|  |  |  |
| --- | --- | --- |
|  | دانشکده مهندسی کامپیوتر و فناوری اطلاعات | F:\Arm\AKUT.svg.png |

آزمایشگاه اینترنت اشیاء

گروه پلتفرم

**عنوان سند:**

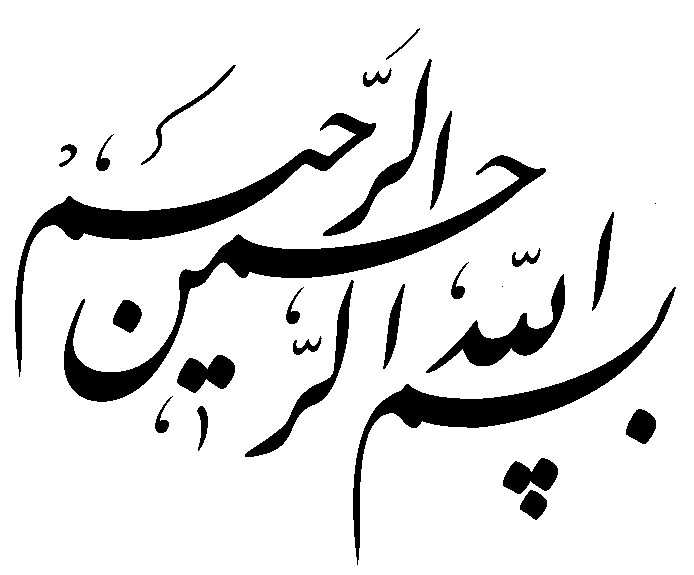
**کتابخانه­های ارائه شده برای اسکریپت نویسی**

**كد سند:**

**IoT-Imp-UserLib-v1.1**

**تاريخ:**

**14/01/97**



اطلاعات سند

|  |  |
| --- | --- |
| نام پروژه: | طراحي و پياده‌سازي پلتفرم اينترنت اشياء |
| نام سند: | کتابخانه­های ارائه شده برای اسکریپت نویسی |
| کد سند: | IoT-Imp-UserLib-v1.1 |
| وضعيت: | نهایی |
| تاريخ انتشار نهايي: | --- |
| نوع طبقه‌بندي سند: | محرمانه |

تاریخچه بازبینی‌ها

|  |  |  |  |
| --- | --- | --- | --- |
| ردیف | نام بازبینی کننده | تاریخ بازبینی | تغييرات |
| 1 | تيم فني | 18/9/96 | ویرایش بر اساس نظرات کارفرما |
| 2 | تیم فنی | 20/11/96 | افزودن کلاس های جدید |
| 3 | تیم فنی | 14/01/97 | ویرایش فنی و ادبی |
| 4 |  |  |  |

چکيده

این سند جز اسناد فاز طراحی می­باشد. در این سند طراحی کتابخانه­هایی که در اسکریپت نویسی در اختیار کاربر قرار خواهد گرفت تعریف و تشریح شده­اند. کاربر سیستم تنها توانایی استفاده از کتابخانه­های تعریف شده در این سند را دارا خواهد بود.

فهرست مطالب

[فصل 1: کتابخانه­های اسکریپت نویسی **5**](#_Toc510008125)

[1-1- تعریف کتابخانه­ها **5**](#_Toc510008126)

[1-1-1- کلاس Scenario: **5**](#_Toc510008127)

[1-1-2- کلاس db\_cr: **6**](#_Toc510008128)

[1-1-3- کلاس notification\_actions **7**](#_Toc510008134)

[1-1-4- کلاس ui\_geopy\_gps](#_Toc510008135) **[7](#_Toc510008135)**

[1-1-5- سناریوهای تست کتابخانه](#_Toc510008136) **[7](#_Toc510008136)**

1. کتابخانه­های اسکریپت نویسی
   1. تعریف کتابخانه­ها

با توجه به اینکه کاربران پلتفرم قابلیت اسکریپت نویسی دارند لیست کتابخانه­هایی که یک کاربر می­تواند در زبان پایتون استفاده کند در چهار کلاس Scenario، db\_cr، notification\_action و ui\_geopy\_gps تعریف شده­اند که در ادامه جزئیات هر کلاس ارائه گردیده است. در نهایت سناریوهایی نیز برای تست این کتابخانه­ها طراحی شده است که در انتهای گزارش ذکر شده­اند.

* + 1. کلاس Scenario:

کلاس Scenario شامل توابع دریافت داده، ارسال داده و توابع مربوط به زمان است.

