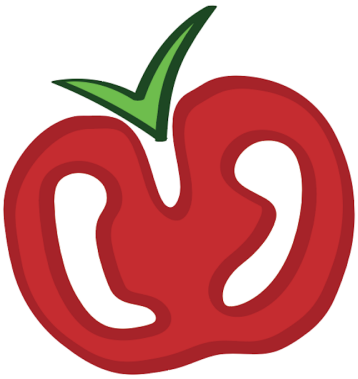


Project Plan:
Tomato – A Group Work Space Monitor Tool

T-110.5130 - Mobile Systems Programming

Julius Eerola, Anssi Grekula, Jin Jin, Joni Lampio, and Juho Marttila

Feb 21st, 2016

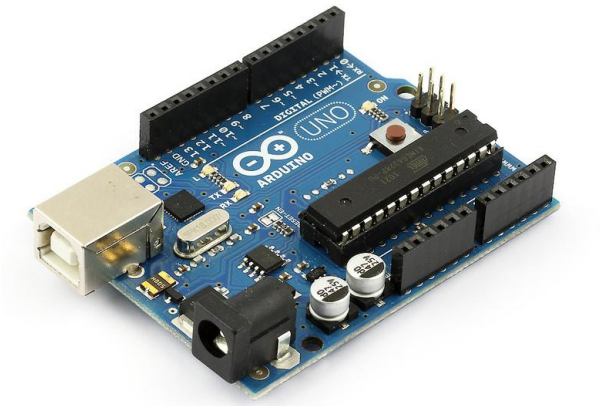


Image credits: Snootlab (https://flic.kr/p/adQqaY, CC BY 2.0) and
danishprakash (https://fi.wikipedia.org/wiki/OnePlus_One#/media/File:OnePlus_One.png, CC BY-SA 4.0)

Table of Contents

1. Introduction

Finding a free work space (for personal or group work) can be an unnecessary hassle for students and employees across organisations. The most common modern solution is an online room reservation system, but the effort of reserving a room feels often over the top, especially if the need is just for half an hour or so. Additionally, rooms are sometimes reserved "just in case" for a longer period of time than necessary, which leads to lots of wasted time when nobody is using the room although it is empty. What's important, is whether the room is actually empty or not now. And the only current way to know that is to go and see. This is something we want to make easier.

The system, named ***Tomato*** after a group work room in the library of CS building, is a group workspace monitoring tool, which uses a device with sensors to monitor primarily the occupation of a room (group work space, meeting room or similar) and secondarily the air quality in the room. That information is then delivered to people currently using or willing to use the room. In short, Tomato acts as additional pair of eyes that always keep watch on the room. Moreover, it also lets people in the room know when it's time to take a break, open the door and stay effective through the whole session.

The system will be installed in one group work room of Learning hub Greenhouse (the Library of Computer Science Building). With a system like Tomato, the members of an organisation, for example Aalto students or employees, could utilize available meeting spaces more efficiently and comfortably.

2. Stakeholders and staffing

This project has four stakeholder groups: the project team, the mentor, course personnel, and of course, the (potential) users of the system being developed during the spring.

The project team is an interesting mixture of different backgrounds and even nationalities. We have development background for Android OS as well as some previous experience in Arduino development. There is also some understanding of people and their interactions with technology and each other, as well as some design thinking and service concepting experience, without forgetting the business aspect.

2.1 The project team:

Jin Jin, *Android & Arduino Development*

- Geoinformatics, Aalto ENG
- nmbqz@me.com
- 464989

Anssi Grekula, *Design*

- Information Networks, Aalto SCI & Sociology, Uni. Helsinki
- anssi.grekula@aalto.fi
- 82045D

