Project Guide DBL Embedded Systems 2IO75
"The Fascinating Factory Floor"

Bachelor Program Software Science 2022 - 2023
Project Coordinator: Pieter Cuijpers

April 12, 2023

Contents

1 Introduction

The subject of the Design Based Learning project "DBL 2IO75" is the specification, design, and construction of a robot that is controlled by a microcontroller; a computer *embedded* in a larger robot. This means that you be will solving problems regarding the *programming* of the microcontroller, as well as the *electronics* connecting your microprocessor to the robot, as well as the *mechanics* of that robot. The robot you build will be part of a larger factory floor, so you as a designer team have to ensure that whatever you build fits well to that floor. Finally, as the interaction between software and the physical system is crucial for the correct functioning of a robot, you will need to think about all kinds of possible *faults and failures* that may occur in the physical world, and how your software should deal with them.

This DBL will give you the opportunity to train a number of software-skills as well as a number of skills particular to dealing with embedded systems, such as:

- Basic electronic and mechanical design;
  You will be building a robot in its entirety. This means including the electronics and mechanics involved. A number of basic and more expensive building blocks (such as the Fisher-Technics components used for the mechanics, and the RaspberryPi used as a controller) will be provided to you. Some cheap electronic components you may need will have to be bought at your groups own expense. When those expenses exceed 25 euros in total, they need to be approved by the student assistant tutoring your group.

- Interfacing between embedded software and hardware;
  You need to carefully design the low-level electronics needed to steer motors and read sensors in order to avoid overloading (and breaking, baking or frying) of the RaspberryPi's. Also, you need to write the low-level code associated with these electronics. As you will find out, 'the real world' never behaves entirely as you expect on the first try. This experience is likely to prove useful in many real projects later in your career.

- Managing complexity through model-based engineering;
  The job of a physicist is to create a model that mimicks the real-world as closely as possible. The job of an engineer is to create things in the real-world that mimick the behavior of a model as closely as possible. In this way, engineers make the world a more predictable and manageable place (which is considered good, when applied wisely).

  During your training in computer science, you will notice that we often use formal models of some kind to describe the software we are creating. In this course you will take your first steps in applying this knowledge, and very likely notice that it is more difficult than you might expect to make software that adheres to a model. Especially when you are developing control code for the interaction with physical phenomena, or perhaps when your group decides to feature a communication protocol to synchronise with the robots developed by others, as is common in embedded systems.

- Simulation in-the-loop design;
  One way to create software that does fit the model nicely, is by first testing it on a simulator of the final machine that it is for. This allows a development team to discover bugs before the actual machine is completely ready for testing. During the DBL you will mimick this by building a (simpler or more complicated) simulation model of your robot, and try to get your software working for both the simulation and the real thing. (One difference with industrial practice, is that development of the robot goes faster than in the real world, and that your simulator is probably a bit over-simplified.)

- Testing and dealing with faults and failures;
  "If anything can go wrong, it will." Murphy's law applies to any software engineering project, but even more in embedded systems. As you will quickly notice, mechanical and electrical

components are unreliable, which means your software needs to have features to recognize this and deal with the faults and failures that inevitably happen in any physical system. Modeling mechanical failures in your simulation environment may be very difficult, so you this is something that can only be really tested on the final product. The grading teachers *love* that kind of testing, especially the project coordinator! He would like to stimulate you to deal with faults and failures in a structured way, from the start of your design efforts. To show of how well you did this, there is a demo session at the end of the DBL in which you demonstrate that your robot is "teacher proof". This means that it not only works "as it should", but it can also recognize when and where your project coordinator, say, prodded a pencil into it, or blocks off a light-source, or whatever else is on their mind that day.

Apart from a gentle introduction into some of the software challenges of embedded systems, any design based learning project is also about teamwork and development of your professional skills. More explicitly, this DBL serves as a training to achieve the following goals:

- Structured cooperation;
  The multi-disciplinary character of an embedded systems project forms a natural environment to learn how to divide a project into subprojects and tasks, and practice communication between groups and individuals working on those subprojects. After this DBL, you have gained experience in how to run meetings between project members in a professional and structured way. In particular, you have chaired meetings, made minutes of meetings, given presentations, kept time-records, written self-reflections, and participated in the planning and organization of a group effort. SCRUM is a popular method in industry for coordinating software development, and throughout this DBL you will receive training in how to follow this approach.

- Reflection on your role in a group;
  Awareness of strengths and shortcomings is the first step towards being able to fully exploit those strengths and towards learning to deal with shortcomings. Furthermore, any team operates best when all members are aware of their role in a team. In order to make you more aware of your own strengths and shortcomings, we ask you to reflect on your role as a team member in this DBL. We suggest you use the *Belbin classification of roles* for this (GOOGLE IT), as they give a lightweight way to look at your own strengths and weaknesses from different viewpoints. But if you like, you can also suggest other personality tests for your group. Even though the scientific value of these tests is sometimes (often?) debatable, the discussions they provoke within the group can become quite insightful, especially when the group is able to create a safe atmosphere in which everyone sincerely shows interests in the others' views. Furthermore, we would like you to set some personal goals for this DBL in which those strengths are further developed and those shortcomings remedied. In the standard SCRUM workflow, the retrospective held at several points during the project is a nice time for such reflections.

