CURRICULUM VITAE

CONTACT INFORMATION

Title: **Master of Engineering**

Candidate information:

Full name: Nguyen Trong Quang

Birthday: 9th September 1988

Gender: Male

Marital status: Married

Additional information: Being a great **Master of Engineering** is not only a means of living, but also a deep indulgence and non-stop learning. With my qualifications, mature and professional outlook, I'm very self-confident that my skills and hard-working attitude will make a valuable contribution into profitability of the company. Besides, I'm always willing to encounter the challenging because I strongly believe: "There is nothing impossible for those who have strong will"

Contact information:

Address: 02 N4 Street, Son Ky Ward, Tan Phu District, Ho Chi Minh City, Vietnam

Mobile: 0356188981

Email: ntquang0909@gmail.com

CAREER OBJECTIVE

Desired position: **Manager / VP / Head / Architect / Director / CTO**

Industries: Embedded Software, Robotics, AI, Consultant, R&D, Full Stack, Data Science, Blockchain, DevOps/DevSecOps, Solution/Technical/Software/Cloud Architect, CyberSecurity, Big Data/Data Lake/Data Warehouse, DBA

Job status: Full time / Part time / Remote / Hybrid / Freelance

Type of employment: Permanent

Career objective: Becoming a valued employee of a company and have a special career that can be proud of

Desired salary: **Negotiable**

EDUCATION LEVEL

Sept 2011 - Nov 2014 **Ho Chi Minh City University of Technology -** Vietnam

- Faculty: Electrical - Electronics Engineering
- Major: **Automatic Control Engineering**
- Level: **Master**
- Classification: Good (GPA: 7.20)

Sept 2006 - Apr 2011 **Ho Chi Minh City University of Technology -** Vietnam

- Faculty: Electrical - Electronics Engineering
- Major: **Power Engineering**
- Level: **Engineer**
- Classification: Good (GPA: 7.08)

WORKING EXPERIENCE

Jan 2022 – Now **NEXON DEV VINA -** Korea

Full time job: **Software Development Manager**

Project: **Data labeling and annotation**

Achievements:

- Managing a software development team, building processes, managing projects, problem-solving and release schedules in delivering solutions that meet high standards for customer experience and operational excellence
- Working with cross functional stakeholders, communicating with technical and non-technical stakeholders across multiple business departments
- Agile/Scrum management methodology, inspire, motivate and encourage the team leads
- Developing cloud software services and design for scalability, performance and reliability, hands-on architectural or distributed systems, releasing and maintaining large-scale services or applications
- Professional software engineering practices, best practices for the full software development life cycle, including coding standards, code reviews, source control management, build processes, testing, and operations
- Provide technical estimates and advice to stakeholders and senior management when determining project priorities and feasibility
- Provide technical guidance to scrum team while planning and executing software development projects
- Coach team in high performing software development and maintain uniform high standards across the team
- Use data, evidence, documentation, discussion and company objectives as a basis for determining and executing development strategy across the teams

Oct 2019 – Jan 2022 **Pizzlysoft -** Korea

Full time job: **Senior Solutions Architect**

Project 1: **CyberSecurity**
Project 2: **Food Order Management System**

Achievements:

- Cloud (AWS, GCP, Azure, DO)
- Infrastructure Architecture from code to customer (Code, App/Web Server, Ingress Controller, API Gateway, Load Balancer, App Security, DNS, DDoS, CDN, Customer)
- Zero Trust Security
    + Authentication and Authorization: Manage user, app, and service identities and authorize them to perform actions with HTTP Basic authentication, JSON Web Tokens (JWTs), and OAuth 2.0, OpenID Connect through integrations with identity providers (IdPs) such as Okta and Azure Active Directory (AD), Ping... Add single sign-on SSO to apps from any OpenID Connect-compatible provider, apply positive security using whitelists based on IP address, client IDs, JWT claims, or blacklist-specific IP ranges
    + Data Encryption and Integrity: Secure communications regardless of location with authentication and encryption to ensure validity, confidentiality, and integrity of data through TLS passthrough, SSL/TLS termination, and mTLS
    + Access Control and Access Policy: Align with organization's security needs and enable self-service and governance across multi-tenant teams through role-based access control (RBAC) and custom resource definitions (CRDs)
    + Monitoring and Observability: Share metrics and logs, gain actionable insights into the security posture of apps, APIs, and infrastructure through prebuilt integrations with favorite observability tools, including OpenTelemetry, Grafana, Prometheus, Datadog and Jaeger
    + WAF and DoS Protection: App and API Security using WAF (Command Execution Vulnerabilities, OWASP Top 10, Information Leakage Vulnerabilities, Buffer Overflow Vulnerabilities) and DoS (Flood Attach, Slow POST, Slowloris, Challenger Collapsar, Distributed Attack, Slowread)
    + API and microservices management benchmark, WAF security performance test with gigaom/vegeta
- Ingress controller and service mesh (Nginx Ingress Controller, Nginx Service Mesh, Nginx App Protect, cert-manager, let's encrypt, Auth2 Proxy)
    + Layer 7 (HTTP/HTTPS, HTTP/2, gRPC, WebSocket) and Layer 4 (TCP/UDP) load balancing with advanced distribution methods and active health checks
    + Blue-green and canary deployments to avoid downtime when rolling out a new version of an app

    + Dynamic reconfiguration, rate limiting, circuit breaking, and request buffering to prevent connection timeouts and errors during topology changes, extremely high request rates, and service failures
- Web Application Hosting in the AWS Cloud
    + DNS services with AWS Route 53: Provides DNS services to simplify domain management
    + Edge caching with AWS CloudFront: Edge caches high volume content to decrease the latency to customers
    + Edge security for AWS CloudFront with AWS WAF: Filters malicious traffic, including cross site scripting (XSS) and SQL injection via customer-defined rules
    + Load balancing with Elastic Load Balancing (ELB): Spread load across multiple Availability Zones (AZ) and AWS Auto Scaling groups for redundancy and decoupling of services
    + DDoS protection with AWS Shield: Safeguards infrastructure against the most common network and transport layer DDoS attacks automatically
    + Firewalls with security groups: Moves security to the instance to provide a stateful, host-level firewall for both web and application servers
    + Caching with AWS ElastiCache: Provides caching services with Redis or Memcached to remove load from the app and database, and lower latency for frequent requests
    + Managed database with Amazon Relational Database Service (AWS RDS): Creates a highly available, multi-AZ database architecture with six possible DB engines
    + Static storage and backups with Amazon Simple Storage Service (AWS S3): Enables simple HTTP-based object storage for backups and static assets like images and video
    + Simplify SSL certificates management using ACM
    + Use NAT gateways in each public subnet enable Amazon Elastic Compute Cloud (AWS EC2) instances in private subnets to access the internet
    + Using a shared storage service, like Amazon Elastic File System (AWS EFS), if servers have access to shared files
    + Disaster recovery options in the AWS cloud (backup and restore, pilot light, warm standby, multi-site active/active)
- Data Warehousing with Amazon Redshift
    + AWS for Data, Getting Started with Amazon Redshift
    + Setting Up Your Data Models and Ingesting Data, Data Transformation Strategies
    + Scaling and Performance Optimizations
    + Amazon Redshift Machine Learning
    + Collaboration with Data Sharing, Securing and Governing Data
    + Migrating to Amazon Redshift, Monitoring and Administration
- Migration and modernization strategy for integrated eligibility systems with AWS
- Google Cloud Platform (GCP)
    + Computing and GCP basics, resources & resource hierarchy, introduction to IAM, google cloud SDK

