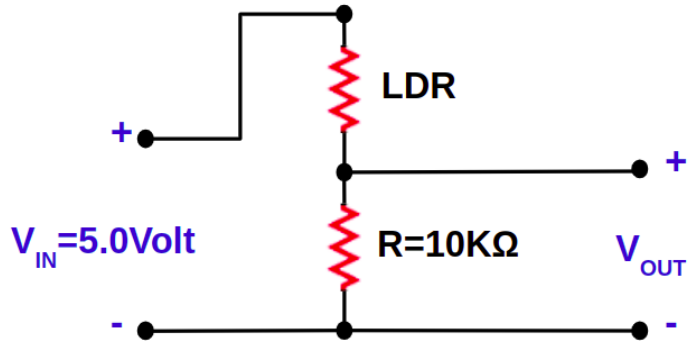
- Placas Arduino UNO y circuito con leds sensores (provistas por la cátedra).
- Entorno de desarrollo de Arduino UNO (puede descargarse de https://www.arduino.cc/en/software).
- Intérprete de lenguaje de programación Python 3.
- Módulo pySerial para la comunicación serial.
- Servidor web. Puede utilizar:
    - Frameworks de desarrollo web. Se sugiere Flask, FastAPI o Django.
    - Apache Web Server (En el trabajo práctico N°5 de la asignatura Redes de Computadoras se dan instrucciones de instalación y uso).
- (Opcional). Puede escribir el código a ejecutar en el Arduino UNO empleando un simulador adecuado. La cátedra provee una simulación con el circuito listo para ser utilizado en el enlace: https://wokwi.com/projects/391281841803809793
  (en el Anexo 5 se dan instrucciones de uso de este simulador). Luego podrá copiar el código a la memoria flash del Arduino UNO.

Actividad 1:

Realizar una aplicación que realice las siguientes tareas a través de una página web sencilla:

1. Controle el brillo de los led 9, 10 y 11 del Arduino UNO utilizando el conversor Digital a Analógico del mismo. El brillo de estos leds debe controlarse de forma independiente para cada led.
2. Encienda y apague el led 13 del Arduino UNO. Deberá utilizar dicho pin como salida digital.
3. Muestre la intensidad luminosa captada por el LDR (conectado al pin A3 del Arduino UNO). Deberá utilizar el conversor Analógico a Digital.
4. El control del brillo de los led 9, 10 y 11, el pin 13 y el valor de la intensidad luminosa sensado por el LDR deberá realizarse y mostrarse a través de una página web que deberá poder accederse desde otras computadoras conectadas en la misma red LAN o Internet.

El circuito del LDR es el siguiente:



Siendo:

$$V_{OUT} = \frac{10*10^{3}\Omega * V_{IN}}{R_{LDR} + 10*10^{3}\Omega}$$

La resistencia del LDR es función de la intensidad de la luz según la siguiente expresión aproximada:

$$R_{LDR} = \frac{1*10^{6}(\Omega * Lux)}{Intensidad}$$

Nota: Una expresión más exacta es la siguiente $R_{LDR} = \frac{1*10^{6}(\Omega * Lux)}{Intensidad^{2/3}}$ pero se sugiere utilizar la aproximación indicada arriba.

Sugerencias:

- Para la comunicación entre el Arduino y la computadora en la cual correrá el servidor se sugiere una aplicación creada con Python. Investigue el uso del módulo pySerial.
- Para la comunicación entre el Arduino y la computadora en la cual correrá el servidor se sugiere crear una trama de datos con campos de longitud fija de tipo String. Implemente las aplicaciones en el Arduino y en la computadora en base a dicha trama. La definición adecuada de esta trama le permitirá escribir todos los módulos de software a ejecutar en el sistema embebido y servidor de forma independiente y luego integrarlos.

Anexos

Anexo 1: Instrucciones útiles de Arduino
Anexo 2: Comunicación por el puerto serial
Anexo 3: Diseño de aplicaciones web con Flask
Anexo 4: Introducción a HTML y formularios
Anexo 5: Simulador de Arduino UNO sugerido por la cátedra.

Anexo 1: Instrucciones útiles de Arduino

En este anexo se presentan instrucciones útiles de Arduino para este trabajo práctico. Para un listado completo de instrucciones, consulte: https://www.arduino.cc/reference/en/#functions
Nota: Podrá escribir el programa a ejecutar sobre el Arduino UNO en un simulador y luego pasarlo al Arduino real. En el Anexo 5 se sugiere un simulador.

