ALMACENAMIENTO DE DATOS A TRAVÉS DE COMUNICACIÓN SERIAL Y REGEX EN PYTHON

M. en C. Esther Viridiana Vázquez Carmona
evazquezc1801@alumno.ipn.mx
M. en C. Rodrigo Vázquez López
rvazquezl1800@alumno.ipn.mx
M. en C. Luis Alberto Flores Montaño
luisfloresmontano@hotmail.com
Dr. Juan Carlos Herrera Lozada
jlozada@ipn.mx

Instituto Politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico en Cómputo

Boletín No. 83
1o. de marzo de 2021

Resumen

Actualmente uno de los propósitos importantes dentro de la investigación y la industria, es la optimización de procesos, específicamente en los que intervienen las tareas de supervisión, para ello, se han implementado los sistemas de monitoreo que tienen por objetivo resolver este tipo de problemáticas a través del uso de sensores. Este documento aborda la continuación de un trabajo previo sobre sistemas de monitoreo que inició con la construcción de un prototipo de variables climáticas en el que se interconectaron sensores de humedad, presión, temperatura y un sensor GPS, posteriormente se integró la parte de comunicación en el que los datos de los sensores son enviados a través de un sensor o transductor (antena 3DR) que toma los datos vía puerto serial. Para este trabajo se abarca la parte de almacenamiento en variables de los datos provenientes de dicha antena de telemetría, validando una expresión regular utilizando una sintaxis contenida en un patrón proveniente de este puerto.

1. Introducción

Hace algunos años se inició el desarrollo de los sistemas de monitoreo (observación continua del estado de una variable) sin embargo, hoy en día han ido mejorando con el fin de adaptarse a las nuevas tecnologías y servicios (Suhissa, s.f.), por ejemplo, la disponibilidad de redes de datos de alta velocidad, sensores de bajo costo, telemetría y el desarrollo de bibliotecas en Python y otros lenguajes que minimizan la interconexión entre estos. Como se mencionó anteriormente este trabajo es la continuación de un prototipo de variables climáticas que está integrado por módulos, el primer módulo corresponde a la interconexión de sensores (DHT-22, BMP-180 y NEO-6M), el segundo módulo

está encargado del envío de datos y recepción de estos. Anteriormente se trabajó con el envío de datos de los sensores que se realiza a través de una antena de telemetría (Vázquez, 2020), ahora se integra el tercer módulo que se encarga del almacenamiento de los datos que provienen de la antena receptora utilizando una expresión regular que facilita la búsqueda de patrones en una cadena (string tipo de dato en Python). A continuación, se describen algunos conceptos que integran el desarrollo de dicho módulo (Regex, s.f.).

1.1 Recepción de datos

La recepción de datos consiste en leer los datos provenientes desde un dispositivo o puerto, posteriormente almacenarlos y así poder manipularlos de acuerdo con nuestras necesidades. En este caso se utiliza el puerto serie, sin embargo, anteriormente este puerto tenía la desventaja de ser un puerto lento en comparación el puerto paralelo, actualmente estos puertos han sido sustituidos por nuevos puertos serie: USB, FireWire o Serial ATA.

1.2 Expresión regular

Una expresión regular proviene de la matemática de lenguajes formales que se utilizan frecuentemente en programación, son palabras formadas por caracteres que sirven como identificadores estas expresiones incluyen metacaracteres que funcionan para reconocer las alternativas de una palabra (ArcMap, s.f.), estos se pueden observar en la tabla 1. Adicionalmente se incluyen caracteres especiales llamados sets los cuales se describen en la tabla 2 (Python, 2001).

Tabla. 1.
Principales metacaracteres utilizados en una expresión regular

Metacarácter	Función
|	Funciona como un separador para palabra alternativas.
?	Indica que existe máximo una coincidencia del carác expresión que está antes de este.
*	Indica que existen cero o más coincidencias del carác expresión que está antes de este.
+	Indica que existe al menos una coincidencia del carác expresión que está antes de este.
{n}	Indica que existe al menos n coincidencias del carácter o exp que está antes de este.
{n, m}	Indica que existe n y m coincidencias del carácter o expresió está antes de este.
.	Funciona como comodín que suprime cualquier otro carácte
()	Funciona para agrupar términos que especifican el orden de

Tabla 2.
Principales conjuntos de sets utilizados en una expresión regular

Set	Función
[xyz]	Encuentra una coincidencia de acuerdo con alguno de los caracteres que se encuentran dentro del corchete.
[m-z]	Encuentra una coincidencia de acuerdo con alguno de los caracteres que se encuentran dentro del rango en el corchete.
[1-9]	Encuentra una coincidencia de acuerdo con alguno de los caracteres numéricos que se encuentran dentro del rango en corchete.
[^xyz]	Encuentra una coincidencia de acuerdo con alguno de los caracteres que NO se encuentran dentro del corchete.

