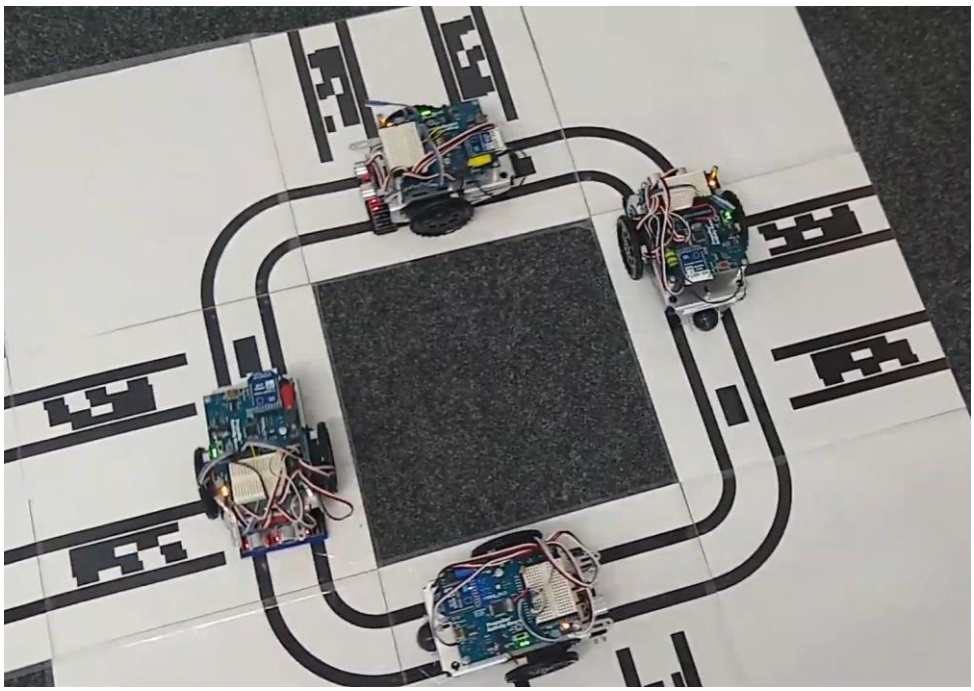
Code Overview



1. Projektstruktur

C-Project (see Section 2 and 3.2)

- Source code of the C++ project running on the ActivityBots
- uses the Library PopWare
- Follows the lines on the floor and waits for commands from the PC

Routing Table (see Section 2 and 3.1)

- Source Code of the Kotlin/Java Project running on the PC
- Manages communication between Bots and PC
- Executes the Algorithm and sends instructions to the individual Bots

Barcode Encoder

- Helper project to automatically generate printable roundabout segments
- Encodes the given node ID into a barcode and saves it into a svg or pdf file
- InkScape must be installed for the conversion from svg to pdf

Ant Shield

- 3d-printable shield for improving the performance of the LineFollower
- Blocks incoming light from the sides and therefore improves the detection rate of the black lines
- Available as Creo Part, Step and STL

