Project Plan

OpenSTS

Alltrons
Eindhoven

Michał Raczkowski

4465024

Date	11-03-2024
Version	1.0
Status	Done
Author	Michal Raczkowski

Versions

Version	Date	Author	Amendments	Status
1.0	11-03-2024	Michał Raczkowski	-	Done

Communication

Version	Date	To
1	11-03-2024	University coach and company coach

Contents

1 Project Assignment **3**

1.1 Context . . . 3
1.2 Goal of the project . . . 3
1.3 The assignment . . . 3
1.4 Scope . . . 4
1.5 Conditions . . . 4
1.6 Finished products . . . 5
1.7 Research questions . . . 6

2 Approach and Plannings **7**

2.1 Approach . . . 7
2.1.1 Test approach . . . 7
2.2 Research methods . . . 7
2.3 Time plan . . . 9

3 Project Organization **10**

3.1 Team members . . . 10
3.2 Communication . . . 10

4 Risks **11**

4.1 Risks and fall-back activities . . . 11

1 Project Assignment

1.1 Context

Alltrons is a premier research and development partner, specializing in transforming ideas into scalable products. With a focus on IoT and robotics, Alltrons leverages a diverse team of specialists across UI/UX design, software development (including frontend, backend, and embedded systems), electrical engineering, mechanics, and business strategy. This multidisciplinary approach eliminates the need for companies to engage multiple parties or expand internal teams for R&D projects. Alltrons embodies innovation, providing comprehensive solutions from concept development to production, streamlining the path from ideation to market-ready products.

1.2 Goal of the project

The goal of this project is to evolve our current IoT solution framework from being based on general-purpose, Linux-based devices (like Raspberry Pi) to a more specialized, embedded architecture (using chips such as nRF9160, or nRF52840). This evolution includes creating a desktop application dedicated to programming the new embedded hardware, enabling Over-The-Air (OTA) updates for seamless, future-proof upgrades.

The motivation behind this transition is driven by the necessity for a solution that is not only more cost-efficient but also superior in terms of energy and resource efficiency. The move away from a Linux-based framework allows us to address these needs while expanding our potential client base. Embedded systems offer a targeted approach to IoT development, aligning with the market's demand for leaner, more efficient technologies.

The desktop application will play a pivotal role in this project, offering users the ability to configure embedded devices through a USB interface for initial setups such as GSM provider preferences, server connections, and network configurations. It's crucial to note that while the application facilitates initial device programming and configuration, the OTA update mechanism operates independently of this utility.

This project's overarching aim is to refine our IoT solutions' efficiency and market applicability, positioning us to cater to a broader array of client needs with a more refined, cost-effective, and technologically advanced development framework.

1.3 The assignment

The assignment involves creating a desktop application designed for the configuration and adjustment of embedded devices within our existing infrastructure. This application will serve as a pivotal tool for deploying a wide range of IoT solutions. The primary function of this desktop application is to provide a user-friendly interface for configuring embedded hardware, tailored to the specific needs of various IoT applications.

1.4 Scope

Scope of the project:

Includes	Excludes
Design and development of the desktop application for embedded device configuration	Development of the OTA update mechanism (handled separately)
Integration with existing embedded hardware platforms	Creation of new embedded devices or hardware modifications
USB interface development for device configuration	Long-term maintenance and support beyond initial deployment
User interface design for easy navigation and operation	Development of additional applications for other platforms (mobile, web)
Testing and deployment of the application	Expansion to non-IoT embedded systems or unrelated hardware
Documentation	

1.5 Conditions

The preferred frontend framework should be **React** because it is the main frontend framework used within the company.

1.6 Finished products

A Product Breakdown Structure of the end and intermediate products that the project will deliver with a short description in text of each product. The end products are more than the project plan and the product itself. Also, for example, requirements and architecture documents and research and test reports are typical parts of a PBS. These documents are important for the relevant stakeholders during development as well as during the transfer and during the management phase. During the project you can change the PBS and you can add or remove products in consultation

1. **Desktop Application for Embedded Device Configuration**: A fully functional desktop application tailored for configuring embedded IoT device. This tool will enable users to set up device via a USB interface, covering initial setups like preferences, server connections, and network configurations.
2. **Integration Modules**: Software components designed to ensure seamless integration between the desktop application and existing embedded hardware platform.
3. **USB Interface Development**: A robust USB interface for the desktop application, facilitating direct communication and configuration of embedded device.
4. **User Interface (UI) Design**: UI for the application, designed to simplify the configuration process for users with varying levels of technical expertise.
5. **Documentation**: Documentation covering the application's design supporting future developers.