1.1 Estructura básica de un programa en el IDE de Arduino

Un programa en Arduino posee dos funciones: setup() y loop()

```
void setup() {
/*Dentro de la función setup se implementa el código que configura la plataforma. Acá deben configurarse pines como entrada/salida, periféricos a utilizar, habilitar interrupciones y declarar el nombre de las respectivas rutinas de servicio.
La función setup solo se ejecuta una vez al iniciar la plataforma.*/
}
void loop() {
/*La función loop se ejecuta repetidamente sin fin. Acá debe colocar el código que se ejecutará repetidamente mientras la plataforma esté encendida*/
}
```

1.2 Funciones útiles en C para Arduino

```
pinMode(6, OUTPUT);
```

Declara el pin 6 como salida. No retorna nada.

```
pinMode(3, INPUT);
```

Declara en pin 3 como entrada. No retorna nada.

```
digitalRead(pin);
```

Lee el valor digital del pin. Retorna un entero que puede ser 1 o 0.

```
digitalWrite(pin, value);
```

Escribe un valor digital en el pin indicado, value puede valer HIGH o LOW.

```
analogRead(A3);
```

Lee el valor analógico del pin A3. Retorna un valor entre 0 y 1024.

```
subcadena=cadena.substring(0, 2);
```

Obtiene una subcadena a partir de una cadena de caracteres. En el ejemplo, la subcadena toma desde el caracter 0 en cadena (incluyendo el caracter 0) hasta el caracter 2 (sin incluir el caracter 2).

```
variable=variable1.toInt();
```

Transforma variable1 en un entero que se almacena en variable.

Anexo 2: Comunicación por el puerto serial

El puerto serie y el programa en Python deben configurarse con la misma velocidad, misma cantidad de bits de datos y misma cantidad de bits de paridad.

2.1 Lado del Arduino

2.1.1 Configuración del lado del Arduino

```
Serial.begin(9600,SERIAL_8N1);
```

Configura el puerto con velocidad 9600, 8 bits de datos, sin bit de paridad y un bit de parada. Para más configuraciones consultar:
https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/

2.1.2 Envío de datos

```
Serial.println(datos);
```

Los datos deben ser del tipo string.

2.1.3 Recepción de datos

```
if(Serial.available()>0){
    string_recibido=Serial.readStringUntil('\n');
    ---otro código---
    }
```

La función `Serial.available()` devuelve la cantidad de bits disponibles en el puerto serie.
La función `Serial.readStringUntil('\n')` lee datos desde el buffer de recepción hasta encontrar el carácter indicado.

2.2 Lado del programa en Python

En este anexo se presentan solo las funciones necesarias para una comunicación mínima con un Arduino. para mayor información, visitar:
https://pyserial.readthedocs.io/en/latest/

Intalar con: `pip3 install pyserial`

2.2.1 Configuración del lado del programa en Python

```
import serial
```

La función de arriba importa el módulo serial.

```
ser = serial.Serial(port='/dev/ttyACM0', baudrate=9600, bytesize=serial.EIGHTBITS,
parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE, timeout=1,
```

xonxoff=False, rtscts=False, write_timeout=10, dsrdtr=False, inter_byte_timeout=None, exclusive=None)

La función de arriba inicializa y configura el puerto serial. En sistemas Linux, el puerto serial al cual está conectado el Arduino se mapea en la carpeta /dev. Para consultar todas las opciones de configuración, visitar: https://pyserial.readthedocs.io/en/latest/pyserial_api.html

2.2.2 Envío de datos

ser.write(cadena.encode("utf-8"))

cadena debe ser una variable de tipo string.

2.2.3 Recepción de datos

cadena = ser.readline().decode("utf-8")

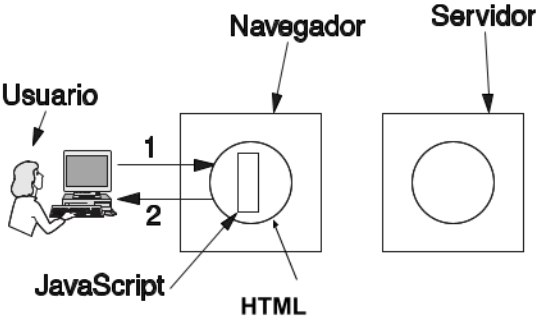
La función de arriba lee una línea del puerto serie (hasta encontrar un carácter de fin de línea \n). Si hay más de una línea en el puerto serie, se leerá solo la primera. La variable cadena debe ser de tipo string.

