Weather - Station

Version - 0.1.0

Moser Martin

Table of Contents

Introduction

Disclaimer

Arduino IDE

At the time this document was written, the Arduino IDE with version 1.8.13 is used. If another version is used, the IDE may looks not the same or possible some steps are not one by one reproducible. But the IDE is in such a state that it does not change to much from version to version so it should not be a problem do follow along.

Known problems

Decimal places

The number of decimal places could be changed in the `SettingsWindow`. But the displayed number of decimals does not change if the window is closed. They will only change when the values are updated. For example, if the connection is open and the Arduino sends the new values, these will be displayed with the new number of decimals.

Libraries

Kotlinx Serialization

TornadoFX

JSerialComm

Shortcuts

Table 1. Shortcuts

Shortcut	Action
`Ctrl + C`	Connect to the Arduino
`Ctrl + C`	Disconnect from the Arduino

Setup

Java

The project uses the OpenJDK from Adopt. Currently version 11 (LTS) is used. The JDK for each Operating System could be downloaded here.

Please note, the application was only tested with the AdoptOpenJDK 11. It is possible to use another Java distributor or version. But it is not sure that the application or the libraries are (fully) compatible with the used Java. It is also not recommended to use a older version.

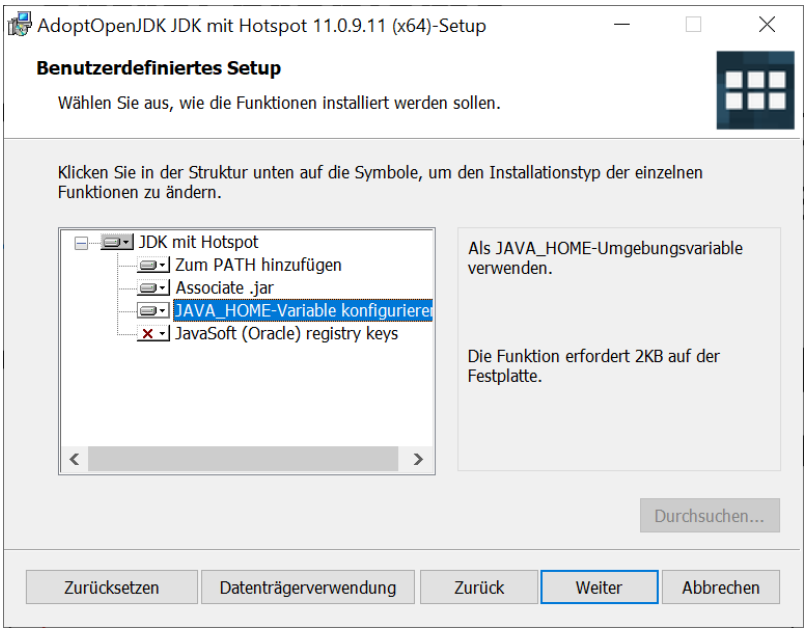

Figure 1. Adopt OpenJDK Installer

It is strongly recommended to create the `JAVA_HOME` variable. This variable points to the JDK so that other programs, like Gradle, could access the JDK. Adding could be done manually or using the installer. S_n = X_1 + … + X_n für n \ge 1 k To let the installer add it, it must be set to true in the custom setup step. In Figure 1 the required point is highlited in blue. If the symbol left to the text looks like this, the installer will set `JAVA_HOME`. If there is a red cross (like the symbol below) it will not set it. To enable it click on the symbol.

For the rest of the installation follow the wizard. It is not required to change anything else. SensorTypek

Arduino IDE

Used version at the time of writing Arduino SAMD Boards package = version 1.8.9 Arduino_MKRENV = version 1.1.0.

MKR board

To be able to program the Arduino using the official Arduino IDE, the board has to be installed first.

This could be done with the integrated board manager. Select 'Tools' → 'Board' → 'Boards Manager...''. This will open up a new window which provide functionality to add, update or remove boards. Search for a package called 'Arduino SAMD Boards (32-bits ARM Cortex M0+)'. This package is required for the Arduino MKR 1010 WiFi. Select the newest version of the library and install it.