Julius Eerola, *Android & Arduino Development*

- Computer Science, Aalto SCI
- julius.eerola@aalto.fi
- 269854

Joni Lampio, *Android & Arduino Development*

- Computer Science, Aalto SCI
- joni.lampio@aalto.fi
- 465441

Juho Marttila, *Project manager, Android & Arduino Development*

- Industrial Engineering, Aalto SCI
- juho.marttila@aalto.fi
- 356178

The mentor for this project is M.Sc. (Tech.) Teemu Kämäräinen, a doctoral candidate at Department of Computer Science, Aalto SCI. His research focus is cloud gaming but he is also excited about the Internet of Things and the Arduino board as an accessible tool for IoT-related experimentation. He also provides the project team the required materials, such as Arduinos, sensors, and Arduino extensions ("shields"). Kämäräinen is the main responsible of giving the project brief and advising the group to succeed in fulfilling the necessary requirements. He is also partaking in evaluating and grading the team's effort. However, he is in no way involved in the actual design and development work.

Course personnel consists of the responsible teacher, teaching researcher, D.Sc. (Tech.) Sakari Luukkainen and course assistant, doctoral candidate, M.Sc. (Tech.) Olli Mäkinen, both from Aalto SCI. They evaluate and give grades to all project teams, including ours.

The intended users of the system are firstly Aalto tech students who study in CS building regularly and secondly Aalto campus management staff. The students do group works or course projects in different spaces around the campus. Campus management staff are planning for effective use of the campus spaces. We need to observe the users' behaviour, listen their needs, and show them our early prototypes in order to deliver a service that fits them. As we belong to the target user group, we are able to test the system ourselves – to an extent. Naturally there's a bias that needs to be acknowledged and addressed.

3. Goals and scope

3.1 Goals

This project aims at making it easier for students to decide where to go to do group or personal work by showing current information of (at first) one group work room. Ideally the service would list various rooms and their availability, but for proof of concept and to enable fast iterations, we cover only one room.

The business goal for Aalto campus management staff would be to get better data from the usage of the rooms and thus be able to make more informed decisions about how to use and also maintain the rooms.

For users of the room(s) the service saves extra effort of walking all the way to the library just to see that the room is taken.

3.2 Individual learning goals

Jin Jin

- Learning Android customized UI
- Google APP Engine API
- Arduino sensor data integration with cloud
- Backend API development

Anssi Grekula

- Learning the gitflow
- Learning how to make an Android app
- Understanding Material Design principles by designing an app layout
- Understanding design principles for connected products (IoT)
- Learning to connect Arduino to Cloud

Julius Eerola

- Learning Android development
- Learning Arduino development
- Learn to work with Google App Engine platform

Joni Lampio

- Learn more on mobile application design
- Learning Android development
- Learning to work with Google App Engine platform

Juho Marttila

- Learning Android development and API
- Learning how to utilize Arduinos and similar devices

3.3 In scope

To maximize learning, the finished system will cover only one room, but do that extensively. We will investigate different ways to interpret data from multiple sensors and to trigger meaningful, non-obtrusive notifications.

3.4 Out of scope

Scaling up to more rooms or to an ecosystem level is left out of the scope of this project. Also integration to the room reservation system and enabling messaging between users (e.g. the one who has reserved a room and someone else who finds the room empty and wants to use it) are not implemented. In the end a further development backlog is created to make it easy to build continuation on top of the initial idea.

Process-wise, proper usability study with test users outside of the development team is left out of scope.

4. Work practices and tools

4.1 Time tracking

A team member will write down to a merge request how much time he needed for implementing a corresponding feature. Team members decide together which feature each member is going to do in the weekly meeting. By default, each feature should take a week to implement but larger features may require more time. Additionally, team members log their hours in a spreadsheet for a more detailed reporting.

4.2 Documenting

Technical documentation is written in markdown language and is located in the Git repository. Documents to the course personnel and other stakeholders will be available in PDF file format. The team will use Google Drive to manage non-technical documentation.

Each team member will participate in documentation. The team will review documents in a weekly meeting.

4.3 Communication

Slack is the main communication channel for team members. In addition, the team will use email for communicating with other stakeholders when necessary. Finally, merge requests in Git work as a channel for technical discussion.

Weekly, the team will arrange a meeting to discuss about the project progress, problems and new features.