    + Compute: basics of enterprise computing, concepts of google compute engine, managed instance groups, containers and its orchestration using k8s, concepts of google kubernetes engine, google app engine, cloud run and cloud functions
    + Networking: GCP network, GCP network components (VPC, subnets, routes, DNS,...), firewalls, DNS and load balancing
    + Storage: storage options on GCP, google cloud storage (GCS), database options (Cloud SQL, Spanner, Firestore, Disks, BigTable, BigQuery), data transfer
    + Auditing and logging: google cloud operations suite (cloud monitoring, cloud logging, cloud APM, cloud error reporting)
- Build kubernetes cluster with rancher, vagrant, virtualbox, docker
- Deploy an application on kubernetes with helm/kubectl/service mesh
- Infrastructure as code (Terraform, CloudFormation, Ansible,...)
- nDPI (open-source high-speed Deep Packet Inspection library)
- PF_RING (high-speed packet capture, filtering and analysis)
- n2disk (10/40 Gbit network traffic recorder with indexing capabilities)
- ntopng (high-speed web-based traffic analysis and flow collection)
- Suricata (free and open source, mature, fast and robust network threat detection engine)
- DPDK (Data Plane Development Kit that consists of libraries to accelerate packet processing workloads)
- Protocol buffers (language-neutral, platform-neutral, extensible mechanism for serializing structured data)
- gRPC (high performance, open source, general-purpose RPC framework)
- gRPC-gateway (gRPC to JSON proxy generator following the gRPC HTTP spec)
- Performance, memory safety and concurrency with Go, Rust programming language, web performance optimization
- Monitoring (Prometheus, AlertManager, Grafana)
- Logging (Fluentd/Fluent-bit, Elasticsearch, Kibana)
- Tracing (OpenTelemetry Collector, LightStep/DataDog)
- Drone CI/CD
- Camunda, cadence, zeebe (Workflow and Decision Automation Platform)
- Microservice architecture pattern
    + Application architecture patterns
    + Messaging style patterns
    + Reliable communications patterns
    + Service discovery patterns
    + Transactional messaging patterns
    + Data consistency patterns
    + Business logic design patterns
    + Querying patterns
    + External API patterns
    + Security patterns
    + Observability patterns
    + Deployment patterns

- Interprocess communication in a microservice architecture
    + Synchronous remote procedure invocation (REST, gRPC, partial failure using the circuit breaker, service discovery)
    + Asynchronous messaging (message broker, competing receivers and message ordering, duplicate messages, transactional messaging) and benefits of communicate using asynchronous messaging in order to increase availability
    + Reliably sending messages as part of a database transaction
- Managing transactions with sagas in a microservice architecture
    + Using the Saga pattern to maintain data consistency
    + Coordinating sagas using choreography and orchestration
    + Using countermeasures to deal with the lack of isolation
- Developing business logic with event sourcing
    + Using the Event sourcing pattern to develop business logic
    + Implementing an event store
    + Integrating sagas and event sourcing-based business logic
    + Implementing saga orchestrators using event sourcing
- Implementing queries in a microservice architecture
    + The challenges of querying data in a microservice architecture
    + Queries using the API composition pattern
    + Queries using the Command query responsibility segregation (CQRS) pattern
- External API patterns
    + Designing APIs that support a diverse set of clients
    + Applying API gateway and Backends for frontends patterns
    + Designing and implementing an API gateway
    + Implementing an API gateway using GraphQL
- Developing production-ready services
    + Developing secure services
    + Designing externalized configuration services
    + Designing observable services (health check API, log aggregation, distributed tracing, application metrics, exception tracking, audit logging)
    + Simplifying the development of services by applying the microservice chassis and then service mesh (istio, linkerd, conduit,...)
- Deployment pattern (serverless (lambda functions), containers (docker), virtual machines, and language-specific packages)
- Deploying services on kubernetes with zero-downtime and using a service mesh to separate deployment from release

Mar 2018 – Oct 2019 **SNAP Innovations -** Singapore

Full time job: **Technical Leader**

Project: **Retail Trade**

Achievements:
- Spring Boot (starting point for building all Spring-based applications)