1.3 Biblioteca Regex

Esta biblioteca se caracteriza por proporcionar operaciones de coincidencia de expresiones regulares que admiten modificadores, identificadores y caracteres de espacio en blanco (Soporte-platzi, 2015). En la tabla 3 se presentan los principales identificadores. La cual integra operaciones como el método search() que busca un patrón en un string, también incluye el método match() para devolver la posición si la coincidencia se encuentra al inicio, además el método finditer() a diferencia de los métodos antes mencionados se encarga de buscar todas las coincidencias y no solo la primera que observa solo una coincidencia. Como se mencionó anteriormente también existen un conjunto de caracteres llamados sets, que trabajan en conjunto con el método findall() de manera que devuelven un vector con substrings y no solo las posiciones (Rungta, 2020).

Tabla 3.
Principales conjuntos de identificación utilizados en pyhthon para la Biblioteca Regex

Identificador	Función
\d	Espera un dato integrado por un número o dígito.
\D	Espera un dato de cualquier tipo excepto un número o dígito
\s	Espera un dato integrado por un espacio, tabulación, nueva l
\S	Espera un dato de cualquier tipo excepto un espacio, tabulac nueva línea.
\w	Espera un dato integrado por letras o carácter alfanumérico.
\W	Espera un dato de cualquier tipo excepto un dato integrado p letras o carácter alfanumérico.
.	Espera un dato de cualquier tipo excepto letras (puntos).
\b	Espera un dato de cualquier tipo excepto una nueva línea.
\.	Espera un dato de cualquier tipo excepto puntos.

2. Desarrollo

Como se mencionó anteriormente se inició con el desarrollo de un prototipo que incluyen los sensores conectados a una placa Arduino que se encuentra conectada a su vez a una microcomputadora Raspberry Pi 3, posteriormente los datos se envían a través de una antena emisora de radio telemetría como se puede observar en la Figura 1. Ahora corresponde almacenar los datos provenientes de la antena receptora en diferentes variables por cada una de las señales a medir. Es por ello que se desarrolló un script en Python que consiste en leer la cadena de caracteres proveniente de Arduino y de la antena receptora, la cual contiene caracteres que no sirven y que se tienen que descartar, para solo guardar los valores numéricos.

Figura 1. Módulos que integran el sistema de monitoreo de variables climáticas. Inicia con el módulo de interconexi

2.2 Diseño de Software

Como se mencionó anteriormente, los datos de la placa Arduino que viajan a través de la antena emisora de telemetría y llegan a la antena receptora, presentan la estructura que se muestra en la figura 2.

```
b'Humedad,Temperatura,Presion,Altitud,Latitud,Longitud,Fecha,Hora,\r\n'
b'24.90,34.72,779.33,2159.85,0.00,0.00,0/0/2000,0:0:0,\r\n'
b'25.00,34.72,779.33,2159.87,0.00,0.00,0/0/2000,0:0:0,\r\n'
b'25.10,34.73,779.34,2159.76,0.00,0.00,0/0/2000,0:0:0,\r\n'
b'25.30,34.75,779.39,2159.21,0.00,0.00,0/0/2000,0:0:0,\r\n'
b'25.60,34.75,779.33,2159.82,0.00,0.00,0/0/2000,0:0:0,\r\n'
b''
```

Figura 2. Estructura de los datos que provienen de la antena emisora de telemetría y la placa Arduino.