4.4 Version control

Git is the version control system of the team. The repository is hosted at Niksula, more precisely https://git.niksula.hut.fi/grekula1/iot-aalto.git. New features are represented as issues so the version control acts also as a project management tool. Each issue will have a label indicating its importance and a short description.

The team will use the *Feature Branch Workflow*[1]. This workflow is described in the following table:

#	Steps	Links and commands	Responsible
1.	Create an issue for a new task.	https://git.niksula.hut.fi/grekula1/iot-aalto/issues	team
2.	Assign the issue to a team member in a weekly meeting.		team
3.	Create a new branch from the master branch.	`git checkout master` `git checkout -b my-task`	assignee

[1] https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

4.	Implement the task and push it to the remote repository.	`git add <file>`
		`git commit -m 'Hello world'`
		`git push -u origin our-feature/my-task`
			assignee

5.	Make a merge request and assign a team member to review the implementation.	https://git.niksula.hut.fi/grekula1/iot-aalto/merge_requests	assignee

6.	Check and verify that quality goals are satisfied. After that, merge the task to the master branch.	`git checkout our-feature/my-task`
		…
		`git checkout master`
		`git pull`
		`git pull origin our-feature/my-task`
		`git push origin master`
			reviewer

7.	Have a discussion about implemented tasks in the weekly meeting.		team

4.5 Design

Architectural design is done in a meeting where the system architecture is visualized on a whiteboard and group members discuss possibilities, aiming at optimal solution, taken the previous experience of group members and the technological course project requirements.

User interface and user experience design follows Google's Material design guidelines and best practices as described in Rowland, Claire, et al's *Designing Connected Products: UX for the Consumer Internet of Things* (O'Reilly Media, Inc. 2015).

The design is tested by dogfooding[2], which means testing by using the system ourselves. Since the course focus is on building a technologically functional system, proper usability testing is scoped out of the project.

4.6 Quality assurance

Testing

A team member implementing a task will also create corresponding unit tests. The reviewer who gets the merge request will first run these unit tests so that he is able to verify that the code works before inspecting code quality.

Unit testing depends on the platform used in the task. First, the backend, i.e. Google App Engine code is tested with the Python unit testing framework PyUnit. Second, Android unit tests are written in Java using JUnit testing framework. Finally, Arduino is tested with example data.

When a feature, i.e. a user story requirement is completed, the team will test it in a real environment and see if the system is working correctly.

[2] https://en.wikipedia.org/wiki/Eating_your_own_dog_food

Software development process

The team will use a systematic software development process, Scrum. During each sprint, that takes two weeks on average, a feature (=requirement) will be finished. The feature is split into tasks, i.e. subfeatures that are put to the sprint backlog. In the weekly meetings each team member gets one or several tasks to finish during the sprint, depending on how much effort each task requires. When all tasks are finished, the feature is released and a new feature is selected.

4.7 Tools

This table summarizes all the tools used to create the system.

Purpose	**Tool**
Internal communication	Slack
Stakeholder communication	Email
Document management	Google Drive
Version control & Project management	Niksula GitLab
Arduino development	Arduino IDE
Android development	Android Studio
Cloud backend	Google App Engine (Django)
Web frontend	JavaScript HTML, CSS

5. Schedule and resources

5.1. Initial schedule

Initial schedule and work allocation for the project is outlined on the following table. Naturally the exact content and scope of each sprint is decided in the beginning of the sprint, but as for now this serves as an initial plan as well as a tool to allocate project hours:

Work element	**Responsible**	**hours/ person**	**hours (total)**	**DL**
Group forming and initial scoping	team	10	30	Feb 21
Concept and technical planning	team	20	60	Feb 21
Arduino and sensor installation into the room	team	2	6	
Project Plan	team	10	30	