- Structured documentation and communication of decisions and results;
  At the core of a scientific way of working, lies a scientific way of reporting. By reporting on our efforts we can achieve a number of things. For example, we can show progress to our superiors, who need to make strategic decisions on how we should spend our time. We can also use it as a historic record of our actions, to find out what went wrong after a malfunction occurred. Or we can use it as a basis for explaining a design to customers, users, trainers, maintenance engineers, etc. Finally, building structured documentation and writing reports and papers is a way to structure our own thoughts and improve our own thinking process. Proper scientific reporting focuses on the one hand on rationale behind all decisions made, but also gives enough details to ensure the process is reproducible by others.

  During this DBL, we intend to keep the development process agile and the reporting light-weight and close to the development. Further on in this guideline, you will find tips on how to keep your reporting light-weight by writing comments on your work and keeping

logs on-the-go, and by tracking your activities as a group thorough a list of decisions and a minimal architecture document.

- Presentation skills;
  Apart from written documentation, an engineer will also often find themselves in a situation where one needs to either inform or persuade others of their ideas. In those situations, which can range from formal or informal meetings, to conference talks, to television and YouTube performances, it is vital to be able to engage your audience, to establish a connection, and often also to deliver a clear, concise and well-structured story under the pressure of a live performance. As part of this DBL, there will be workshops during which you receive feedback on these skills, and ample opportunity to practice.

Finally, in order to grade your work, the grading teachers use a rubric explained at the end of this guide. Keep an eye on it as you go along during the project! At the start of the project, the rubric is not yet complete. You are asked to fill some parts of it in as a group, setting your own learning goals and focus points, and make explicit what you consider a sign of good quality. That part of the rubric may evolve a bit during the first part of the project.

2 The Project Assignment

The main project assignment, is to simply build an embedded system - a robot in this case - together with your group members. There is a basic assignment which you can follow, but you are encouraged to use your own creativity and come up with alternative ideas and challenges.

The robot you build will become part of a larger factory floor on which the robots of other groups also operate. This means you have to adapt your design to fit the floor, but it also gives opportunities for additional challenges.

2.1 Basic assignment: a sorting robot

The center-piece of the assignment is a conveyor belt system, called the *factory floor*, provided by the practicum coordinator. On this factory floor, there are black and white disks being moved around. The goal of the basic assignment is to build a *sorting robot* that is able to pick those disks from the belt and sort them for color, putting the white disks in one bucket and putting the black disks in another. Ideally, the robot should be designed in such a way that it takes the behavior of other robots on the belt into account, and maintains a 'fair' distribution of disks among the robots without knowing how many robots are in operation at any given time. For this, the robots will have to communicate with one another somehow. But not all groups will want to implement that last part, so that is a topic we'll discuss later.

Typically, a simple sorting robot will have a pushing mechanism for removing disks from the belt, a light sensor for determining the presence of a disk and another light sensor for determining its color (black/white). Finally, there will be a mechanism for putting a disk in one bucket or another. These actuators and sensors will be connected by a piece of software that implements a collecting and sorting strategy.

Examples of how to use light sensors to detect the presence of disks can be found online, but some experimentation will be necessary to figure out how to use light sensors to determine the color (black or white) of a disk. *As a hint, use the fact that a white disk reflects considerably more light than a black one.*

This scenario sketches the 'happy flow' of the robot; all behavior that is necessary for it to achieve its function. However, during this DBL you will notice that physical machines seldom follow the intended scenario exactly, and you are expected to consider the possibility of faults and failures of the electronics and mechanics (as well as purposeful obstructions by a demonic grading teacher during demos). A simple sorting robot will also have functionality in its software, like time-outs and other measurements, to ascertain that the physical robot is still functioning according to expectations. Furthermore, in case your group chooses to implement communication strategies, the robot software should report malfunctions to the other robots on the factory floor, and take appropriate action itself. You cannot possibly catch all possible faults and failures, but a good design includes a study of which faults and failures might occur, which effects this will have on the operation of the robot, and includes remedies for the ones deemed most important.

2.2 General requirements

In order to stimulate creativity and simply having fun during this DBL, you are encouraged to think of any alternative disk-manipulating robot. It doesn't need to sort; it may perform other functions as well in whatever way you like, as long as it involves moving objects off (and possibly back on-to) the factory floor in an intelligible way. Of course, because as educators we have to keep an eye on the difficulty level of the assignments, the requirements specification for any alternative robot should be handed in for appraisal before the deadline indicated in the next section, and explained using a small elevator pitch. (So far, we have never needed to reject any ideas for alternative robots. Sometimes, we just increased or decreased the difficulty level a little bit.)

When writing down your requirement specification, have a look at the material of the course 2IX20 Software Specification. You may find some inspiration there!

Both the standard sorting robot, as well as any alternative robot, have to abide to the following requirements:

1. The robot *mechanically* moves items with varying optical properties (black and white disks - or do you have alternative suggestions?) according to some verifiable, non-trivial (in the eye of the practical coordinator) purpose. "Verifiable" meaning that the operator of the robot can distinguish correct operation from incorrect operation of the robot, and "purpose" meaning that there is actually something to distinguish.

