



Using open source paradigms to teach system development

Dimitrios Platis
@PlatisSolutions
dimitris@platis.solutions

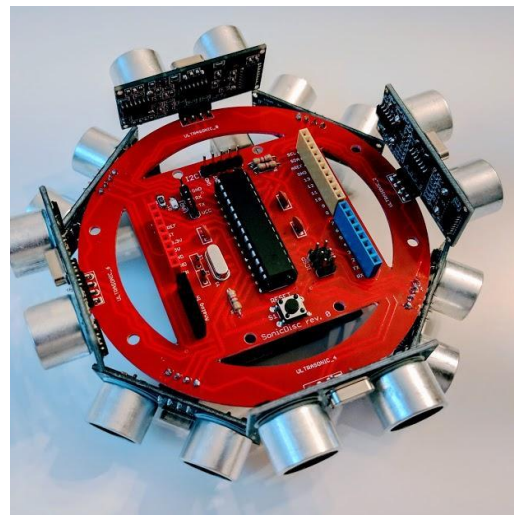
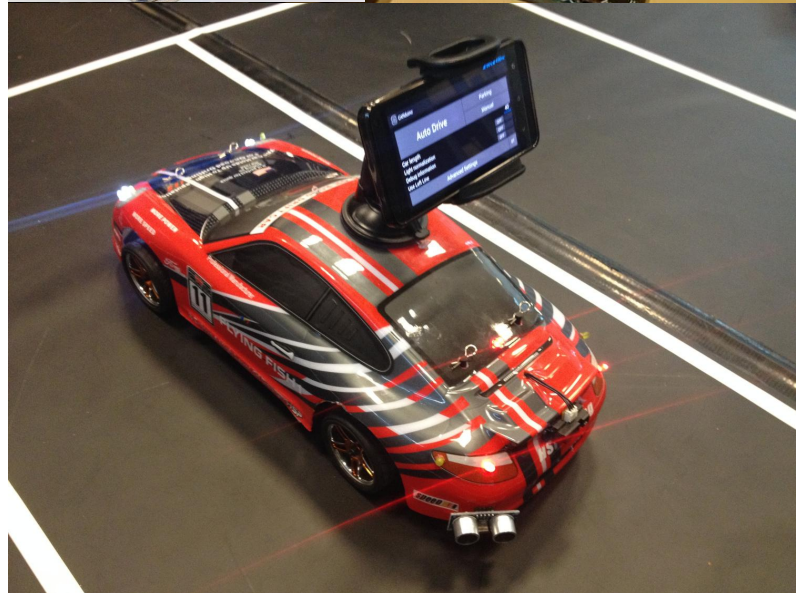
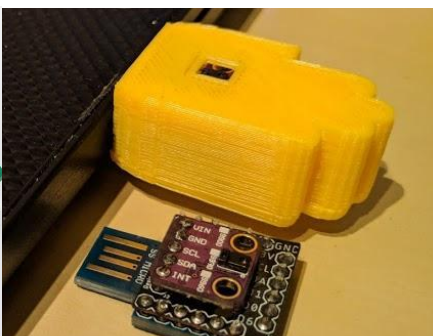
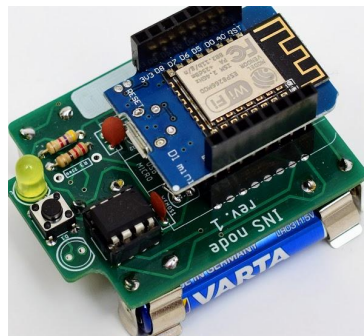


About me



Dimitrios Platis

- Grew up in Rodos, Greece
- Software Engineer @ Edument, Gothenburg
- Course responsible @ Gothenburg University
- Interests:
 - Embedded systems
 - Software Architecture
 - API Design
 - Open source software & hardware
 - Robots, Portable gadgets, IoT
 - 3D printing
 - Autonomous Driving
- Website: <https://platis.solutions>





