Professorship of Embedded Systems and Internet of Things
Department of Electrical and Computer Engineering
Technical University of Munich

Deliverable 1 - Part 1

IoT Remote Lab, SS 2020

M. Sc. Ege Korkan

? Exercise 1.1: Preliminary

- For all the questions below, make sure to work in a git directory. We evaluate how well you use git, meaning working in branches, not committing everything in one go, meaningful commit messages etc.
- Each file needs to be named the way they are specified in the questions. If the file name is wrong and we cannot find it, you will get no points. **For all the questions in the sheet, create a directory or folder called D1. All the following locations take this as root directory.**
- You are encouraged but not obliged to have a good README.md file that explains the directory structure.
- The questions in this section are not mandatory and are not part of a deliverable.

1. Make sure that you can use git comfortably. Here are some git and GitHub tutorials:
   - A thorough git tutorial from our sister chair RCS. `https://github.com/alxhoff/git-tutorial`
   - More simple and passive git tutorials from the WWW:
     - A long one: `https://git-scm.com/book/en/v2`
     - A visual and short-ish one: `http://marklodato.github.io/visual-git-guide/index-en.html`
     - More like a cheatsheet: `https://rogerdudler.github.io/git-guide/`
2. Make sure that you can write (and read) Markdown comfortably. Here is a really nice Markdown tutorial: `https://www.markdowntutorial.com/`

? Exercise 1.2: Let's write some JSON

For all the questions of this section, create a directory called D1_1 and save the files in there. We will start with simple JSON structures and make it gradually more complex. **When we say something like "The value should be string of a certain length, the characters in the string can be anything. This applies for other types as well.**

1. We will start with primitive types:
   1. Write a JSON file called `string.json` that contains a JSON string of maximum length 16.
   2. Write a JSON file called `number.json` that contains a JSON number with a precision of 10^{-3}.
   3. Write a JSON file called `boolean.json` that contains a JSON boolean.
   4. Write a JSON file called `null.json` that contains a JSON null.
2. Now arrays and objects:

   1. Write a JSON file called `array1.json` that has 4 items of strings of maximum length 1.
   2. Write a JSON file called `array2.json` that has 3 items of arrays that have 2 numbers of multiple of 2 and between 10 and 121.
   3. Write a JSON file called `array3.json` that has 4 items. The first item is a boolean, second one is an object with two keys named `key1` and `key2` which both have string values, third items is a negative number bigger than -42 and the fourth one is the array of the previous question.
   4. Write a JSON file called `object1.json` that is empty.
   5. Write a JSON file called `object2.json` that has four properties called `name`, `email`, `lastKnownLocation`, `nickname`. The name and nickname are strings, email should be in email format and lastKnownLocation should be an object with two keys of value type number called `x` and `y`.
   6. Write two JSON files called `objectF.json` and `objectC.json` that will represent the temperature values of a sensor. The sensor can deliver in Fahrenheit or Celsius and objectF corresponds to the Fahrenheit values and objectC corresponds to the Celsius values. The sensor can physically measure between $50°C$ and $150°C$. The JSON files are objects with two keys, `unit` and `value` which both have to be present. `unit` can be `Fahrenheit` or `Celsius` and the `value` is in the physical range of the relevant unit and has number type.

3. Let's write corresponding JSON Schemas for the payloads of the previous question. Name your schemas like `array1.schema.json`. Do not overspecify, i.e. if we say a string can be maximum length 10, specifying the minimum length of anything makes it invalid.

   Note1: For the `objectC.json` and `objectF.json`, you must write only 1 JSON Schema called `objectCF.schema.json`.

   Note2: We expect you to use the schema of array2 for array3 **without** copy pasting it.

   Note3: Your schemas should allow the previous payloads to be valid but should not accept invalid payloads.

4. The following subquestions are more advanced and definitely require you to learn some vocabularies you haven't seen in the lecture.

   1. A sensor measures the vibration of a mission critical, continuously running water pump every 30 minute and uploads it at the end of every day as a JSON array. The pump system has a natural frequency of 25 kHz where it gets unstable after a prolonged use of more than 3 days. Thus, you want ensure that there are all the values and also you want to check if there are frequency values in the range of 24 and 26 kHz in the daily uploaded array. Write the JSON Schema which would fail the validation if there are such values or an incorrect amount of values. Name it `frequency.schema.json` . Use zeroes to represent kHz.
   2. A location (or coordinates) data sent by a sensor is in the form of an array with the first item corresponding to the x axis, second to the y, third to the z. However, in addition to this, some sensors can also supply the location in other coordinate systems. These coordinates are in form objects with three keys with cylindrical system having `r`, `theta` (between `0` and `360`), and `z` and the spherical system having `r`, `theta` (between `0` and `360`) and `phi` (between `0` and `180`). Write a single JSON Schema that can validate such arrays, name it `coordinates.schema.json` .

      Note: You do not need to know these different coordinate systems but they are cool.