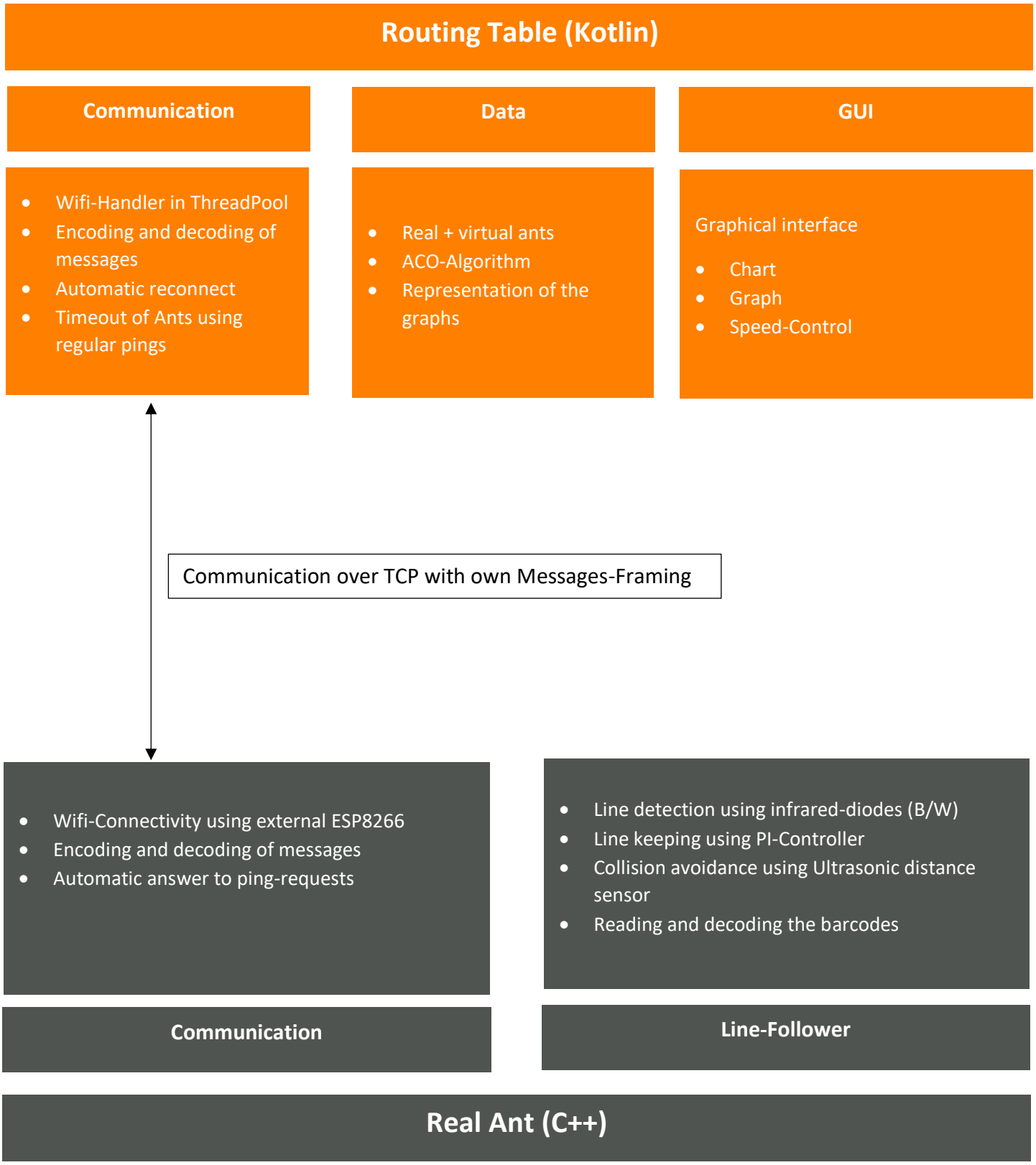
Printables

- This folder contains all printable rail-segments that make up the map the Ants drive on
- The segments have to printed on A3 Paper, cut to size, optionally they can be laminated to improve longevity
- Additionally, a printable calibration strip is given to check how well the LineFollower is working
- The individual sensors should only kick on in the center of the strips, otherwise it has to be recalibrated
- A representation of the map shown in the demo video can be found in the file 2013Map.xlsx

Documentation

- Further documentation
- Overview of the code with project structure as well as relevant code segments
- Assembly instructions of the ActivityBots hardware

2. System Overview



3. Relevant code segments for changes

3.1 Routing Table – Server (Kotlin)

- Connection Settings (IP-Space of the Bots)
  - de.jonasnick.antnet.routingtable.communication.ConnectionManager
    - `prefix`
- Predefined constants of the Algorithm
  - de.jonasnick.antnet.routingtable.data.`AntConstants`
- Predefined map
  - de.jonasnick.antnet.routingtable.data.map.`Maps`
- Function for setting the pheromones on target arrival
  - de.jonasnick.antnet.routingtable.data.map.Connection
    - `copyWithAdjustedPheromones`
- Function for pheromone evaporation in each tick
  - de.jonasnick.antnet.routingtable.data.AntManager
    - `tick -> "nodeManager.updateAllConnections"`
- Function for selecting the next edge to follow
  - de.jonasnick.antnet.routingtable.data.ants.AbstractAnt
    - `selectNext`