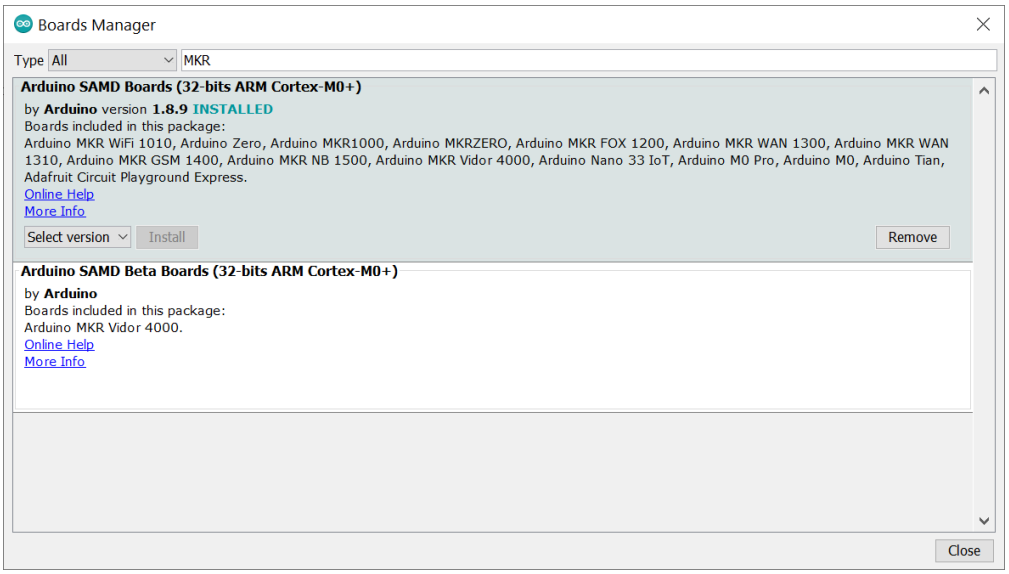

Figure 2. Arduino IDE Board Manager

After installation, the board must be selected. This is required, for the IDE to know for which controller the code must be compiled. Select 'Tools' → 'Board' → 'Arduino SAMD Boards (32-bits ARM Cortex M0+)' → 'Arduino MKR WiFi 1010'

MKR Env shield

The usage of the Env shield requires a library which provides the functions to read the sensors. It is an official Arduino library, so the installation could be done using the Arduino IDE. Select 'Tools' → 'Manage Libraries...'. This will open the Library Manager, where libraries could be installed, updated or uninstalled. Search for a library called 'Arduino_MKRENV' and install the newest version.

It is not required necessary to use the library but it is really recommended since it already implement all functions. If the library is not used all of the functions must be implemented by hand.

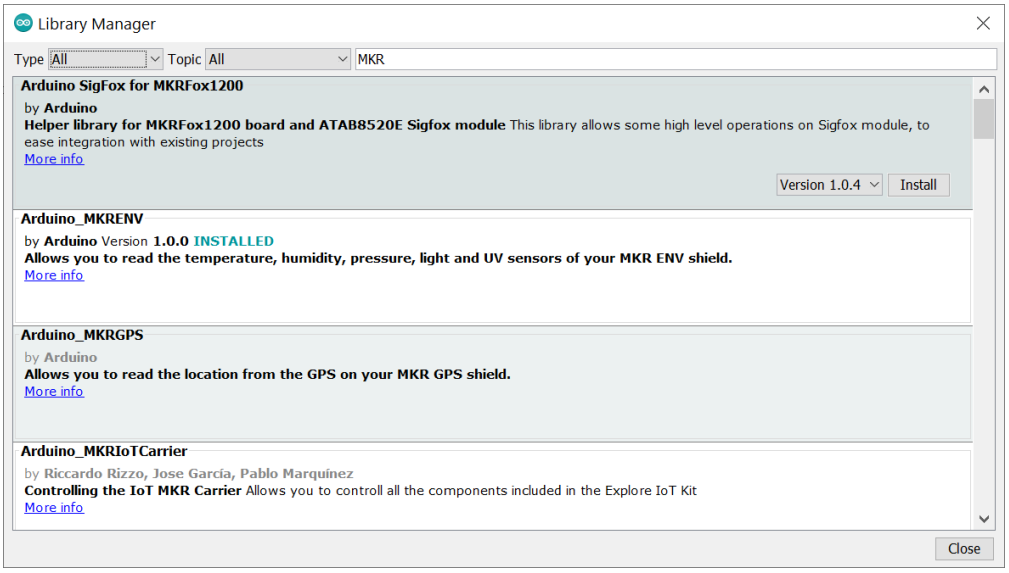

Figure 3. Arduino IDE Library Manager

Make

This is only required for Windows. Linux system have `make` already pre-installed. The program could simplify some workflow processes. For example, building the documentation or starting the lint function. It is not required to use `make` but it is highly recommended.