UTBILDNING



PROJEKT



TEKNIKHUBBEN





Background

DIT112

DIT112

- Software Engineering & Management BSc
- Compulsory course
- 2nd term
- 7.5 credits
- ex-DIT524 (15 credits)

- ~70 students
- ~12 groups
- Some experience in JAVA
- Have heard of SCRUM
- A bit of experience in git
- A lot of imagination

DIT112 learning outcomes

- Define software in a system context
- Describe system requirements, system and software design, and relations between the requirements and software design
- Organize software development teams and conduct software development projects, using modern software engineering methodologies such as agile development
- Elicit, analyze, and document requirements in the form of a requirements specification
- Design software and document outcome of design work
- Implement software according to a documented software design
- Reflect on integration between software and non-software components
- Evaluate traceability between requirements, design, and implementation artefacts

When software development becomes engineering


It is not about hacking something together that "works", but establishing a development process that is:

- Repeatable
- Defined
- Controlled



Smartcar

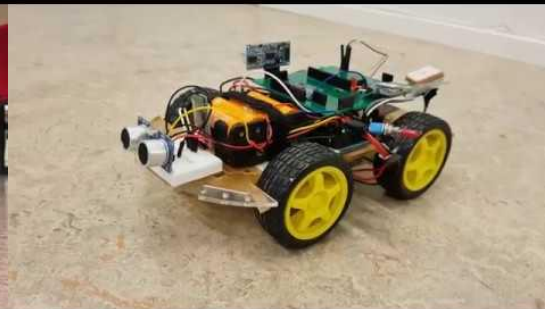
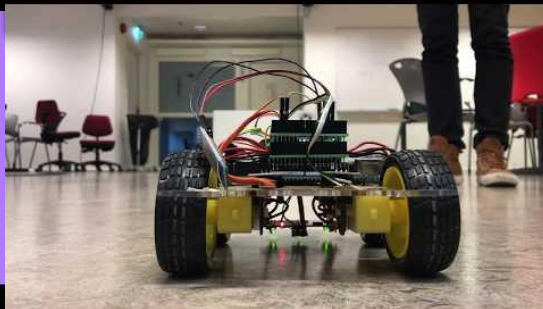
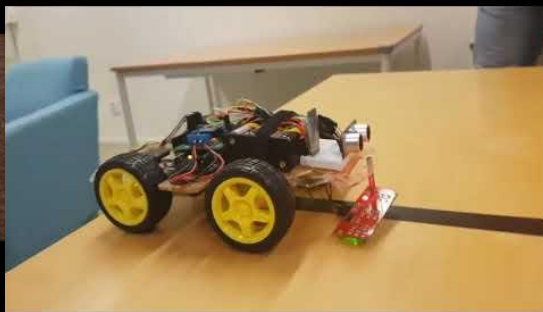
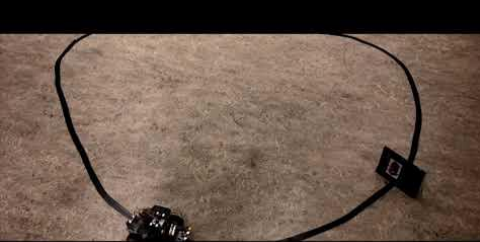
A versatile and easy to use vehicle platform
for hobby-grade projects





Smartcar

- Easy-to-use software library
 - Hardware agnostic
 - Support for multiple sensors
- ESP32 microcontroller
 - WiFi
 - Bluetooth
 - FreeRTOS
- L3G4200D gyroscope
- Directional speed encoders
- VL53L0X "micro-LIDAR"
- 5V tolerant I/O pins
- 8 AA batteries
- Open source software & hardware



DIT112-V19
DIT112-V20

Challenges

Immature system development process

Sound familiar?

