Prototyp - Smart Urban Farming

Beate Scheibel, Markus Zila

Modelling Language

Dieser Teil ist vorläufig abgeschlossen. Wir haben eine neue Modellierungssprache umgesetzt, mit der eine Location(Longitude, Latitude und Himmelsrichtung), ein Farming Module(Höhe,Breite,Länge) und die Pflanze(Spezies) und die verschiedenen “Needs” der Pflanze(Water,Sunlight,Temperature-jeweils low/medium/high) spezifiziert werden können. Diese Klassen können durch die zwei Beziehungsklasen - “In” zwischen Farming Module/Location und zwischen Crop/Farming Module und “Requires” zwischen Crop/Needs(Water,Light,Temperature)- miteinander verbunden werden. Nach erfolgreicher Modellierung der “Urban Farm” wird mithilfe der bereits vorhanden Funktion (Export XML) ein XML generiert.

Mechanisms and Algorithms

Nachdem das XML generiert wurde, wird dies mithilfe eines neu geschriebenen XML-Parsers in das Java-Programm eingelesen. Dabei werden alle Informationen die modelliert wurden, geparst und (in Zukunft) in einer Datenbank (wahrscheinlich mysql - aufgrund der Kompatibilität mit raspberry pie) gespeichert, da die Datenmenge für den Arbeitsspeicher zu groß ist. Die Datenbank wird aus drei Tables(Location, Farming Module, Crop) bestehen.

- Location (ID, Long, Lat, Direction)
- FarmingModule(ID, Lenght, Width, Height, Location(ID))
- Crop(ID,Species,Name,Water,Light,Temperature,FarmingModule(ID))

Das Rule-Based-System wird mithilfe von easy-rules (https://github.com/j-easy/easy-rules) implementiert. Ein kurzes easy-rules Beispiel ist bereits Teil des Java-Programmes.
Folgende Regeln sollen für die Beta-Abgabe implementiert werden:

- HIER KOMMEN UNSERE VORSCHLÄGE HIN!!
- ….
- If water_need = “high” and water_level <(bestimmte Menge - müssen wir noch austesten) - then warn_user(“Please water crop ‘Tomate’”)
- If temperature_need =”low” and temp_level > 20°(oder andere Zahl) - then warn_user(“Please turn on the AC’”)
- If light_need = “medium” and light_level > (evtl sonnenstunden pro tag aus wetter dienst rauslesen?) then warn_user(“Please close the blinds’”)

Derzeit gehen wir davon aus, dass die Rules “hard-gecoded” werden, im Laufe des Semesters kann dann eventuell noch fuzzy-rules implementiert werden.

Physical Implementation

Das eigentliche Farming Module (Plexiglas + Erde + Pflanzen) wird erst Teil der Beta-Abgabe, ebenso wie die Sensoren erst dann in die Implementation integriert werden.

Derzeitiges Setup:

- Raspberry Pi 3 B
    - Arduino IDE
    - Java jdk
    - Maven
    - SSH und FTP konfiguriert
    - RXTX Adaption and istallation
- ESP32 TTGO
    - Kontakte gelötet
- Java Server
    - EasyRules (RBS)
    - RXTX Framework for Serial Connection
    - Servlets (using Tomcat7)
    - Parser

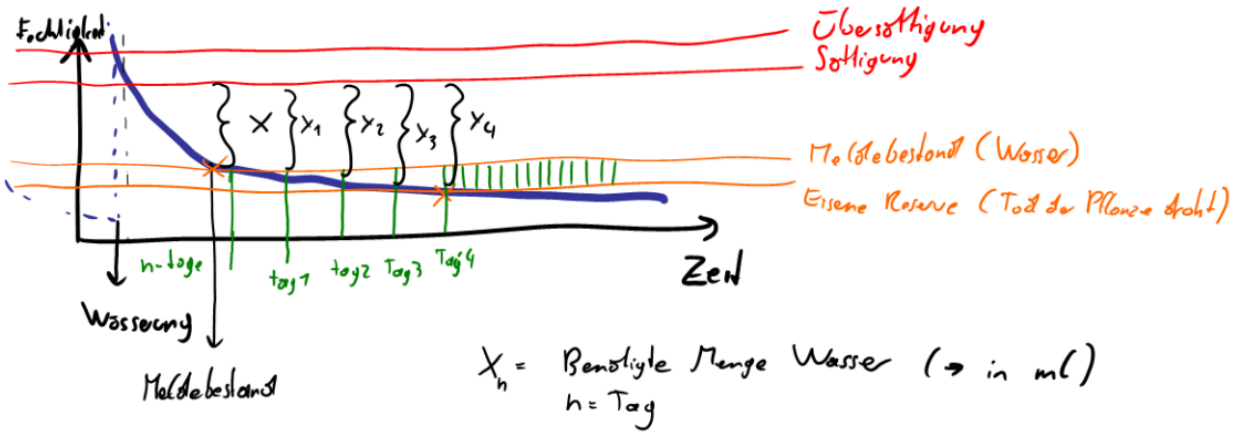
Probleme der Inbetriebnahme

- RXTX wurde nicht für Android ARM Platform (Linux distribution) konzipiert
    - Lösung: Modifikation von Oracle
    - https://blogs.oracle.com/jtc/serial-port-communication-for-java-se-embedded
- Vorinstallierte Arduino IDE auf den Raspberry hat nicht funktioniert
    - Lösung: Deinstallieren und neu Installieren
- RXTX muss auf den Computer wo Server läuft installiert werden
    - Es gibt unterschiedliche libraries, die ähnlich aber nicht gleich reagieren
- Developing Operation System: Windows10, Mac, Ausführungssystem Rasberrien (Linux, Debian)
    - Lösung, ständiges Prüfen und github shares
- Servlet Formular anbindung funktioniert nicht
    - Lösung: Annotation über Servlet „@MultipartConfig“
- Bytestream Parser funktioniert nicht
    - Lösung: Alternativ Parser mit Scanner Library
- Server Startup routine einstellen (kein main-Klasse zum ausführen)
    - Lösung: Startupservlet
- Arduino Connection blockiert sich selbst
    - Lösung: Arduino Thread Sigelten Funktionalität einrichten.

