Rohan Nagesh

rnagesh@seas.upenn.edu • +1(445)-256-0613 • https://www.linkedin.com/in/rohannagesh •Philadelphia, USA

EDUCATION

University of Pennsylvania, Philadelphia, USA May 2025
MS in Electrical Engineering
Relevant Coursework: • Embedded Systems (Smart Devices) • Introduction to Networks and Protocols • Digital Integrated Circuits and VLSI Fundamentals • IoT Edge Computing • Chips Design • Computer Organization and Design

The National Institute of Engineering, Mysuru, India September 2020
Bachelor of Engineering in Electronics and Communication Engineering
Relevant Coursework: Microcontrollers • Microprocessors Systems • Operating Systems • Embedded Systems • C++ using Data Structures • Verilog • VHDL •Programming in C & Data Structures

SKILLS

Programming Languages and Software Tools: C, Python, Xilinx Design Suite, Keil Microvision, Python, Arduino IDE, Verilog HDL, Embedded C, TPT, UDE, Inca, Request One, ECU.Worx, DGS-SCM, Doors, Altium, Linux, CircuitLab
Electrical Equipment: DMM(Multimeter), Oscilloscope, Spectrum Analyzer, Function Generator
Relevant Skills: Torque structure, ISO26262, Model-based software development, Digital Circuits, Embedded software design and testing, VLSI Design, EV architecture, Misra C, Internet of Things

WORK EXPERIENCE

Cirrus Logic, Austin, TX May 2024 – Present
Embedded Systems Engineer Intern
- Engineered and implemented comprehensive system testing frameworks using modular Python scripts for pre-silicon (FPGA) and post-silicon (IC), ensuring the quality of camera autofocus and optical image stabilization ICs.
- Conducted in-depth root cause analysis and debugging with I2C traces via logic analyzer, identifying critical issues.
- Developed robust test cases from system requirements, enhancing the test framework to ensure rigorous testing of intended functionality in both ideal and non-ideal environments.
- Automated testing processes by developing Python code and utilizing inheritance, enhancing the existing test framework to reduce tester effort by 20%. Owned 45% of the test cases for a project.
- Performed board bring-up for test setups, referencing schematics and datasheets to ensure proper configuration and functionality.
- Automated test execution using Jenkins, improving efficiency and reliability, and tracking progress using Jira.

Bosch Global Software Pvt Ltd, Bengaluru, India Jan 2022 – Jul 2023
Software Engineer
- Developed and implemented Level 2 Functional Safety software for Electrical, Hydrogen, and Thermal systems of Fuel Cell Control Units of Fuel Cell Electric Vehicles (FCEV), meticulously aligning with ISO 26262 standards and employing embedded C, model-based software design, auto code and Misra C principles.
- Created comprehensive software documentation and precise specifications for modules within the FCEVs. Significantly improved testing efficiency by 40% through the innovative use of modular Python scripts for Time Partition testing, and meticulously performed Unit, System, and PIL and HIL Testing to ensure software quality.
- Provided active mentorship to new team members, facilitating their growth in developing and rigorously testing FCEV functional safety software.

Associate Software Engineer Jan 2020 – Dec 2021
- Designed, configured, and meticulously validated Level 3 Functional Safety software, focusing on hardware safety monitoring for Commercial Electric Vehicles.
- Conducted thorough requirement analysis and executed HIL Testing to identify deviations from requirements and weak points ensuring robustness and reducing development effort by 30%.
- Implemented streamlined routines for firmware flashing and re-flashing onto the bootloaders of ECUs in vehicles, contributing to more efficient software maintenance and updates.
- Underwent intensive training encompassing embedded software development for ECUs of vehicles communication protocols like CAN and FlexRay, Requirements handling, Software development life cycle, and comprehensive Functional Safety practices.

PROJECTS

Petpal January 2024 – May 2024
- Developing PCBA hardware integrating an Arm Cortex-based SAMW25 MCU with SEN-10245, SHTC3-TR-10KS, and LSM6DSL sensors, alongside BQ24075 power management, to maintain consistent 5V and 3.3V operation utilizing custom-designed buck and boost circuits in Altium.
- Engineering firmware employing FreeRTOS and Embedded C to enable sensor data acquisition via I2C for comprehensive pet well-being assessment, featuring over-the-air firmware update (FOTA) capability and user to debug via serial port.
- Implementing dynamic actuator control through PWM signals for precise pheromone dispersion and adaptive temperature modulation using a fan, prioritizing pet comfort.
- Architecting a streamlined software framework for real-time data processing and user interaction via an intuitive GUI, empowering pet owners with comprehensive insights via a cloud-based dashboard leveraging Wi-Fi, NodeRed, and MQTT
- **Won the first-place award from Microchip in the hackathon as a part of the final project demonstration.**

Radarduino November 2023 – December 2023
- Devised an Advanced Radar System that utilized a rotating ultrasonic sensor mounted on a stepper motor to detect the angle and distance of approaching objects.
- The system comprised two Arduino boards managing the motor, LCD via SPI, and sensors, communicating via UART for which drivers were programmed in bare metal Embedded C.
- The angle-distance data was visualized on an animated LCD, featuring both automatic and manual rotation modes using a joystick.
- Development followed the V-model product development life cycle, incorporating a modular approach for efficient teamwork.

Custom Linker Scripts, Makefiles, and Assembly Programming Oct 2023 – December 2023
- Developed custom linker scripts to optimize memory utilization for ARM Cortex-M0, strategically allocating memory in RAM or Flash and enabling seamless compilation of code across multiple C files.
- Automated the build process by designing efficient Makefiles, streamlining the creation of executables and improving development efficiency.
- Implemented a reset routine in Assembly to boot the system, configuring the vector table address in the program counter and the main stack pointer address in the stack pointer for reliable system initialization.