2. The operation of the robot involves the use of motors and sensors in a meaningful way. A minimum requirement is that the robot distinguishes the varying optical properties of the items and processes that information in a meaningful way. Usually, this is because the purpose of the robot is somehow related to these properties. For example, in the standard sorting robot, light sensors are used to detect the presence of a disk and the color (black/white) of a disk, and motors are used to push a disk onto a conveyor belt, and to drive the conveyor belt towards either of two buckets, one for white and one for black disks. Additional sensors may be used to verify if all the parts of the robot are operating correctly.

3. For the mechanics of the robot, you are restricted to using the FisherTechnic parts provided to you at the start of the practical; *unless* you specify clearly and exactly which other mechanical parts you would like to use and get approval from your tutors to use them (getting approval is usually not too difficult - this requirement is just there to make sure you think about what you need). The size of the robot should not exceed two ground-plates. The design of the factory determines where disks may be off-loaded precisely. You cannot alter the factory floor, so study that design!

4. Regarding the electronics, you are expected to use the RaspberryPi's provided to you, but you may (and will need to) extend this with other electronics to drive motors and read sensors. If you want to use different processing boards, like the Arduino or additional RaspberryPi's, this is okay, provided you get approval for your idea from your tutors first. You may need to buy or borrow some of the electronics you need for this course, but the most expensive and hard-to-get parts are provided for, if you want to buy special components (other than the regular wires, resistors, and small power sources) ask your tutor for approval. *Don't destroy the RaspberryPi provided to you.* The electronics of a RaspberryPi are quite sensitive to overloading, and for example connecting a motor directly to the output pins of a RaspberryPi can cause currents that fry the chips of the Pi (Google is your friend if you want to learn how to protect it)... Your tutor will want a functioning Pi back at the end of the course!

5. In principle, the robot performs its function autonomously. There may be an initialization phase, in which communication between user and robot is allowed. But after initialization, the user should not need to interfere with the workings of the robot unless a fault or failure is detected.

6. The robot *detects mechanical and electronic failures*, such as blocking of wheels and other obstructions, loose wiring, varying lighting conditions, etc. Detection of these failures should be communicated by the robot to the user and to the other robots, and should result in appropriate follow-up behavior by the robot. Also signals from other robots that a failure has occurred should be treated appropriately. Typically, "appropriate" means that the robot stops its operation and gives instruction to the user how to resolve the situation (in an as detailed way as possible), but sometimes more intelligent actions may be possible. Care has to be taken, of course, that a sudden stop by one robot does not unnecessarily interfere with the functioning of other robots in the factory. Take this into account when designing the protocol mentioned above.

7. The robot *reports on its internal state*, or allows other kinds of logging that make it possible for the developer to run testing scenarios on it. The groups demonstrate this by showing testing scenarios that lead the robot into as many a-priory predicted states as possible.

8. At the end of the project, the robot must be demonstrated to work correctly. It is wise to make early recordings of the correct working of (parts of) your robot as evidence, in case something unfortunate happens at the 'moment of truth', e.g. an irreparable malfunction at the final demonstration. If you have evidence of earlier successes, the grading teacher will be more lenient towards interpreting your rubric.

During the design process, you should keep an eye on all these, and possibly (likely) other additional requirements you come up with in the process. Of course, *design decisions regarding these points should be easily traceable in your reporting.* In your mid-term presentation you will reflect on these points, and in final poster you point out the most important decisions with references to your logs and measurements. If one of the above is not met, your group needs to explain why not, both technically (what is wrong with the robot) and process-wise (why did your team fail to meet the specification).

2.3 Set your own challenges

Implementing the general requirements above while following a decent engineering process and SCRUM workflow should be sufficient for any group to pass this course. However, in order to obtain a reasonably high grade, it is expected that you extend the challenge somewhat for yourselves. Further on in this guide, you'll find a rubric that makes the precise grading process explicit, but let's first discuss the kind of challenges that can be used as additional challenges.

As you read through these challenges, and certainly once you've picked one or two to focus on, try to formulate exactly what you consider signs of quality, both in the end-product as well as in the engineering process that is followed for this challenge? Write them down on the back of your personal group rubric, and pitch them to your tutor and to the practical coordinator to see whether they agree.

2.3.1 Communication protocols

The disks are delivered to the robot by a factory-floor conveyor belt. In the communication challenge, the robot needs to follow a protocol for removing items from or placing items on this factory belt, specified in cooperation with the other groups that follow this challenge. *This protocol assures that - over the course of a specified time of operation (say 5 minutes) - the difference between the amount of items that has been processed by each of the robots remains within a specified bound (say 3 items).*

As an extension to the challenge, one of the groups is expected to take the lead to create this protocol. It is expected to be a master-slave protocol communicating over Ethernet. This special group is expected to also implement that master on a separate Raspberry Pi.

2.3.2 Simulation-in-the-loop testing

A popular good practice in embedded systems industry, is to test the control software of an embedded system, before the actual physical system is built.