Rule Based System configuration:
Das RBS sollte einen Zyklus von ~ 10 Minuten haben, in der die Regeln neu überprüft werden, je nach Änderung in der Faktenbasis sind Aktionen zu setzen. Wir abstrahieren diese Aktionen mit Notifikation.

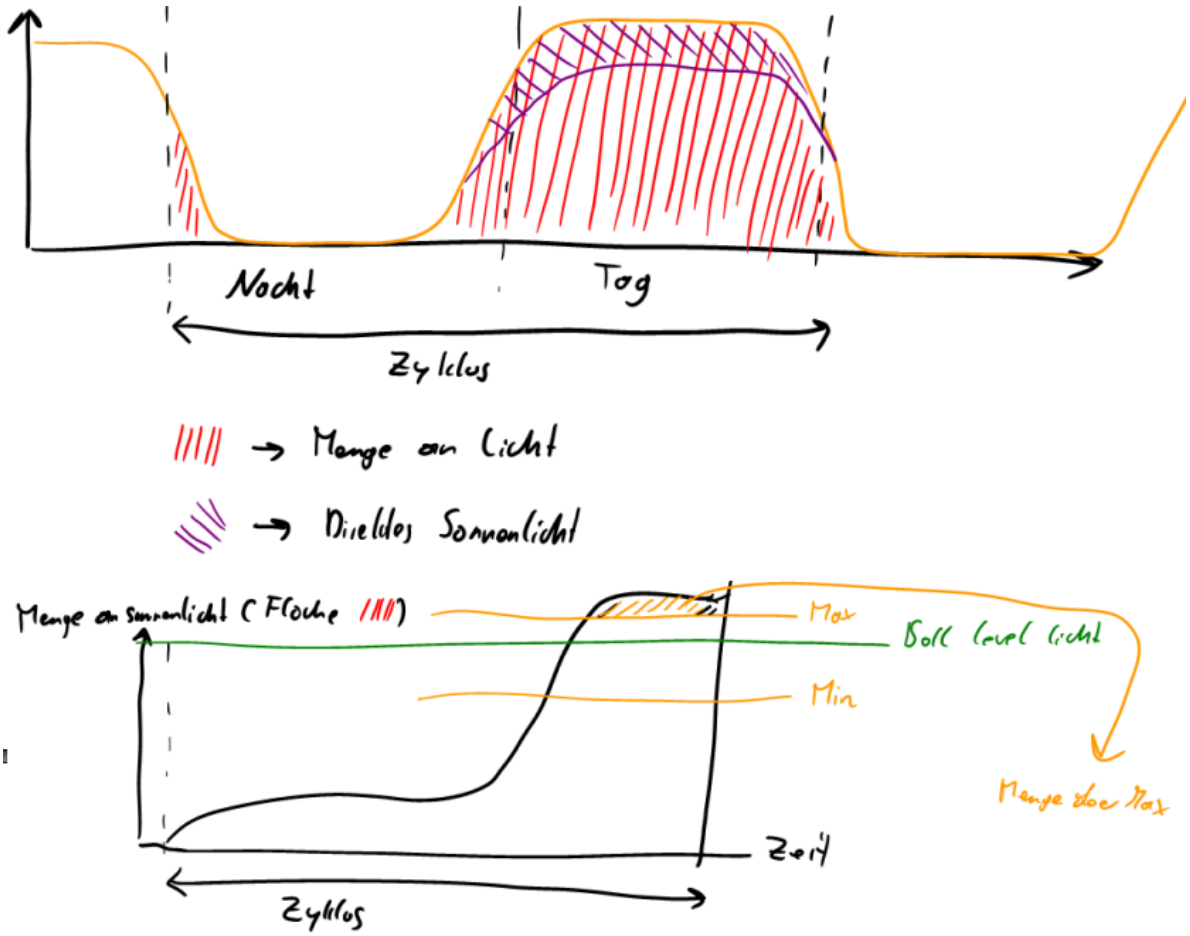
Beim Aspekt der Fechtehaltung ist das Konzept des RBS wie folgt:

- Input
    - Feuchtigkeitswert (aktuell+ Historie)
    - Zeitstempel der letzten Notifikation
    - Pflanzentype
- Reasoning
    - ? Sollte eine bewässerung stattfinden
    - ? sollte die Erde getrocknet werden (bei Regen oder übergießung)
    - ? wie lange ist die letzte Notifikation her (liegt der Wert im kritischen Bereich)
    - ? Wie viel muss gegossen werden, damit Sättigung eintritt



Beim Aspekt des Sonnenlichtes ist das Konzept des RBS wie folgt:

- Input
  - Menge des Lichtes
  - Pflanzentype
  - Notifcation Status
- Reasoning
  - ? soll die Pflanze abgeschattet / in die Sonne gestellt werden
  - ?Muss eine Notification gemacht werden
  - ? möglicherweise Open Wetter API Abfrage und kommendes Wetter feststellen



Beim Aspekt der Temperatur ist das Konzept des RBS wie folgt:

- Input
    - Temperaturverlauf
    - Pflanzentyp
    - Notifaction status
- Reasoning
    - ? überprüfe Max Temperatur Signifikant überschritten unterschritten