El almacenamiento de estos, en este caso se realiza a través de un script en Python (W3schools, 1999) que lee los datos a través de una expresión regular y posteriormente estos valores se transforman; de manera que la estructura de este sea solo sea el valor numérico evitando los caracteres especiales que no son imprimibles o visibles en la consola como \r (retorno de carro o en inglés carriage return). La figura 3, muestra el desarrollo del algoritmo.

```
import serial, time
import numpy as array  #Bibliotecas necesarias para leer datos desde el puerto serie y el módulo Regex
import re

arduino=serial.Serial(port='/dev/ttyUSB0',
                      baudrate=57600,
                      parity=serial.PARITY_NONE,    #Parámetros para leer los datos desde
                      stopbits=serial.STOPBITS_ONE, #la antena de telemetría
                      bytesize=serial.EIGHTBITS,
                      timeout=1)
while True:

    rawString=arduino.readline().decode("utf-8")
    miCadena.append(rawString)
    temp=str(miCadena[i])
    i=i+1

    t_string=str(re.findall(r'\d*\.\d*',t_cadena)) #Se utiliza el método findall()
    h_string=str(re.findall(r'\d*\.\d*',h_cadena)) #Donde espera leer un retorno de carro seguido
    p_string=str(re.findall(r'\d*\.\d*',p_cadena)) #de un dígito, un punto y otro dígito dado que es
                                                   #un número de tipo flotante.

                                                   #Adicionalmente este valor se guarda en la variable
                                                   #que se encuentra despues de la coma.

    print("t vale",t_string)
    print("h vale",h_string)
    print("p vale",p_string)
```

Figura 3. Algoritmo que recibe los datos en forma de cadena desde el puerto serie USB, posteriormente realiza el al

párrafo

3. Resultados

Los datos son almacenados en variables, lo cual facilita su manipulación para su transmisión hacia o desde los actuadores/sensores. Estos datos se imprimen en pantalla para observar cada uno de sus valores, cada valor lo precede una letra que indica: T= temperatura, H=humedad y P=Presión, estos se muestran en la figura 4.

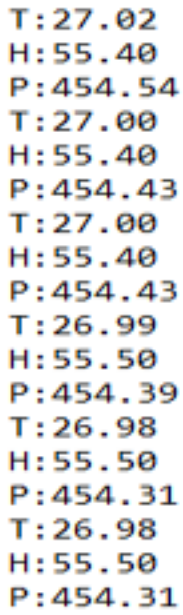

Figura 4. Datos que se obtienen de los sensores vistos desde el script en Python utilizando la biblioteca Regex.

Conclusiones

Se desarrolló un script en Python que interpreta una cadena de caracteres a través de la biblioteca Regex, la cual proporciona operaciones de coincidencia para expresiones regulares. Este script permite el almacenamiento de valores numéricos en diferentes variables, por tanto, la cantidad de bytes ocupados en memoria disminuye debido a que elimina secuencias de escape o símbolos innecesarios, cabe destacar que existen otros procedimientos para dicho almacenamiento. Además, esta estructura de dato formará parte del último módulo de este sistema de monitoreo para realizar el envío de datos hacía una plataforma IoT.

Referencias

<>

ArcMap. (**s.f.**). Metacaracteres que se utilizan para compilar expresiones regulares—ArcMap. *Arcgis.com* Recuperado el 14 de enero de 2021, de https://desktop.arcgis.com/es/arcmap/latest/extensions/data-reviewer/metacharacters-used-to-build-regular-expressions.htm

Python. (**2001**). *).re — Regular expression operations — Python 3.9.1 documentation. Python.org* Recuperado el 14 de enero de 2021, debr> https://docs.python.org/3/library/re.html

Regex. (**s.f.**). *Pypi.org* Recuperado el 15 de enero de 2021, de : https://pypi.org/project/regex/

Rungta, K. (**2020, enero 1**). *Python RegEx: re.match(), re.search(), re.findall() with Example. Guru99.com* Recuperado el 14 de enero de https://www.guru99.com/python-regular-expressions-complete-tutorial.html

Soporte-platzi. (**2015, julio 17**). *Guía de expresiones regulares en Pythonlibro,revista o nombre de la página wePlatzi.com* Recuperado el 14 de enero de 2021, de https://platzi.com/blog/expresiones-regulares-python/

Suhissa. (**s.f.**). *Sistemas de monitoreo - Suhissa.* Recuperado 14 de Enero de https://suhissa.com.mx/sistemas-de-monitoreo/

Vázquez, E. V., Vázquez, R., Montaño, A. & Herrera, J. C. (**2020, 1 julio**). *Adquisición de datos a través de comunicación serial utilizando el módulo de radio telemetría 3DR y PYTHONBoletín UPIITA*(79), 1–7. Recuperado de http://www.boletin.upiita.ipn.mx/index.php/11-numeros/1831-numero-79

W3schools. (**1999**). *Python RegExW3schools.com* Recuperado el 15 de enero de https://www.w3schools.com/python/python_regex.asp