Final Report	team	15	45	
Sprint 1: Requirement User Story 1	*team*		*83.4*	*Mar 2*
Sprint planning	team	2	6	
Hardware installation (Arduino + sensors)	Joni	2.4	2.4	
Arduino software development	Joni, Julius	10	20	
Android software development	Juho, Jin	10	20	
Android UI design	Anssi	15	15	
Cloud backend software development	Julius, Joni	10	20	
Sprint 2: Requirement User Story 2 & 4	*team*		*83.4*	*Mar 16*
Sprint planning	team	2	6	
To be decided in sprint planning meeting			77.4	
Sprint 3: Requirement User Story 3	*team*		*83.4*	*Mar 30*
Sprint planning	team	2	6	
To be decided in sprint planning meeting			77.4	
Sprint 4: Requirement User Story 5	*team*		*83.4*	*Apr 13*
Sprint planning	team	2	6	
To be decided in sprint planning meeting			77.4	
Sprint 5: Requirement User Story 6	*team*		*83.4*	*Apr 27*
Sprint planning	team	2	6	
To be decided in sprint planning meeting			77.4	
Final User Acceptance Testing	team	4	12	Apr 27
TOTAL HOURS			***600***	

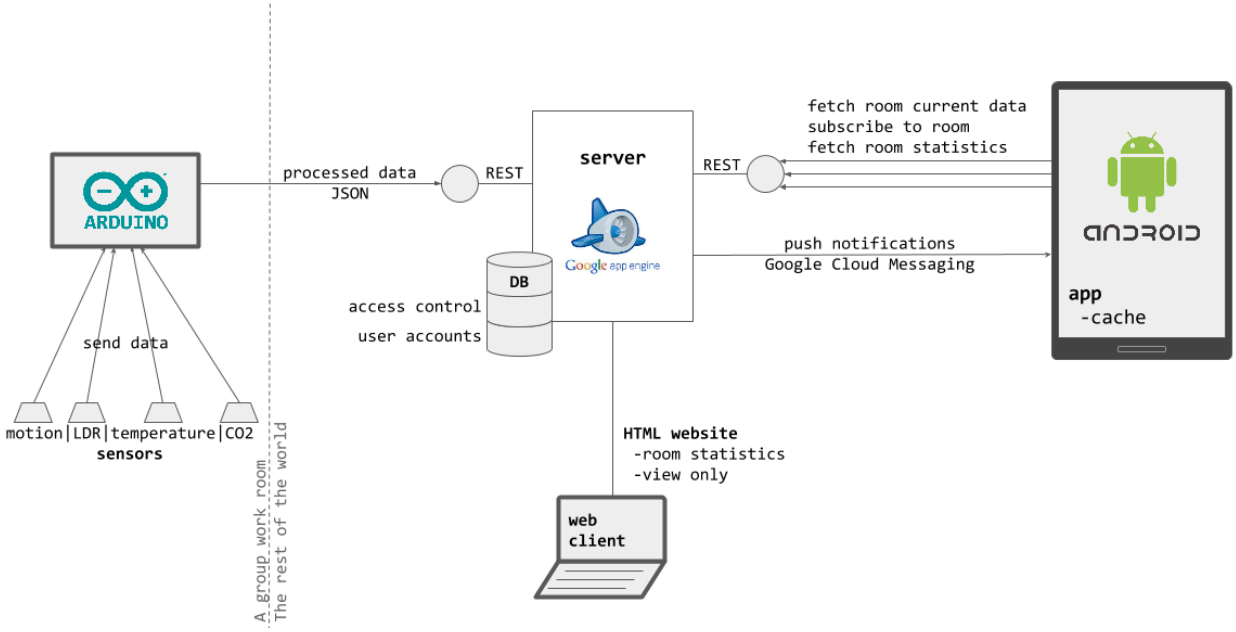
5.2 Resources

Resources needed to develop the system are listed as follows:

- Arduino boards *(provided)*
- Sensors *(provided)*
  - Motion sensor
  - Temperature sensor
  - CO2 sensor
  - LDR (Light dependent resistor)
- Android device *(personal)*
- Laptop computers *(personal)*