+ Build spring boot with maven and java
+ Using spring initializr
+ Build anything: REST API, WebSocket, web, streaming, tasks and more
+ Rich support for SQL and NoSQL (accessing data with MySQL, accessing data with JPA,...)
+ Supports modern messaging middleware (ZeroMQ, Kafka, ActiveMQ and RabbitMQ)
+ Embedded runtime support (Tomcat, Jetty, and Undertow)
+ Simplified security
+ Works in the favorite IDE (Spring Tool Suite, IntelliJ IDEA, and NetBeans) and Java programming

- MERN (MongoDB, Express, React, Node) Web Development
    + HTML, CSS and JavaScript
    + Advance Frontend with ReactJS
    + Backend with Express and MongoDB
    + MVC (Model-View-Controller) architecture pattern
- Python, Django, Django REST framework Web Development
    + Build websites with Python and Django
    + Build web APIs with Django and Django REST framework
    + Production websites with Python & Django
    + MVT (Model-View-Template) architecture pattern
- LAMP (Linux, Apache/Nginx, MySQL/MariaDB, PHP/Python) Web Development
- Blockchain Technology
    + Transactions are grouped into blocks and each block is connected to the previous, forming a chain of blocks
    + Blockchain Core Components: Ledger, Smart Contracts, Nodes
    + Blockchain solutions mainly span over three layers: Networks, Platforms and Applications
    + The Networks layers includes the infrastructure and storage for blockchain solutions, Public or Permissionless Blockchain Network (Bitcoin and Ethereum) and Private or Permissioned Blockchain Network
    + The Platforms layers is that allow developers to build blockchain applications and solutions. The available platforms vary depending on the selected type of network (Ethereum, Polygon, Hyperledger Besu, Quorum, Corda, Hyperledger Fabric, Cardano, Solana). One of the most common used platforms globally are Ethereum for building solutions on public network and Hyperledger Fabric for building solutions on private networks
    + The Applications layers contain Blockchain applications (Metaverse, NFTs, dApps, Crypto Exchanges, DEX, DeFi, DAO, Self-Sovereign Identity). One example is Decentralized Applications (dApps), which are applications that run in a blockchain network rather than on a central server. Another application is Decentralized Autonomous Organizations (DAOs), which are online organizations organized by smart contracts and managed by business owners in a distributed form rather than a centralized hierarchy

- Designing a Blockchain solution and architecture of a web 3.0 application
    + Blockchain Application Architecture – Centralised: like traditional web 2.0 applications, the user interface is querying the Blockchain with centralised back-end system in the middle
    + Blockchain Application Architecture – Decentralised (dApps): digital applications or programs that live and execute on a Blockchain or peer-to-peer network instead of a single computer and back-end system including smart contracts, making the application more fault-tolerant. Thus, it is recommended to use dApps when user experience and decentralization are prioritized
    + Build Web Applications on top of the Ethereum Blockchain
- Ethereum (Building Smart Contracts and DApps)
    + What Is Ethereum, Ethereum Basics
    + Ethereum Clients, Cryptography, Wallets, Transactions, Smart Contracts and Solidity, Smart Contracts and Vyper, Smart Contract Security, Tokens, Oracles, Decentralized Applications (DApps), The Ethereum Virtual Machine, Consensus
- Sequelize ORM
- Apollo GraphQL (Apollo Server, Apollo Federation)
- Prisma (Next-generation Node.js and TypeScript ORM)
- RapidJSON (a fast JSON parser/generator for C++ with both SAX/DOM style API)
- A small, clean, linux-only thread-pool implementation using epoll with support for sockets, scheduled callbacks
- Boost C++ libraries
- QuickFIX (a free and open source implementation of the Financial Information Exchange (FIX) protocol
- Build trading systems, trading algorithms and related technologies for financial markets (Forex Exchange, Crypto Exchange)
- Agile/Scrum Methodology for software development life cycle (SDLC)