- Scope creep
- Lack of communication
 - Features
 - Defects
 - Vision
- Intermittent quality
 - Frequent regressions
- Lack of domain knowledge
- Untracked work
 - Important for grading
- Unintegrated features



Improving maturity

Inspired by FOSS development



Working agile

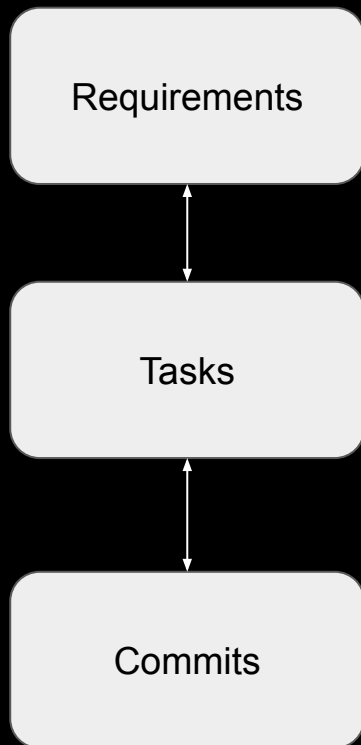
Agile in DIT112

- Product owner
 - Also customer at times
- Small & valuable increments
- Weekly sprints
 - Demos
 - Planning
- User stories
 - Persona
 - Acceptance criteria
- PO accepts only what is integrated (i.e. on *master*)

Requirements traceability



2-way traceability



Software project terminology

↔ Requirements (or Epics)
↔ Tasks (or User stories)
↔ Commits

- Multiple user stories per epic
- One epic per user story
- Link commits to user stories

GitHub features

↔ [Milestones](#)
↔ [Issues](#)
↔ Commits

- ✓ Multiple issues per milestone
- ✓ One milestone per issue
- ✓ [Link commits and pull requests to issues](#)

Labels used for grouping sprint backlog items



Automated testing

Testing

- Verify requirements
- Avoid regressions
- Discover defects before production

NO TESTS



**MANUAL
TESTS**



**AUTOMATED
TESTS**



**AUTOMATED
TESTS IN CI**





Continuous Integration



Automated, defined & continuous

- Build
- Test
- Release
- Deploy



- ✓ Merge to master allowed only when CI passes
- ✓ Personal branches ignored
 - We don't care about your side-project

GitHub Actions



Documentation

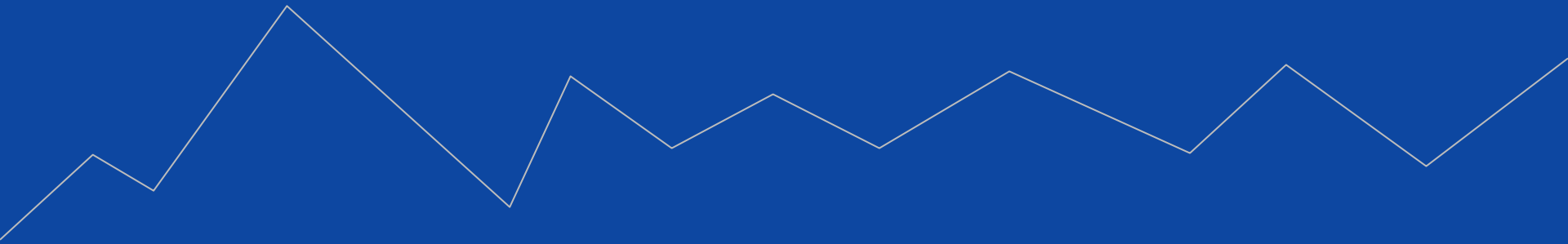


Sustainability & on-boarding

- README.md
 - What/Why/How
 - Demo video
- Wiki
 - User manual
 - Requirements specification
- GitHub pages
 - API documentation