If your group wants to follow this best practice, there are many many ways to simulate the behavior of a real robot using software. There are extensive simulation packages like Matlab, but also a very simple Java program that accepts the same inputs as the real robot and mimicks outputs or interacts with a human to mimick outputs can be considered a simulator. In the past, groups thought of building a digital-twin robot using MineCraft, or visualizing it using Unity.

In practice, using a simulator for testing means that you have to be able to run your control software either on the real Raspberry Pi or on an emulator, and that you can couple the inputs

and outputs to the simulation software. Good luck figuring out how! (It is not forbidden to talk to each-other and to other groups on how they do this :-)

Perhaps as a small encouragement: last year some groups were saved by the fact that they had a simulation which they could demonstrate their software on, because their physical robot broke down on demo day!

2.3.3 Model-driven design and testing

Another good practice related to testing, and to ensuring quality software in general, is to use higher-level modeling to describe the workings of your software. For example, after first describing your software as a state-machine, you can design testing procedures that ensure every state of your software is actually visited by some test. This is called ensuring test coverage (look it up :-) there are multiple kinds of coverage, not only by state, but also by transition, function call, etc. etc.) Following clearly described testing procedures that are based on global software models is certainly a sign of a good engineering procedure.

Another possible use of model-driven design, is for example to describe the communication protocol suggested earlier as a parallel composition of automata and prove that it will never cause robots to get into a deadlock, or trigger other unwanted behavior. Software verification using high-level models is an advanced approach to software engineering - especially when combined with the next possible challenge: code generation.

2.3.4 Code generation

When you come to think of it, even writing code in Java and then compiling it, is a kind of code generation. After all, you are generating machine code from a higher level language. Nevertheless, when we say code generation, we usually mean the generation of code from a much higher level description of the software, such as an automaton model, a message sequence chart, or a UML model. In the past, students have been using the Dezyne language developed by Verum to describe the control software of their robot as a composition of event-based block descriptions. Those models of the control software could be verified for simple properties, and furthermore they could be translated into C code which - after adding a bit of hand-written drivers for motors and sensors - would steer the robot.

This type of software development is hard, but also satisfying when it works, because it means that you can easily adapt your code by making only small changes to the higher level models, and you know that your code is error free by design. The only testing that is then needed in the end, is meant to make sure that the hand-written parts are operating correctly. The rest can be done within (for example) the Dezyne environment. Feel free to investigate other tools for code generation as well, of course.

2.3.5 Dedicated languages like RUST

In principle, the choice of programming language is up to the groups. But you should realize that some languages better support an embedded systems engineering workflow, while others are more suited for large software projects, or for data processing. One modern language that we would like you to consider because the practical coordinator is interested in its capabilities regarding test-driven development, is RUST. This language allows you to think of software development in a way that starts out from the question: how are we going to test our software? It would be interesting to hear from groups that use RUST and its suggested development workflow, and their reflection on whether adopting the programming principles behind RUST has actually helped them or not.

Of course, if you have another language that you would like to promote in your group, please do so, but do keep an eye on whether it is suitable, and do formulate what you expect this language will bring you!

3 Reporting

As mentioned before, good reporting is fundamental to good engineering and scientific practice. Reporting suits several different needs, which require different points of focus. In particular, reporting should make your efforts *reproducible* for others, meaning that you should be aware of what details are relevant to someone trying to mimic your actions in order to get a similar result. Furthermore, reporting should help to evaluate your approach and may be used at a later point to maintain your engineering product. As such, it should contain a complete record of design decisions, as well as the rationale behind those decisions. Generally speaking, the why is more important than the what, for understanding a design.

In order to keep the overhead of reporting to a minimum, it is important that during your SCRUM efforts you do three things.

- Keep track of all decisions made during and outside your meetings, including the rationale behind these decisions, possible alternatives, expected risks, and the person(s) responsible for executing the decision. In order to explain the rationale behind decisions, add a set of architectural pictures of the design. For a good overview of the progress of the group, keep track of these decisions in an easily accessible document or git repository. Naturally, some of this information also will show up in duplicate as comments in your code.
- Keep a log-book on your personal actions. This logbook should have records of how much time you spent on what, and provide background material on the decisions mentioned in the decision list. The logbook should be detailed enough for others to take over your work at any point in time, or reproduce it at a later point. For software developers, one way to keep proper logs is by writing extensive commit messages in a GIT repository.
- You need to present your intermediate and final designs for an audience, in the form of a presentation and in the form of a poster, but also you need to hand-in all the logbooks and write a short explanation of your design in the form of a poster. This poster should contain architectural pictures as well as explanation of the most important design decisions, and there should be references to log-book entries and the design-decision document whenever relevant.

At relevant times during the project, you will be asked to upload the current status of your most important documents to CANVAS. Not everything, but everything that is relevant so that your student assistant tutors can judge your progress.

3.1 A note on the V-model in reporting

In Figure 1, you see the famous V-model for software engineering depicted. In fact, this model can be applied to any engineering design cycle, including mechanical, electrical, and embedded systems like the one you are going to work on during this practical. The V-model provides a useful workflow, especially when you are in a well-known domain and know very well what to expect upfront. However, if you are in a new or rapidly changing domain, the tendency is to use much shorter cycles than those suggested in the figure.