* تابع sleep(seconds): اجرای کد را برای مدت زمان داده شده به تعویق می‌اندازد. واحد تاخیر آن بر حسب ثانیه است.
* تابع schedule(delay\_seconds, action\_fucntion, args): این تابع در ورودی میزان تاخیر (بر اساس ثانیه) ، تابع مورد نظر جهت اجرا و آرگومان‌های تابع را می­گیرد. در ادامه پس از گذشت تاخیر ذکر شده در ورودی، تابع را اجرا می‌کند.
* تابع wait\_for\_data(timeout):

این تابع برای دریافت داده است که می‌توان آن را به صورت سنکرون یا آسنکرون استفاده نمود. پارامتر آن به شرح زیر است:

1. timeout: میزان زمانی که منتظر می‌ماند تا داده را دریافت کند(بر اساس ثانیه).

این تابع پاسخی که از سرور دریافت می‌کند را به عنوان خروجی برمی­گرداند.

* تابع send\_to\_down\_link(message, timeout):

این تابع پیغام (داده) را روی یک پورت ارسال می‌کند و پیام تایید دریافت می‌کند. پارامترهای آن به شرح زیر است:

1. message: محتوای پیامی که ارسال می‌کند.
2. timeout: میزان زمانی که منتظر میماند تا داده را دریافت کند (بر اساس ثانیه).

پیام را ارسال می­کند و پاسخ تاییدی را که از سرور دریافت می‌کند به عنوان خروجی برمی­گرداند.

* + 1. کلاس db\_cr:

این قسمت برای ارتباط با پایگاه داده در نظر گرفته شده است و بر اساس نظر کارفرما تنها عملیات خواندن و ایجاد مجاز شمرده می­شوند.[[1]](#footnote-1)

* تابع create\_one(document): یک سند به صورت دیکشنری پایتون گرفته و آن را در پایگاه داده ذخیره می‌کند و id آن را برمی‌گرداند.
* تابع create\_many(document\_list): یک لیست از مستندات را گرفته و در پایگاه داده ذخیره کرده و لیست id آن‌ها را برمی‌گرداند.
* تابع read\_one(partial\_document): قسمتی از یک سند را گرفته و اولین سند دارای آن قسمت را برمی‌گرداند.
* تابع read\_many(partial\_document): قسمتی از یک سند را گرفته و همه اسناد دارای آن قسمت را برمی‌گرداند.
  + 1. کلاس notification\_actions

این قسمت برای اطلاع رسانی رویدادهای سیستم به کاربر است.

* تابع send\_email(host, port, username, password, sender, receivers, message): به ترتیب ip هاست سرور ارسال ایمیل ، پورت سرور ارسال ایمیل، نام کاربری، کلمه عبور، آدرس ایمیل ارسال کننده، لیست ایمیل دریافت‌کنندگان و متن ایمیل را گرفته و پیغام را از طریق سرور مشخص شده به لیست دریافت‌کنندگان ارسال می‌کند. در صورت موفقیت مقدار True برمیگرداند. در صورت عدم موفقیت و دریافت خطای smtplib.SMTPException، پیغام خطا را چاپ کرده و مقدار False برمی­گرداند.
* تابع send\_sms(username, password, from\_number, to\_number, message): به ترتیب نام کاربری، کلمه عبور، شماره تماس ارسال کننده، شماره تماس دریافت کننده و متن پیام را گرفته و توسط پنل مشخص شده در کد، پیامک را ارسال می‌کند. در صورت موفقیت مقدار True برمیگرداند. در صورت عدم موفقیت پیام خطا چاپ می‌شود و مقدار False برمی­گرداند.
  + 1. کلاس ui\_geopy\_gps

در این بخش از کتابخانه­ی geopy به منظور دریافت اطلاعات جغرافیایی استفاده شده است.

برای دریافت مختصات و آدرس کامل یک محل ازNominatim().geocode(“address”) استفاده میشود. خروجی آن شامل متغیر address به عنوان آدرس کامل، متغیر latitude عرض جغرافیایی، متغیر longitude طول جغرافیایی و متغیر raw به عنوان کل اطلاعات می­باشد.

برای دریافت آدرس با استفاده از مختصات از تابع Nominatim().reverse(latitude, longitude) استفاده می‌شود که متغیرهای address، latitude، longitude و raw مانند حالت قبل برگردانده می‌شوند.