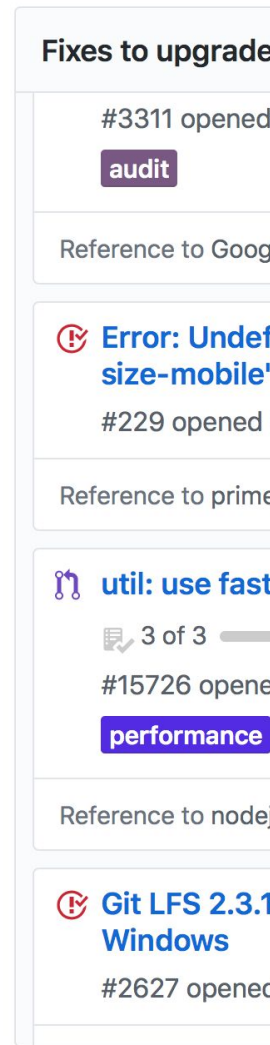
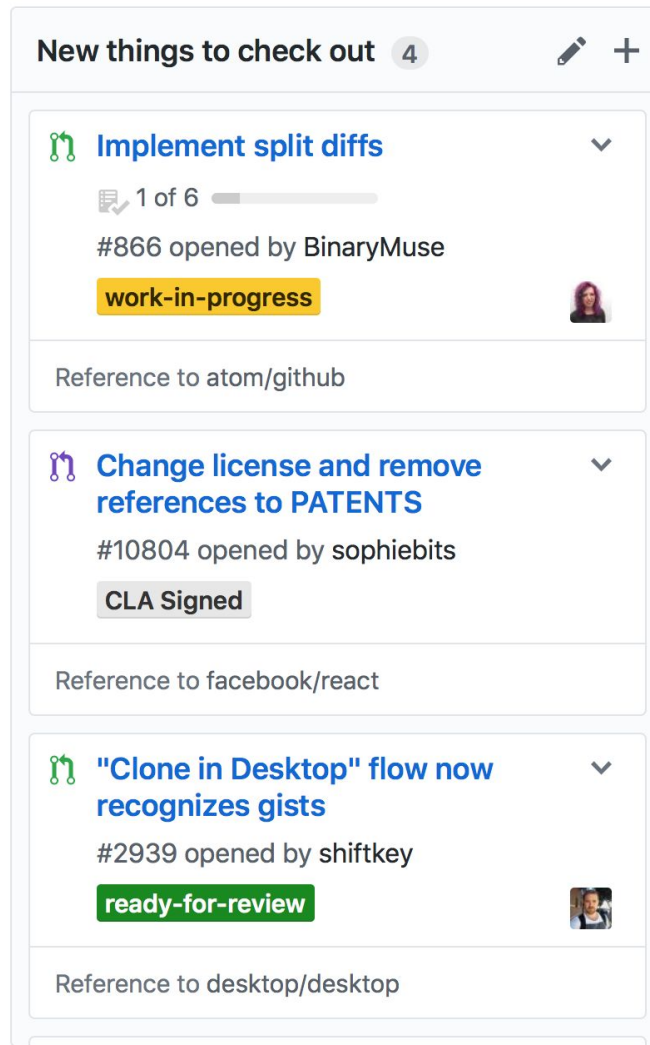
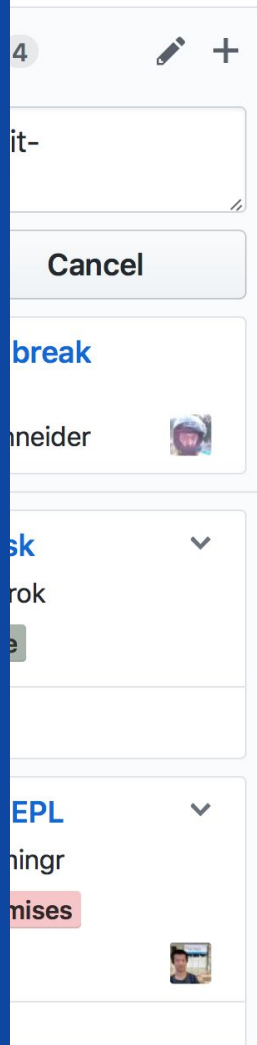