During this DBL, you are going to work in a SCRUM-like fashion, which (in a loose way) means you try to go through an entire V-model in a very short time (a single sprint), touching on all aspects of from system-level requirements to implementation back up-to validation testing, but for only a part of the system your are going to demo at the end of the sprint. So, even though you will, strictly speaking, not be following the V-model during the DBL, it is a good idea to read up on it, and be aware of its structure and naming-conventions when you report on your work in decision lists and logbooks. The V-model forms a common point of reference to talk about designing systems for most software engineers in industrial practice. Also, you will notice that the documents you produce do adhere to the different phases of the V-model. They are not exactly written in that order, but they do depend on each-other in the way that is indicated.

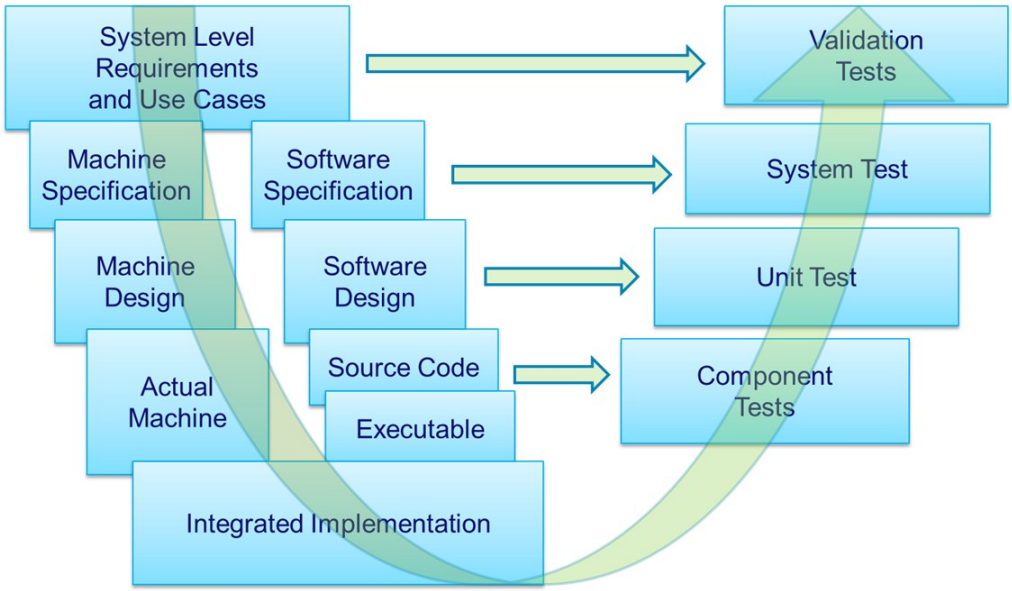

Figure 1: The V-model, not really followed in SCRUM, but still useful as a reference

Discuss the V-model and your preferred style and structure of reporting with your fellow group members, and decide how you will go about this. What is relevant for you and what is not? That could very well be your first decision in the list of decisions.

4 Deadlines and deliverables

This section has been added for the sake of giving a complete description of the DBL. In case of ambiguities, the information and times mentioned on CANVAS are leading.

4.1 Mon Apr. 24 : Kick-off (Aud 3, 13:30 - 15:30)

During the kick-off presentation the practical coordinator will tell you pretty much the same things as are in this guide, but of course he will also explain more details and give you a feel for the intention behind some of the rules outlined here... Presence at the kick-off is mandatory.

4.2 Mon Apr. 24 : First meeting (OGO Rooms, 15:30 - 17:30)

Directly after the kick-off, you and your group will go to your assigned OGO rooms for your first meeting. See CANVAS for the precise group formation and scheduling of rooms.

You are expected to work 16 hours a week on this DBL, but for only 8 hours an OGO room has been reserved. Some groups will have an OGO room in slot B, some groups will have an OGO room in slot D. In the hours where we could not reserve an OGO room for you, we expect you to be able to find a place to work with your group yourselves. Let your tutor know where you will be!

In general, the advise is to keep online meetings to a minimum, as work on the physical robot will require presence not only of those working on it, but also of those interfacing with it (i.e. the software developers). Also, SCRUM retrospectives and group atmosphere are much harder to

manage in online meetings. As a practical coordinator, I will not actually check the way in which you meet, but in case cooperation in your group doesn't work out, reports from your tutor about meeting behavior does give me an indication of the way in which your group took responsibility for the process that was followed.

During this first meeting, figure out who will be chairman quickly, and who will take the minutes of the meeting. I suggest the following agenda:

- Round of introductions
- Round of expectations
  What does every individual expect to learn?
- Reading of this Project Guide
  Summarize the Project Guide together.
- Round of ideas
  Does anyone have particular ideas for the robot already?
- Study of the Scoring Rubric in this Project Guide, and discussing where your group would like to put their focus. You don't have to decide today yet, but you need to know how the others feel about it.
- Make an appointment with your tutor to get access to the lockers containing the Fisher Technics kit.
- Discuss experience with SCRUM and who will/has attend(ed) the SCUM-training (see furtheron)
- The week is very short due to holidays, still you have a deadline for handing in a general planning this week, and a deadline for pitching your group ideas next week. Discuss and plan, to ensure you will make these deadlines.
- Summarize action points and post minutes on canvas

4.3 Mon Apr 24 - Deadline: enroll for SCRUM master training and presentation workshops

By the end of the meeting, make sure your group has chosen a SCRUM master, make sure this SCRUM master has registered in CANVAS for one of the upcoming trainings, and make sure that every group member has registered for one of the presentation workshops. Moreover - coordinate the registration for the presentation workshops in such a way that no two group members visit the same workshop slot (see further-on).