1.7 Research questions

To ensure the successful development and deployment of the desktop application for configuring and adjusting existing infrastructure embedded devices for IoT solutions, the following research questions have been identified as critical:

1. **Compatibility and Integration**:
    - What are the primary technical challenges in achieving seamless compatibility between the desktop application and a wide range of embedded devices (e.g. nrf9160, nRF52840)?
    - How can the application support diverse IoT solutions and configurations
2. **Interface Design**:
    - How can the application provide a streamlined process for configuring critical parameters (GSM provider, server, network settings) via USB interface?
    - How to design a user interface that simplifies the configuration process for users with varying levels of technical expertise?
3. **Security**:
    - What security measures are necessary to protect the configuration process and sensitive device settings from unauthorized access?
    - How can the application support diverse IoT solutions and configurations
4. **Technical Feasibility and Development Strategy**:
    - What development frameworks and technologies are most suited for building a cross-platform desktop application that meets the project's requirements?
    - How can the project leverage existing infrastructure and tools to accelerate development and reduce costs?

2 Approach and Plannings

2.1 Approach

Our project adopts an Agile methodology, concentrating on the development of a desktop application for configuring embedded IoT devices. Key steps include:

- **Agile Development**: Work is divided into two-week sprints, focusing on incremental progress and flexibility to adapt to project needs.
- **CI/CD Implementation**: We'll use Continuous Integration for automated testing to maintain code quality and Continuous Deployment for streamlined updates, aligning with our objective to deliver a user-friendly configuration tool.
- **Technology Choices**: React will be used for the frontend to comply with company standards. Backend and USB integration will prioritize compatibility with embedded platforms like nRF9160 and nRF52840.
- **Scope Considerations**: Excludes OTA update mechanism development, focusing instead on USB configuration capabilities and user interface design for ease of use.
- **Final Deliverables**:Emphasize testing, deployment, and documentation to ensure a comprehensive and reliable application.

This streamlined approach ensures the project remains focused on its core goal: creating a desktop application that simplifies embedded device configuration within our specified scope.

2.1.1 Test approach

A blend of manual and automated tests will ensure functionality and user experience, supported by code quality reviews.

2.2 Research methods

Research methods from the DOT framework that will be used in the project:

- **Library**:
  - **Benchmark creation**: Comparing different technologies and hardware to find the best fit for the project's goals and cost-effectiveness.
  - **Best, good & bad practices**: Using the best practices of software development adapted accordingly, which technology is used to create reliable and readable software
  - **Design Pattern Search**:Applying design patterns while developing software to keep it structured, which will ensure expendability and flexibility
  - **Expert Interview**: Conversations with specialists within and outside the company to gather knowledge about the usage and best fit of technology.

- **Field**:
  - **Explore user requirements**: Discussing with stakeholders about the final usage of the application, its purpose, and the plan for future development.
  - **Domain modelling**: Discussing with stakeholders about usage, target group, previous usage, and the current state of the project to fully understand the application of software in the field and its goals
  - **Problem analysis**: Discussing with stakeholders about what improvements and problems need to be solved to achieve the desired functionalities.
- **Lab**:
  - **Component test**: Testing each component of the solution: UI, backend of the desktop application, software responsible for connection with hardware, and firmware of the hardware.
  - **Hardware validation**: Validating if the choice of hardware is suitable for our solutions, and it has the ability to fulfill the needs and requirements of the project, also validating its capabilities and compatibility.
  - **Non-functional test**: Checking software under its adaptation for extensibility and readability in a way that can be developed in the future by others. Also, checking its performance.
  - **Security test**: Preventing unauthorized access and modification to software. Applying techniques to prevent unauthorized access, modification, and reading of code
  - **System test**: Check if every functionality works before deployment
  - **Unit test**: Implementation of unit testing in the codebase of software, but also testing each component separately (UI, backend, firmware).
- **Showroom**:
  - **Static program analysis**: Impression of how well your code is written and trying to find security variables and weak spots
  - **Peer review**: Feedback and assessment of software by peers
- **Workshop**:
  - **Proof of Concept**: Creating a proof of concept to demonstrate that basic functionalities are achievable using the chosen tools, hardware, and architecture.
  - **Prototyping**: Creating prototypes to evaluate functionalities and adjusting requirements accordingly based on testing the prototype.