The easiest way in Windows is to use `Chocolatey`. There are a few other ways how it could be installed but those are not explained here. But Google provides enough links for this.

After `Chocolatey` is installed, a new terminal must be opened with administrator privileges. If the terminal is opened with default rights the installation process asks if it could access administrator privilges. But this fails sometimes. So it is best to search for cmd, right-click it and select `Run as administrator`.

Run `choco install make` to install the packet manager. This is the only required command for the installation.

Architecture

Sensor model

Sensors

Sensorlist

The list of available sensors is initialized using a JSON file. In this file all sensors and their required attributes are described. In order to read and parse the file correctly it must correspond exactly to the specified format.

The file contains an attribute called `sensors`. This is a list type and must contain all sensor definitions. A sensor definition is enclosed by a pair of opening and closing braces `{ ⋯ }`. Each definition requires an `id`, a `name`, a `value_type` and a `unit`. The ordering of these is important. As value type any of the defined constants in the `ValueType` enum could be used.

Listing 1. sensorlist-example

```
{
  "sensors": [ ①
    { ②
      "id": "f7a0d9cc-6f73-4090-9d1d-e8694f6c4c2c",
      "name": "Sensor 1",
      "value_type": "FLOAT",
      "unit": "°C"
    },
    {
      "id": "35958ba9-7447-4756-9b6e-700521a80a88",
      "name": "Sensor 2",
      "value_type": "FLOAT",
      "unit": "°C"
    }
  ]
}
```

① Start of the sensor list

② Start of a sensor definition

Line breaks and spaces are reduntant. All attributes of a sensor could also be in one line. But for clearnesses it is recommended to use the same formating.

The `id` and the `name` are unique values. In the application they are used to identify the right sensor.

Currently there is no mechanism to check if there are no duplicate values. It is the responsibility of the user to enter correct values. However, this is not intended to be permanent, only the current state of development.

Configruation

The whole configuration of the application is stored in a file called `config.properties`. The file is a

`ini` file and must follow some rules to be parsed correctly. In general, the file is split into different `sections`. This is not required but makes the file more readable since it gives a bit more structure to it. Each section contains key-value pairs.

Sections

A section is led by a `[section]` header where `section` is replaced by the name of it. The name of a section should be a string where the first character is upper case. Special characters are not allowed in the name. Spaces and points (.) are allowed but should be avoided as far as possible. In the best case a section name is a simple string which only consists of letters.

Key - value pairs

Keys are strings consisting only letters. They should be short and descriptive. Special characters or spaces are not allowed. Numbers should be avoided as far as possible. A key-value pair must always has the format `key = value`.

Sensor creation

The creation of the sensors (sensor objects) requires a few steps. At first the sensors file must be read. Then this file must be parsed from `String` to a JSON object. And last but not least the JSON object must be mapped to the sensors.

Parsing the `String` to JSON requires the knowledge of the **exact** scheme. The library requires a serializable class for parsing. In short the sensor list file could be described as a list of sensors. Due to the `Properties` in the sensor class it is not serializable and a new class is required. The new class is called `SensorType`. The list of all `SensorType` is stored in a class called `SensorList`.

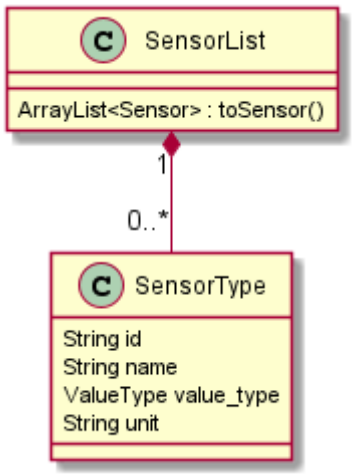

Figure 4. Sensor mapping classes

Figure 4 shows the UML of the required classes for the mapping of the sensor list. The `SensorList` class also contains a function to map all `SensorType` objects to `Sensor` objects. As already mentioned for the library to parse the `String` into JSON it is required that the classes are serialzable. To achive this the `@Serializable` annotation could be used.

```
import kotlinx.serialization.Serializable ①

@Serializable ②
data class SensorType(
  // ...
)

@Serializable
class SensorList(
  // ...
)
{}
```

① Required import to use the annotation

② Make the class serializable

Use - cases

Loading application properties

Case 1: Properties file could not be loaded

Closing the application

Case 1: Connection Status Connected

Show a confirmation dialog that the Arduino will be disconnected before the application will be closed.