4.4 Wed Apr 26 - SCRUM (master training, and other deliverables)

One member of each group needs to volunteer to follow a SCUM-master training at the beginning of the DBL. The days on which this training is given are mentioned on CANVAS. The main training is on the date given in the title. Furthermore, there will be several deadlines and deliverables with respect to learning the SCRUM approach. However, these are organised centrally and are the same for all DBL's. Therefore, those deadlines and deliverables are not part of this guide.

Note that *all* group members are expected to have already followed the online course 2SCR01 on SCRUM, and have read the book "Scrum and XP from the trenches" by Henrik Kniberg.

4.5 Mon May 1 - Deadline : SCRUM planning

On this day, before midnight, you need to hand in the SCRUM planning of your group. Where, when (how often), and how are you going to meet? What is the sprint structure that you intend on using: weekly, bi-weekly, something else, when and where do you schedule your planning meetings, stand-ups, demos, and retrospectives?

4.6 Wed May 3 - Deadline : Robot requirements

As mentioned in the requirements section of this document, you are encouraged to be creative and think of an alternative robot requirement that satisfies the constraints given earlier. You are also encouraged to formulate your groups own learning goals and modify the blue print of the rubric in the end of this guide. To ensure a swift start of the DBL, you only get one chance to propose this requirement to the practical coordinator, and no guarantees are given beforehand which proposals are granted.

Before this day is over you are expected to upload a 1 minute long video on Canvas (absolute maximum is 1 min 30 sec!) in which you present your idea for the robot, the kind of challenge you want to take up in this DBL, and formulate as precisely as possible what you think is a sign of a quality result and a sign of a quality process. The practical coordinator will review those videos and grant you a go/no go, and give feedback on your robot idea - hopefully before the next meeting (but there are a lot of videos to review, so maybe one meeting later). Your tutors will at a later point, whenever it seems appropriate, discuss your formulation of signs of quality in more detail, and reflect on the final rubric together with you.

4.7 Wed May 3 - Deadline : First SCRUM backlog

At the same time as the deadline for the Robot requirements, you should upload the first backlog you've created as a SCRUM team. If all went well, you should be following a SCRUM process by now and you should have created a backlog that reflects the things that you want to achieve as a group. Of course, that backlog will change over time, whether your Robot requirements get granted or not. They change in every project. But upload your first version today.

After this deadline, there will be - undated - deliverables for every sprint in CANVAS. The idea is that you upload a small report on every sprint *before* your retrospective. That report should *at least* contain a copy of your backlog, a copy of the minutes of any meetings you had, a copy of your timesheets, and a snapshot of your SCRUM board. Anything else you find relevant may be uploaded as well, but it is - for example - not necessary to upload all your living documents.

4.8 Open Shop and Factory Floor Inspection

At several timeslots (to be announced on CANVAS) one person from your group may visit the open shop (location t.b.a. on CANVAS) in order to negotiate additional Fischer Technics components you might need for your kit, or in order to measure the Factory Floor and ensure your design matches it.

4.9 Wed May 17th : Intermediate upload of living documents

As a project coordinator, I would like to have a little bit of insight in the progress of the different groups. Therefore, I want all groups to submit a copy of their living documents before we go into the "interim presentations" week.

4.10 Mon May 22 till Fri May 26 : Interim Presentations

The interim presentations are an opportunity for you to achieve two things: firstly, each group member will give his/her own presentation and will receive feedback and do exercises to improve his/her presentation skills; secondly, we will mix groups during the presentation sessions, so the presentations actually also serve the purpose of updating the other groups on your progress.

The idea of these mixed presentations is that we would like the different groups to update one another on problems they have run into, and show the nifty ideas they have on how to solve those problems. **On CANVAS, more precise instructions can be found as to what to prepare exactly. Read them early and carefully - there is a deadline to upload some things *before* the start of this week!**

It is okay to prepare the presentation as a group, or individually, depending on preference. However, everyone will be giving a/the presentation in front of a live audience! There will be a number of parallel workshops, and in each workshop there should - ideally - be only members of different groups.

In the first week of the DBL you already need to register slots for the midterm presentations. Decide among yourselves who is going to which workshop, and who is going to present which contents where. If it turns out more people from the same group end up in the same workshop (try to avoid that) they should different aspects of the machine. Nevertheless, even if that happens you should make sure that each presentation tells a stand-alone story.

4.11 Wed June 14 - Deadline : Final hand-in of all living documents

This is the final week in which you can still work on your robot. On Wednesday, you need to hand in all your living documents, so that the tutors can start pre-grading your work and form and advice for the grading teachers. You can use the remainder of the week preparing your final demo and your poster.