* + 1. سناریوهای تست کتابخانه

در پوشه­ی scenarios، سناریوهایی برای تست توابع تعریف شده در این سند طراحی و پیاده سازی شده­اند. شایان ذکر است که این سناریوها تنها برای تست و آشنایی از نحوه کارکرد این توابع طراحی شده­اند و کاربرد دیگری ندارند. سناریوهای تست به شرح زیر می­باشند:

* سناریو یک- رخداد دریافت داده­ی سنسور شماره "x"بر روی شی "a" از طریق ایمیل اطلاع رسانی گردد: این سناریو از طریق تابع wait\_for\_data منتظر دریافت داده­ی سنسور شماره x از شیء a می­ماند. سپس در تابع action چک می­کند که داده دریافت شده برای سنسور x از شیء a باشد که در صورت درست بودن، با استفاده از تابع send\_email ایمیل را ارسال می­کند.
* سناریو دو- اگر داده سنسور شماره x بر روی شی a دریافت گردید، به اندازه w ثانیه منتظر بماند، آخرین مقدار سنسور y از شی b را خوانده و با مقدار سنسور x جمع کرده و در پایگاه داده ذخیره کند: این سناریو با استفاده از تابع wait\_for\_data منتظر دریافت داده­ی سنسور شماره x از شیء a می­ماند. سپس در تابع action چک می­کند که داده دریافت شده برای سنسور x از شیء a باشد و در این صورت w ثانیه منتظر می­ماند. در ادامه با استفاده از تابع read\_one آخرین مقدار سنسور y از شی b را خوانده و با مقدار دریافت شده از سنسور x جمع می­کند و در نهایت با استفاده از تابع create\_one مقدار مجموع را در پایگاه داده ذخیره می­کند.
* سناریو سه- اگر میانگین ۵ داده اخیر سنسور شماره x بر روی شی a بزرگتر از w بود یک ایمیل بفرستد: تابع read\_data\_from\_db با استفاده از تابع read\_many، 5 داده اخیر سنسور x از شیء a را دریافت می­کند. سپس میانگین این 5 داده محاسبه می­گردد و با مقدار w مقایسه می­شود. در صورتی که مقدار میانگین از w بزرگتر باشد تابع action فراخوانی شده و در آن ایمیل اطلاع­رسانی با استفاده از تابع send\_email ارسال می­گردد.
* سناریو چهار- اگر ترتیب دریافت داده به صورتی بود که در ابتدا داده سنسور x از شی a آمد و سپس داده سنسور y از شی b آمد، میانگین آنها را در پایگاه داده ذخیره کند: این سناریو با استفاده از تابع wait\_for\_data منتظر دریافت داده­ی سنسور شماره x از شیء a می­ماند. سپس در تابع action چک می­گردد که داده دریافت شده برای سنسور x از شیء a باشد و در این صورت به حالت بعدی می­رود. در این حالت نیز دوباره منتظر دریافت داده­ی سنسور شماره y از شیء b می­ماند. در ادامه در تابع action چک می­کند که داده دریافت شده برای سنسور y از شیء b باشد و در این صورت به حالت نهایی می­رود. در حالت نهایی میانگین داده­های دریافتی را محاسبه کرده و با استفاده از تابع create\_one داده­ی جدید را در پایگاه داده را ذخیره میکند.
* سناریو پنج- هر w ثانیه یک بار، میانگین 5 داده اخیر سنسور x از شی a را با میانگین 5 داده اخیر سنسور y از شی a مقایسه کند و اگر بزرگتر بود دستور z را ارسال کند: هر بار با استفاده از تابع read\_many در تابع read\_from\_db، 5 داده اخیر سنسور x از شیء a و همچنین 5 داده اخیر سنسور y از شیء a را دریافت کرده و میانگین آن­ها را محاسبه می­کند. سپس آن­ها را مقایسه کرده و اگر میانگین داده­های سنسور x از سنسور y بزرگتر بود، با استفاده از تابع send\_to\_down\_link یک دستور ارسال می­کند و سپس w ثانیه منتظر می­ماند و دوباره از اول این عملیات تکرار می­گردد.

1. در فایل db\_config تنظیمات مروبط به سرور پایگاه داده شامل ip و پورت سیستم اجرا کننده پایگاه داده و همچنین نام پایگاه داده موجود در آن(db\_name) و collection شامل داده‌ها(db\_collection) مشخص شده است که بعدا در اختیار مدیر پلتفرم قرار خواهد گرفت. [↑](#footnote-ref-1)