Case 2: Connection Status anything except Connected

Close the application without a confirmation dialog.

Problems

TornadoFX ListView

The official releases of TornadoFX have some problems with Java 9+. One problem is the ListView. If an action is defined and the `clickCount` set, the application reports a problem when mouse is clicked. For example if `clickCount = 2` and an item is double clicked with the mouse, an error will be thrown.

This only happens with the mouse. If an item is selected and the click function activated by hitting the enter key everything works.

To use the `listView` properly with a newer version of Java (currently Java 11 is used), a newer version of TornadoFX must be used. This is (currently) only possible by using a snapshot.

The way that seems working is to use version `2.0.0-SNAPSHOT`. The solution is proposed here https://github.com/edvin/tornadofx/issues/899#issuecomment-488249680.

Listing 2. Use TornadoFX 2.0.0-SNAPSHOT

```
repositories {
    maven { url 'https://oss.sonatype.org/content/repositories/snapshots' }
}

dependencies {
    compile 'no.tornado:tornadofx:2.0.0-SNAPSHOT'
}
```

Confirmation dialog onClose

There are no good resources online for the implementation of a confirmation dialog on a close request. Due to this it took some time to implement this.

The main goal of the dialog is to ask the user is sure if he want to close the application. For example, if the application is still connected to an Arduino the user will be asked if the Arduino should be disconnected and the application closed.

The following code demonstrates a simple example how a confirmation dialog could be shown when the user clicks on the 'close' button. This is only a general usage example how the functionality could be implemented and not a real code sample from the application.

To show a dialog the `setOnCloseRequest` must be overridden. This could be done in the `onDock` function of the view.

Listing 3. Disable onCloseRequest

```
// ...

val close: Boolean = false

override fun onDock() {
  currentStage?.setOnCloseRequest { evt ->
    if (!close) evt.consume()
  }
}

// ...
```

The above example shows how close event could be disabled. If the event is `consumed` the application will not close. The next step is to replace the boolean variable by an alert dialog.

Listing 4. Show confirmation dialog onCloseRequest

```
override fun onDock() {
  currentStage?.setOnCloseRequest { evt ->
    val alert = Alert(AlertType.CONFIRMATION)
    alert.title = "Close the application"
    alert.headerText = "Are you sure you want to close the application?"
    alert.contentText = "You have some unsaved stuff. Are you sure you want to continue?"

    val okButton = ButtonType("Yes", ButtonBar.ButtonData.YES)
    val noButton = ButtonType("No", ButtonBar.ButtonData.NO)

    alert.buttonTypes.setAll(okButton, noButton)

    val result = alert.showAndWait()
    if (result.get() == okButton) {
      // ...
    } else {
      evt.consume()
    }
  }
}
```

The above code sample shows the new `onDock` function with the confirmation dialog. The example uses an alert with custom yes and no button.

Dokka dependencie problems

Dokka seems to make problems with missing dependencies. If everything is set up like in the official documentation see there are some problems.

Using dokka was tested on two devices and on both the same problem occured. It is not sure if this problem always happens or if this is just related to the current version.

There is an issue which provides a solution. The solution is proposed here https://github.com/Kotlin/dokka/issues/41#issuecomment-699723119.

Listing 5. Fix dokka dependency problems

```
repositories {
  //... other repos
  exclusiveContent {
    forRepository {
      maven {
        name = "JCenter"
        setUrl("https://jcenter.bintray.com/")
      }
    }
    filter {
      // Required for Dokka
      includeModule("org.jetbrains.kotlinx", "kotlinx-html-jvm")
      includeGroup("org.jetbrains.dokka")
      includeModule("org.jetbrains", "markdown")
    }
  }
}
```

Build the documentation

Use: `asciidoctor-pdf -r asciidoctor-diagram documentation.adoc`

Install rouge

```
gem install rouge
```