4.12 Mon June 19 - Deadline : Hand-in of poster, and video evidence

Now everything is handed in and frozen. If you have video evidence of a working robot that you would like to hand-in just in case the demo goes wrong, this is the time to do so. The only thing left after tonight is to present your robot during the final demo sessions.

4.13 Wed June 21 and Fri June 23 : Final demos - demo market

During the finals of this DBL, your group will present a poster and, together with other groups, demo your functioning robot. Make sure to prepare your demo and your poster in such a way that they provide evidence of the points you mentioned in the final rubric. Also, be ready to present your decision list and minutes, as extra evidence.

The final demos will be organized as a market. A small jury of student assistants and grading teachers will walk around to inspect your work, possibly intervening with the robot to test it really thoroughly and see how it reacts to "disturbances". Convince them that your group really made a well-engineered product. You explain the requirements of your robot, the process of designing it, the most important choices you made during your design, and you provide evidence that your design indeed meets its specification.

If your group entered the communication protocol challenge, we will organize the market in such a way that you can put everything together and demonstrate that the robots of several groups can also work as a combined factory.

4.14 Wed June 28 : Returning practical material (17:00)

After everything is over, sadly, you will have to demolish the robot and return the parts. Make sure that today all practical materials are sorted and returned to their lockers, and the keys of the lockers returned to the coordinator. Failing to do so will result in a negative score for your groups total, so choose wisely whom you make responsible for this task, and try to have it executed well before this deadline.

5 Final scoring guidelines...

At the final demo and poster presentation, each group will demonstrate its robot to two grading teachers and two tutors. This jury will use the Rubric in table 1 as their main guideline for determining a final grade for the group. Notice that the word 'evidence' occurs a lot in table 1. This evidence will have to come from your demo, poster, decision lists and minutes.

Apart from the basic learning goals, you as a group determine yourself what you want to focus on during this course, and what you want to bring up as evidence. So think carefully if you really consider what you present to be quality evidence! Also, make sure you present it clearly as such, otherwise the judges may miss that you intended a certain fact as evidence. So make sure you structure your demo and poster presentation, and any logs, lists and design documents you bring to the table. Also, make sure that you make a good impression on those judges!

Individual scores are in principle equal to group scores, but in exceptional cases the practical coordinator may decide to deviate from this rule, and has done so in the past. This for example happens in case peer-reviews, retrospectives, the opinion of student-assistants, or other factors indicate that one of the group members has contributed significantly more or less than the others.

Functional behavior	The robot - in isolation - does not perform its intended basic functionality according to specification during the demo (- 20 points)	Evidence has been given that the robot - in isolation - performed its intended basic functionality according to specification at some time (+ 20 points)	The robot - in isolation - performs its intended basic functionality according to its specification during demo/testing (+ 20 points)	The robot is sturdy, and mechanically and electronically well-built (+ 10 points)	The robot reports on its internal state in - for debugging purposes sufficient - detail (+ 10 points)	*Fill in your own idea here, for example: in-depth performance measurements.* (X points)	*Fill in your own idea here, for example: verification of deadlock-free communication.* (X points)
Fault detection	The robot detects faults that did not actually occur - false positives (- 15 points)	Evidence is provided that the robot was able to detect and/or identify at least three distinct types of faults at some time (+ 10 points)	The robot *detects* three or more types of faults as faults during demo/testing (+ 15 points)	The robot *identifies* three or more distinct types of faults during demo/testing. (+ 10 points)	The group is able to *predict* correctly how the robot will react to three 'surprise faults' invented and introduced by the jury during testing (and this reaction is non-trivial). (+ 15 points)	*Fill in your own idea here, for example: the robot can recover from detected faults by guiding the user.* (X points)	*Fill in your own idea here....* (X points)
Engineering process	System spec The robot has been well specified. I.e. a clear set of use-cases, usage-constraints, and safety-properties is given that respectively describe intended behavior, desired operating conditions, and behavior that is to be avoided by the robot at all times. (+ 20 points)	Component and unit spec There is a clear decomposition of functionality of the robot into components/subsystems, and of components into units, and the function of these separate components and units has been well specified. (+ 20 points)	Implementation Good programming practice is shown. There is evidence of a good code-and-model reviewing process. Code has been properly commented and documented. (+ 15 points)	Implementation Code is based on a model-based design workflow. There is a clear higher-level model indicating how the group thinks about the software, and a clear link between that model and the software itself. (+ 15 points)	System and component test Testing scenarios are available and have been logged that cover each of the points in "Functional behavior" and "Fault detection". (+ 20 points)	*Fill in your own idea here, for example, Test Coverage: scenarios are available and have been logged that cover all parts of the code. Coverage means, that for each line of code there is a test that 'touches' it.* (X points)	*Fill in your own idea here, for example: A simulation model has been used for "simulation in the loop" testing of the control software. Score may vary depending on the detail of the model.* (X to Y points)
Reporting	Proper logs and records of decisions have been made of all the activities within the DBL. The logs clearly reflect which activities were carried out and when, giving enough details for others to reproduce any experiment or other activity. The records of decisions clearly state which decision was taken and when, which problem it is intended to solve, how it will solve the problem, which other alternatives were considered and why they were discarded, what the risks are of the chosen decision, and who is responsible for carrying it out. (+ 20 points)	Logs, records of decisions, and final poster all clearly explain the system on two levels of abstraction. A high - architectural - level, at which the overall workings are explained in a way that can be grasped quickly, and a low - implementation - level at which the details are given with a reference to the high-level model explaining why the details are such. Clear figures are used for the explanation of the high level. The logs and records of decisions contain clear references to those high-level pictures, thus linking low level descriptions to the higher level architecture. (+ 20 points)	The final poster has a clear structure. It covers at least briefly the user-requirements (system level), robot specification and software specification (component level), robot design and software design (unit level), and robot implementation and software implementation. References are given to the logs and records of decisions, pointing out details of design decisions, details of tests, etc. (+ 20 points)	The general appearance of the poster is pleasing. Tables and graphs have been properly laid out. Figures are readable and free of distractions that do not contribute to the goal of the report. The logs, records of decisions, and poster have been written in correct British or American English (+ 20 points)	The group has given a convincing final demo and pitch, well rehearsed, structured, complete, well-tuned to the audience, enjoyable, yet concise. (+ 20 points)	*Fill in your own idea here, for example: the group has given an outstanding mid-term presentation, both in content as in presentation.* (X points)	... (X points)
Various	A deadline or other group-responsibility was missed (- 5 points per deadline)	Practical material and locker-keys were not returned in time (- 40 points)	Exceptionally bad functioning of the group as a whole on multiple occasions (- 40 points)	Room for the student-assistants to express their feelings about this group (+/- 20 points)	Room for the practical coordinator / grading teacher to express his/her feelings about this group (+/- 20 points)	Room for rewarding points for various other outstanding achievements (+ 30 points)	... (X points)