Cuidado!!: La función de arriba es bloqueante, por lo que no retornará hasta recibir una línea. Se sugiere utilizarse mediante hilos o tomando los recaudos para no bloquear el programa.

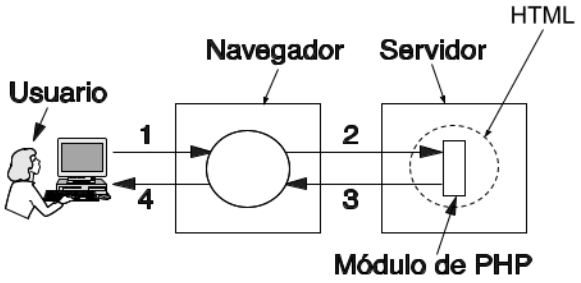
Anexo 3: Diseño de aplicaciones web con Flask

Este anexo asume que posee conocimientos sobre HTML y formularios (se adjunta un resumen sobre HTML y formularios en el Anexo 4).
La siguiente figura resume algunos de los lenguajes más utilizados para crear una aplicación web, entre ellos, HTML, JavaScript, CSS, PHP o ASP, etc.

Ejecución de instrucciones del lado del cliente:
JavaScript, VBScript, applets



Ejecución de instrucciones del lado del servidor
PHP, ASP, CGI



Flask es un framework que permite crear aplicaciones web (backend y frontend) utilizando Python. La característica principal es que permite invocar procedimientos escritos en Python desde una url. Los procedimientos se ejecutarán en la máquina servidor, mientras que las url son invocadas desde un cliente web (Google Chrome, Firefox, etc.). Por ejemplo, el siguiente código crea dos funciones que son llamadas desde las urls http://ip:5000/ y http://ip:5000/controlleds_9_10_11 respectivamente:

```
from flask import Flask
from flask import url_for
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route('/', methods=['GET'])
def index():
    ---código de la función---

@app.route('/controlleds_9_10_11', methods=['POST'])
def controlleds_9_10_11():
    ---código de la función---
```

Puede aprender de forma rápida el uso de Flask desde la página web de sus creadores: https://flask.palletsprojects.com/
Para realizar las consignas solicitadas en este trabajo práctico, se sugiere estudiar (como mínimo) las siguientes secciones de la página indicada arriba:

- Install Flask (utilice pip3 en lugar de pip)
- A Minimal Application
- Debug Mode
- Routing
- Static Files
- Rendering Templates
- Accessing Request Data

Además, deberá conocer el concepto de environment de Python. Se sugiere la siguiente fuente de información: https://www.w3schools.com/django/django_create_virtual_environment.php

Ejecutar la aplicación web:
Para ejecutar el servidor web creado con Flask, debe ejecutar en una términal:

flask --app nombre_aplicacion run

donde *nombre_aplicacion* es el nombre de un programa de Python (archivho .py) sin incluir ".py".
El comando anterior ejecutará el servidor. Deberá ejecutar un cliente web (ejemplo: Google Chrome, Firefox, etc) y en la barra de búsqueda cargar la url (http://127.0.0.1:5000/).
Note que el servidor web se ejecuta sobre el puerto 5000 y la IP de loopback, por lo que solo podrá llamar a la página web desde la misma computadora en la cual se ejecuta el servidor. Puede cambiar estos parámetros, por ejemplo, añadiendo --host=192.168.100.2 al comando anterior puede cambiar la IP sobre la cual correrá el servidor (debe ser la IP de la máquina donde se encuentra el archivo .py). Luego, podrá llamar a la página web desde cualquier computadora desde donde sea visible la IP indicada.

Características adicionales:
Flask permite crear websockets o utilizar JSON para crear aplicaciones dinámicas (característica no solicitada en los trabajos prácticos de esta asignatura).