6. System overview

The high-level overview of the system is presented on the following diagram:



An Arduino device equipped with sensors is installed in the group work room. The sensory data about the occupation status and air quality is initially processed and filtered in the Arduino before they are sent to the cloud server via LAN or Wi-Fi. The server stores the sensor data in a database. The server interprets this data in order to tell users whether there is someone in the room or not, and to set the thresholds of air quality becoming “too bad”. The server also displays the data as a static HTML website which authorized users can access to view the historical statistics. The server also provides a RESTful API which allows an Android application to access the current status of the room (free/occupied). The Android user can also browse statistics and subscribe to the room to receive a push notification from the server when it appears to be free, or, if currently working in the room, get notified if the air quality gets too low and it is time to open the door and have a break.

Use cases for the system are described in the following requirements chapter as user stories.

7. Requirements

The functional and non-functional requirements are listed below as user stories ("As an X I want to Y in order to Z.")

MVP:

1. As a student I want to know whether the room is empty or not at the moment in order to decide whether to go there.

Preliminary scope:

2. As an Aalto campus manager I want to see the room usage statistics in order to optimize the usage of the room.
3. As a student working in the room I want to get notified when the air quality is too bad for effective working in order to take a break and stay effective.
4. As an Aalto campus manager I want to see the room air quality statistics in order to plan for improvements.
5. As a student I want to subscribe to an occupied room in order to receive a notification when it becomes free.
6. As a subscribed student I want to receive a notification when the room becomes free in order to take it before others.

Out of preliminary scope:

7. As a student I want to signal my willingness to use a room to the people currently using the room in order to increase my chances to get the room faster [communication between users to enable flexible negotiations].
8. As a student I want to reserve a room starting from the present moment in order to secure my access to the room remotely.
9. As a student I want to see whether an empty room is reserved or not in order to prepare for quick evacuation when the one who originally had a reservation appears [integration to reservation system].
10. As a student who has reserved a room I want to be reminded of my reservation in order to flexibly cancel it if my plans change.
11. As an Aalto campus manager I want the reservation situation accurately correspond the usage situation in order to ensure effective use of the space.
12. As a student I want to find an empty room nearby in order to do group work.
13. As an Aalto campus manager I want to see usage statistics of as many rooms as possible in order to optimize the space usage in the building and wider campus area.

8. Risk management

The project involves many risks which we have to take into account. We have to be prepared for them to come true, and this is why we have listed the relevant risks in the table below. The severity and probability of each situation is estimated in three step scale (low, medium, high) alongside with its impact and how we are going to prepare for it.

Risk	**Impact**	**Seriousness (low, med, high)**	**Probability (low, med, high)**	**Mitigation**
One group member doesn't do his tasks or quits	Increased workload for others	medium	low	All work is checked out by someone else. Commitment to the team.
Arduino breakdown	Sensor data can't be provided.	high	medium	Get spare Arduinos. Handle Arduinos with care.
Sensor breakdown	Sensor can't be used.	medium	medium	Get spare sensors. Handle sensors with care.
Incompatible sensor	Sensor can't be used.	low	medium	Select only sensors which are known to work with Arduino.
Hardware/tools are not available	Development/testing slows down	medium	low	Order hardware as soon as possible. Ensure that selected tools don't have any availability limitations
Insufficient programming skills	Development slows down	medium	low	Search for solutions and ask help. Take part in android development workshops.
Cloud not available	Development/testing slows down	medium	low	Select reputable cloud provider. Ensure that cloud platform doesn't have usage limitations that would effect to project.
Breakdown of Android device	Testing on real device can't be performed	high	low	Ensure that we have multiple devices which we can use for testing. Use simulator if all devices fail.
We are not able to provide all features before deadline	Missing features	medium	medium	Make sure that at least minimum viable product is reached by prioritising feature development.
Computer breakdown	Code is lost	high	medium	Use Git so we don't lose everything.
Improper use of Git.	Git repository is messed up.	low	medium	Use Git properly and follow our Git workflow guidelines.