Table 1: ***Guideline*** for the final scoring by the practical coordinator, tutors, and jury. The blue and green areas are considered essential and cannot be adapted! To calculate the final grade for the group, first check if the blue score is higher than 120, and check that the green score is higher than 60. If not, the group should not pass the course. If the blue and green scores are sufficient, the final grade is determined by weighing: 5 times the percentage scored in blue boxes, 3 times the percentage scored in green boxes, and 3 times the percentage scored in the white boxes. The scores in red and yellow boxes may be subtracted or added to blue, red, or white at the discretion of the jury. *Initial scores given during the demo market may be adjusted during the plenary discussion among the teachers and tutors afterwards.*

6 Other resources

6.1 Project rooms

During the whole quartile the project DBL 2IO75 is scheduled in slot B and D. For each group a room has been reserved in either MetaForum or Atlas, to be used *exclusively* by that group during the scheduled hours. Some groups have a room reserved in slot B, other groups have a room reserved in slot D. Outside these hours, the groups have to find their own space to work.

6.2 Storage Cabinets

To safely store the construction kits, the microcontroller, and the robot, lockable cabinets will be available in Metaforum and Atlas. There is no deposit for the keys of the lockers, but loss of the keys comes at a price of 50,- Euros. Worse... for some of the lockers there are no spare keys, so loosing it means you may loose access to your robot for a while. Failing to hand in the material and the keys in time at the end of the project will cost the group points (see the previous section - but enough with the threats).

6.3 Hardware

You will obtain a construction kit with FischerTechnik (and some additional) components, and a RaspberryPi microcontroller + power adapter and USB cable.

Information on how to use the FisherTechnik kit and how to electronically connect Fisher Technic components to the RaspberryPi can be found anywhere online. Make sure to study how to connect things properly, however, in order to prevent electronically damaging the RaspberryPi! If you break the RasberryPi, you unfortunately have to replace it yourself.

Some electronic components, especially wiring and a small power source to drive the motors, you are expected to buy yourself. The costs of this should be very limited. Of course, you can also use materials you have lying around. In the past, some groups even decided to 3D-print some of their own components.

6.4 Student assistants

The student assistants (tutors) are the direct coaches of the groups. The main tasks of the tutor are:

- to stimulate and motivate the groups;
- to supervise and coach the learning and working process (SCRUM);
- to provide feedback on logs, minutes, decision lists, and posters;
- to advice the practical coordinator when grading;

Sometimes student assistants may also give technical advice, but they are certainly not required to do so.

In case of technical problems that cannot be solved by an internet search or the advice of the tutor, post a question on CANVAS asking for help or email the practical coordinator.

6.5 DBL coordinator

The DBL coordinator (also called project coordinator or practical coordinator) is responsible for the project as a whole. Together with the student assistants, a technical assistant, a SCRUM coordinator, and a number of grading teachers, the DBL coordinator evaluates the results of the groups and eventually he determines the grades given to the members of the groups. The project coordinator for DBL 2IO75 is:

Name	**Telephone**	**Room**	**Mailbox**	**E-mail**
P.J.L. Cuijpers (Pieter)	(247)5917	MF 6.119	MF 6.113	p.j.l.cuijpers@tue.nl

If problems arise that cannot be solved with the help of the student assistant, or in case of a conflict with the student assistants, you may email the DBL coordinator to make an appointment, or post a message on Teams.