Anexo 4: Introducción a HTML y formularios

Nota: este anexo presenta un breve resumen de HTML. Conocimientos sobre este lenguaje se dictan en la materia Redes de Computadoras. También puede encontrar tutoriales completos y gratuitos de dichos lenguajes y muchos otros en https://www.w3schools.com/

4.1 - Estructura de una página web escrita con HTML

HTML utiliza etiquetas para delimitar secciones. Por ejemplo, la etiqueta <head> inicia el código del encabezado de la página web, y </head> termina el código del encabezado. La siguiente es la estructura general de una página web:

```
<html>
<head>
Código de configuración, funciones de JavaScript, PHP, etc.
</head>
<body>
Código que implementa la página web (texto, botones, imágenes, PHP, etc.)
</body>
</html>
```

4.2 - Algunas etiquetas HTML que puede incluir en su página web

Texto: Texto a incluir
Imágenes: <img src="ruta al archivo relativa a la carpeta de trabajo"> o
<img src="url completa de la imagen">
Salto de línea:
Enlace: texto del enlace
elementos a centrar
<H1>Títulos de mayor tamaño</H1>
<H2>Títulos de segundo mayor tamaño</H2>, y así hasta <H6></H6>
 elementos Añade una división lógica a su página web. Todos los elementos entre las etiquetas   serán tratados como una unidad lógica.

4.3 - Ejemplo Implementación de un formulario HTML

Un formulario permite que el usuario interactúe con la página web. Por ejemplo, a través de botones, cuadros de texto, etc. Usualmente los datos de un formulario serán analizados por un programa en Javascript en el mismo navegador del cliente, y si cumplen los formatos esperados, serán enviados para ser procesados en el servidor por un programa escrito en PHP, ASP, etc.
Un formulario tiene la siguiente forma:

```
<form name=”formulario1” id= "formulario1_id" action="verificar.php"
method="POST">
<label for=”nombre">Ingrese su nombre: </label>
```

```
<input type="text"
name="nombre" id=”nombre_en_formulario1”>
<p><input type="submit"></p>
</form>
```

Veamos línea por línea:

`<form name=”formulario1” id= "formulario1_id" action="verificar.php" method="post" onsubmit="return validar();">`

name identifica de manera única al elemento dentro de un formulario (puede haber varios formularios en una página web y formularios dentro de formularios).

El *id* del formulario (en el ejemplo: formulario1_id) identifica de manera única al formulario dentro de un documento HTML. Es necesario para acceder este formulario desde funciones de JavaScript u otro lenguaje encriptado (No utilizado en este trabajo práctico).

action indica la url del documento donde se encuentra el código que implementa los procedimientos a ejecutar en el servidor (en este caso, un archivo php llamado “verificar.php”). Si se omite, los códigos se encuentran en el mismo archivo.

method=”POST”. Indica la forma o método a través del cual se enviarán los datos al archivo que contiene el código que procesará los datos. Los métodos pueden ser “get”, que indica que los datos se incluyen dentro de la url o “post”, que indica que los datos se incluyen en el cuerpo del paquete http.

`<input type="text" name="nombre" id=”nombre_en_formulario1”>`

input type: agrega un elemento al formulario. Los elementos pueden ser: cuadros de texto para que el usuario ingrese un texto (input type="text"), un botón para enviar el formulario (input type="submit"), campo de texto para que ingresar contraseñas (input type="password"), etc.

name identifica al elemento dentro del formulario. No puede haber dos elementos con igual nombre dentro de un mismo formulario, pero si pueden haber dos elementos con igual nombre en diferentes formularios.

id identifica al elemento dentro del documento HTML. No puede haber dos elementos con el mismo id dentro de un documento HTML, aunque estén en distintos formularios.

value se utiliza para dar o recuperar el valor del elemento. El valor dependerá del tipo de elemento. Por ejemplo, para un cuadro de texto, value será una cadena de caracteres.

`<label for=”nombre">Ingrese su nombre: </label>`

Asigna una etiqueta (texto descriptivo) a un elemento. En el ejemplo, nombre puede ser un cuadro de texto, un botón, un cuadro para contraseñas, etc.

```
<p><input type="submit"></p>
```

Botón de envío. Al apretarlo se llamará al archivo o procedimiento indicado por *action*.

Anexo 5: Simulador de Arduino UNO.

Existen varios simuladores para trabajar con placas Arduino. La cátedra sugiere utilizar https://wokwi.com/, ya que permite utilizar todas las librerías necesarias para implementar todos los trabajos prácticos de la materia.
La cátedra provee una simulación de un circuito implementado en el siguiente enlace: https://wokwi.com/projects/391281841803809793
Podrá escribir el código del mismo modo que en el IDE de Arduino sobre esta simulación. Para guardar el código escrito en la nube del simulador, debe crear una cuenta de usuario en Wokwi.
Podrá utilizar el monitor serial (similar al del IDE de Arduino) para simular la comunicación serial con la aplicación servidor.
Con la herramienta Library Manager podrá agregar librerías externas, por ejemplo FreeRTOS. Necesitará esta herramienta a partir del Trabajo Práctico Nº2.