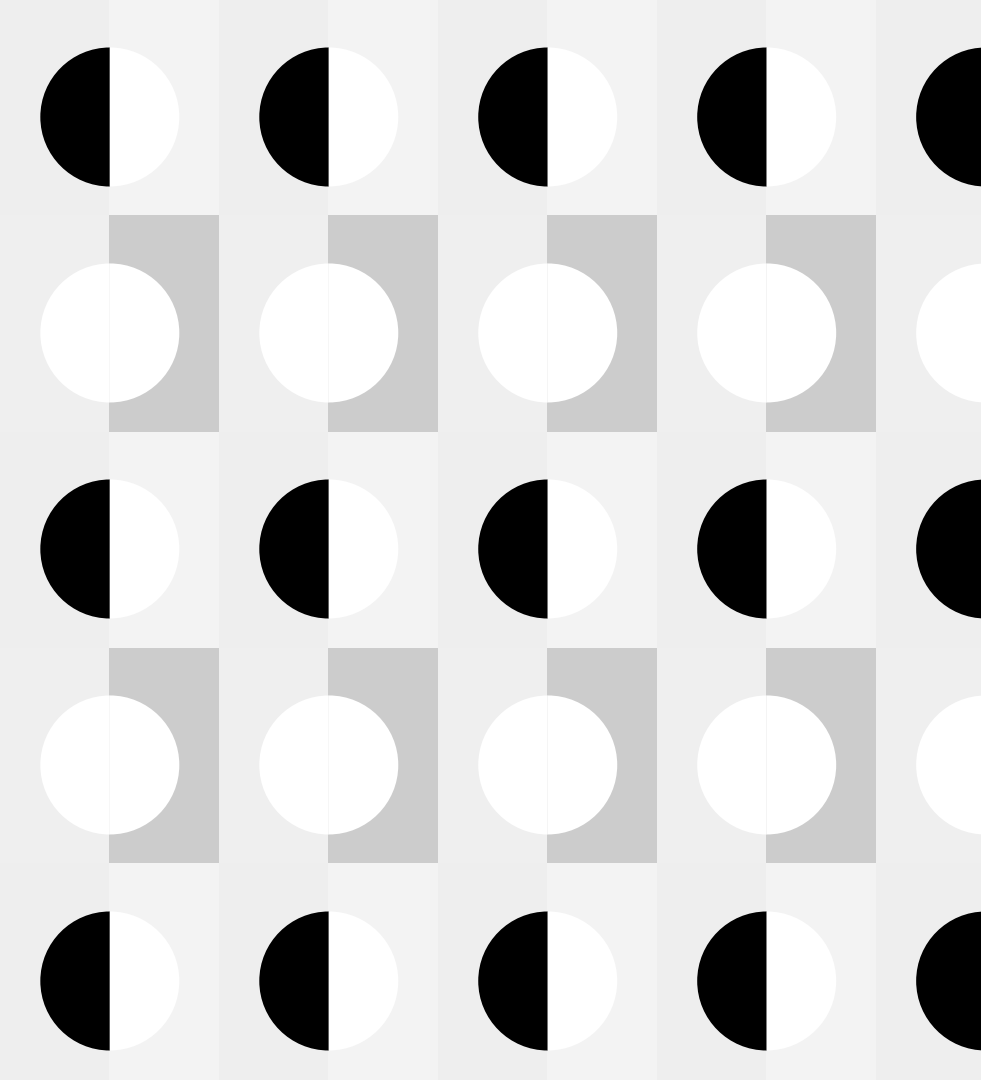
Work tracking



Communication & accountability

- Multiple developers assigned on issue
 - Pair programming
 - Developers not *penalized* for collaborating
- GitHub project
 - Issues broken down to tasks
 - Track upcoming, ongoing, finished work
 - Automatically move issues





Code reviews

When someone tries
to directly push
code to master.



Push to master?
No.

- ✓ Acceptance criteria
- ✓ Definition of Done
- ✓ Code review
- ✓ CI checks

Open development



Peeking is not cheating



-
- Public sprint demos
 - Short, less than 5 minutes
 - Slides discouraged (only 1 allowed)
 - Live demo if possible
 - Public development
 - Solutions to common problems
 - Respect licenses
 - Public discussions
 - Canvas LMS
 - Forum
 - Chat
 - Slack



Takeaways

What's your excuse?