Internship Program – Software & Engineering Tracks

1. Program Overview

Our internship program provides university students with hands-on experience in real industrial product development. Interns will work directly with senior engineers, participate in agile workflows, and contribute to real features, tools, and research activities.

The program offers four specialized tracks:

- Android Software Engineering
- Solution Software Engineering
- Embedded Software Engineering
- Automation Test Engineering

Each track offers in-depth training, mentorship, and project-based learning tailored to the required technical skillsets.

2. Benefits for Students

- Real-world experience working with industrial hardware, scanners, embedded systems, Android frameworks, and enterprise software.
- Mentorship from experienced engineers.
- Hands-on project assignments enabling interns to contribute to production-level software.
- Soft skills development: agile methodologies, documentation practices, teamwork, code reviews.
- Opportunity for full-time employment for high-performing interns.
- Exposure to international engineering processes.

3. General Requirements for Students

- Currently pursuing a degree in Computer Science, Computer Engineering, Electronics
- Engineering, Mechatronics, or related fields.
- Basic understanding of programming concepts and data structures.
- Familiar with software versioning system as Git
- Willingness to learn new technologies.
- Good communication skills.
- Passion for building real products and solving technical problems.

4. Technical Topics

4.1. Embedded Software Engineering Internship

4.1.1 Prefered Background

- Proficiency in C/C++.
- Understanding microcontrollers and hardware fundamentals - UART, SPI, I2C.
- RTOS and Embedded Linux fundamentals.

4.1.2 Topics

4.1.2.1 Software Development Process Improvement

Goal

- Improve the overall software development process by enhancing and harmonizing development scripts, implementing unit testing, and integrating static analysis.

Descriptions

- Aanalyzing, developing, and standardizing scripts to automate workflows and increase development efficiency.
- Designing and implementing unit tests to strengthen code quality and ensure reliable functionality across software modules.
- Integrating and optimizing static analysis tools to enforce coding standards and maintain high-quality, maintainable code.

4.1.2.2 Feasibility Study for Performance-Enhancing Scanner Features

Goal

- Evaluate the feasibility of new intelligent features—such as auto-learn, background auto-detection, and reading-surface detection—to enhance HHS scanner performance.

Descriptions

- Researching and analyzing advanced algorithm or methods that enable scanners to automatically learn optimal settings based on captured data.
- Examining techniques for detecting background or reading surfaces to improve decoding accuracy under varying environments.
- Explores technical constraints, performance impacts, and potential implementation paths for integrating these features into current HHS scanner platforms.

4.2. Automation Test Engineering Internship

4.2.1 Prefered Background

- Python and/or Robot Framework.
- Basic testing knowledge.

- Understanding QA processes and methodologies.
- Defect reporting and analysis.

4.2.2 Topics

4.2.2.1 Automation test script & error resilience

Goal

- Build a highly stable automated test suite with self-healing capability, enabling the system to automatically recover from unexpected exceptions during test execution.

Descriptions

- Convert and enhance manual test cases into automated Robot Framework scripts using existing keywords and libraries while standardizing test structure and organization.
- Design and implement a centralized exception-handling mechanism that classifies errors by type and severity while standardizing error messages to improve debugging efficiency and root-cause analysis.
- Implement automatic reset and recovery mechanisms to return the product to a known stable state when system-level errors occur, ensuring the test suite continues running instead of stopping due to unexpected failures.

4.2.2.2 Tooling evolution

Goal

- Enhance internal user experience of testing tool and integrate containerization technologies to standardize and stabilize the test execution environment.

Descriptions

- Redesign the testing tool dashboard to improve usability and clarity by providing real-time visualization of test execution status, progress, and summaries while enhancing overall layout and interaction flow for daily testing activities.
- Build a Sanity Test configuration module that enables users to quickly select and group critical test cases for rapid post-deployment execution to validate overall system stability.
- Design and implement a “One-click Dockerization” feature that automatically packages test scripts and dependencies into Docker containers and enables consistent, conflict-free test execution across machines.

4.2.2.3 Datalogic logging for Robot Framework

Goal

- Optimize test observability to enable fast and accurate root cause analysis during automated test execution.

Descriptions

- Design a centralized mechanism to collect and store logs after each test execution, enabling efficient filtering and analysis by test suite, test case, error type, and execution time.
- Design a lightweight dashboard to visualize test execution history by displaying key metrics such as run history, failure rates and trends, and top failing test cases to improve visibility into test stability and recurring issues.

4.2.2.4 Edge DevOps infrastructure: K8s on raspberry pi cluster

Goal

- Provide hands-on Edge DevOps experience by building, testing, and automating applications on a Raspberry Pi–based Kubernetes cluster with CI/CD integration.

Descriptions

- Focus on implementing a complete Edge DevOps workflow using a Raspberry Pi–based Kubernetes cluster.
- Cover Kubernetes and DevOps fundamentals, cluster setup and management, packaging and execution of automation tests in Kubernetes, and building a Jenkins CI/CD pipeline to automate test execution, log collection, and reporting in a real-world Edge Computing environment.