2.3 Time plan

- **Phase 1**: Introduction, planning, setup and research (Weeks 1-3)
  - **Introduction**: Introduction, and assimilation in company.
  - **Getting familiar with project**: Understanding concept goals of the project
  - **Planning**: Creating project plan
  - **Research**: Research possible technologies and choosing the best fit for project
  - **Proof of Concept**: Creating a proof of concept to demonstrate that basic functionalities are achievable using the chosen tools, hardware, and architecture.
- **Phase 2**: Core Development (Weeks 4-10)
  - **Initial Development**: Getting familiar with tools, and hardware and development toolchian.
  - **Core Development and Iterative Testing**: Developing main functionalities and their tests.
- **Phase 3**: Feature Finalization and Testing (Weeks 11-13)
  - **Feature finalization**: Finalization and debugging of key features.
  - **Testing**: Conduct extensive testing of features usability security and communication.
- **Deployment, Review and documentation**: Prepare for deployment, project review, and closure. Finalization of documentation.
  - **Project review and assessment**: Project review and assessment by Company and by University
  - **Prepare for deployment**: Preparing for deployment (last tests, and building binaries)
  - **Grade and closure**: Official grade and closure of the project from company and university

3 Project Organization

3.1 Team members

Name + Phone + e-mail	Role/tasks	Availability
Kayle Knops kayle.knops@alltrons.com	CEO/Control	1 hour/week
Luis Pellicer Collado luis.pellicer@alltrons.com	Tech Lead Embedded Software	About 3 hours per week, depends on needs
Kamyar Khodayari kamyar.khodayari@alltrons.com	System Architect	depends on needs
Schürgers,Frank F.P. f.schurgers@fontys.nl	University coach	
Michal Raczkowski m.raczkowski@student.fontys.nl michael.raczkowski@alltrons.com	Intern	40h/week

3.2 Communication

Effective communication is pivotal to the success of this project, ensuring company supervisor, teacher supervisor and other stakeholders are aligned and informed throughout the project lifecycle. Communication strategy includes:

- **Regular meetings**: Weekly meetings will be held to review progress, discuss challenges, and plan for the upcoming weeks. These meetings will include the company supervisor, company mentor, and others invested in the project.
- **Development Updates**: Continuous Integration and Deployment (CI/CD) practices will facilitate real-time updates on development progress. This will ensure immediate feedback and transparency in development cycles.
- **Documentation and Reporting**: Documentation will be maintained and updated regularly, including technical documentation and project reports. Access to this documentation will be provided to company supervisor and teacher supervisor.
- **Feedback Mechanism**:An open feedback mechanism will be established to collect input from company supervisor and teacher supervisor testing phases. This will help in refining the product according to needs and preferences.

4 Risks

4.1 Risks and fall-back activities

1. **Technical Compatibility Risks**: Risk of incomplete compatibility with embedded devices, leading to integration challenges.
    - **Prevention**: Early compatibility testing of hardware.
    - **Fallback**: Modular architecture for easy updates addressing compatibility issues.
2. **Security Vulnerabilities**: Risk of security flaws.
    - **Prevention**: Security assessments and penetration testing in CI/CD.
    - **Fallback**: Emergency plan for quick vulnerability patching.
3. **Project Delays**: Risk of delays due to unexpected technical issues or resource constraints.
    - **Prevention**: Agile methodology with buffer periods for unforeseen tasks.
    - **Fallback**: Prioritize core functionalities and reassess project milestones.
4. **Insufficient Knowledge**: The risk that the intern may lack adequate knowledge in specific technical areas, impacting development efficiency and product quality.
    - **Prevention**: Identify knowledge gaps early and plan for targeted training sessions.
    - **Fallback**: Adjust project timelines to accommodate learning curves or reassign tasks to intern.

By proactively addressing these risks with appropriate preventive measures and fallback plans, the project is better equipped to navigate potential obstacles and move towards successful completion.