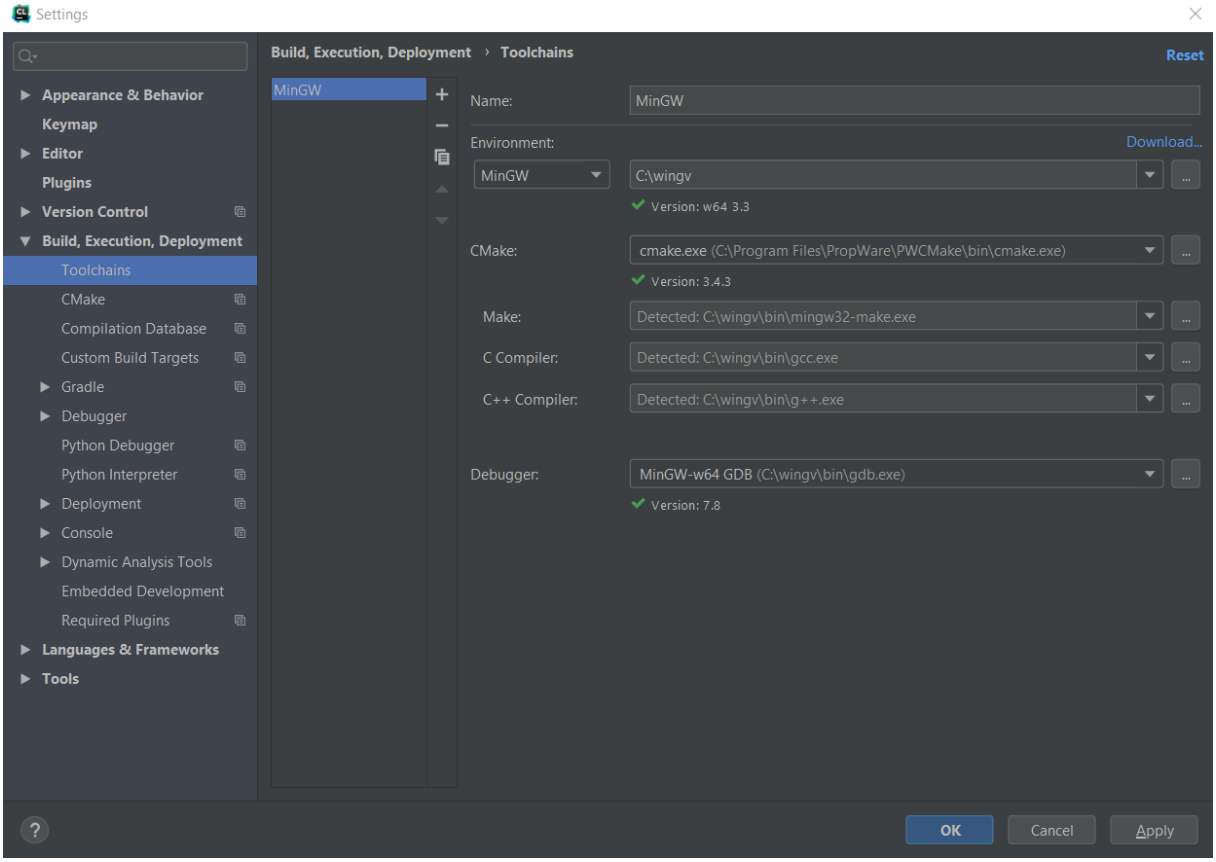
3.2 ActivityBot – Client (C++)

- Parameter of the PI controller
  - src\programms\linefollower\linefollower.h
    - `LineFollower::KID`
  - src\programms\linefollower\linefollower.cpp
    - `LineFollower::KP`
- Min distance of the ultrasonic sensors
  - src\programms\linefollower\linefollower.h
    - `MIN_DISTANCE`
- Controller for line keeping
  - src\programms\linefollower\linefollower.cpp
    - `LineFollower::run` (Zeilen 135-200)
- Processing of a read barcode
  - src\programms\linefollower\linefollower.cpp
    - `LineFollower::barcodeCallback`
- Reading of a barcode
  - src\programms\linefollower\linefollower.cpp
    - `LineFollower::run` (Zeilen 240-300)

4. ActivityBot + PropWare

Propware and this project can be executed on the ActivityBots using the the following instructions:

1. Installation of PropWare using the following instructions
   https://david.zemon.name/PropWare/#/download
2. Installation of MinGW, to be able to use `make`
   http://www.mingw.org/
3. When using *Jetbrains CLion:* Configuration of the Toolchain as shown:



4. Creation of a `/bin` directory and changing into it
   a. `mkdir bin`
   b. `cd bin`
5. Execution of the command (only has to be executed again on changes to CMakeList.txt)
   a. `cmake -G "Unix Makefiles" ..`
6. Connect an Activity Bots and set the switch it to Mode 1
7. Execute one these commands (still in /bin directory)
   a. `make debug`
      To flash the bots in the RAM.
      Program doesn't persist after switching the bot on and off
   b. `make debug-eeprom`
      To flash the bots in the EEPROM.
      Program persists even after shutdown or reset