Jan 2017 – Mar 2018 **FPT Software Co., Ltd -** Vietnam

Full time job: **Senior Software Developer**

Project 1: **Automation Test Framework and Multimedia System Test**

Achievements:
- Fuego test system (jenkins, docker container, shell script, python…)
- Development environment of 2D/3D graphics systems (yocto/open-embedded, open-gles/egl, Qt5, wayland/weston, mesa-egl, alsa…)
- Audio/video automation test on linux (Renesas' boards) based on fuego, ffmpeg, gstreamer, audacity…
    + Audio/video stream, encode audio/video, decode audio/video, transcode audio/video, capture audio/video, audio/video driver
- Object recognition, image processing in OpenCV, algorithms to compare images using OpenCV (sift/surf algorithm, histogram algorithm…)

- C/C++, multi-threading programming, compiler, debugger (IDE: green hills, cubesuite+, CCStudio) on Windows based on Renesas, Texas Instrument micro-controller architecture (V850, RH850, AM335x...)
- Writing batch scripts (.bat)
- AUTOSAR standards
- Automotive communication bus (CAN, LIN, Flexray, Ethernet, MOST…), emulator and development tool (Vector CANoe, Vehicle Spy)
- Integrating Matlab/Simulink/Stateflow models in CANoe's simulation environment (analysis, simulation, test, diagnostic), hardware-in-the-loop (HIL) simulation, .dll file for real-time interaction between CANoe and Matlab/Simulink/Stateflow
- QNX Software Development Platform and QNX Hypervisor
- Scalable Open Architecture for the Embedded Edge (SOAFEE) and AosEdge Edge Orchestration
- Real time operating system (OSEK/VDX, Microsar OS, TI-RTOS, FreeRTOS…)
    + Standard types of threads (interrupt service routine-ISR, tasks, idle)
    + Schedulers (preemptive scheduler, time-slice scheduling...)
    + Communication mechanism (semaphores, message queues, queues...)
    + Critical region mechanisms (mutexes, gates, locks...)
    + Timing services (clocks, timers...)
    + Power management (for low power devices, power management knows the state of the device)
    + Memory management (variable-size heaps, fixed-size heaps...)
    + Peripheral drivers (UART, SPI, I2C...)
    + Protocol stacks (BLE, WiFi...)
    + File system (FatFS...)
    + Device management (exception handling, boot...)

Mar 2015 – Dec 2016 **FPT Telecom JSC -** Vietnam

Full time job: **Research and Development – R&D Engineer**

Project 1: **Virtual Desktop Infrastructure - VDI**

Achievements:
- Red hat enterprise virtualization, KVM hypervisor, SPICE (spice protocol, spice client, spice server, spice guest), red hat openstack platform
- FFmpeg - a complete, cross-platform solution to record, convert and stream audio and video (video/audio codec, video/audio encoder, video/audio decoder, video/audio bitrate, resolution, profile…)
- RabbitMQ - a messaging broker, an intermediary for messaging. It gives applications a common platform to send and receive messages, and messages a safe place to live until received. RabbitMQ supports several messaging protocols, directly and through the use of plugins (AMQP, STOMP, MQTT, HTTP)

+ Producer – queue – consumer
+ Distributing tasks among consumers in sequence
+ Sending messages to many consumers at once
+ Receiving messages selectively
+ Receiving messages based on a pattern
+ RPC (remote procedure call) implementation

- Embedded linux kernel and driver
  + Configuring, compiling and booting the kernel
  + Linux kernel modules
  + Developing linux device drivers
  + Porting the linux kernel to a new hardware platform
  + Linux kernel debugging
- Android system
  + Compiling and booting android
  + Porting android to a new board
  + Device development with android (developing and debugging with ADB, android build system, building a library, develop a java native interface-JNI library, write an application with the android SDK and its API)
  + Qt for android, Qt Quick and QML for android, java programming
- Building embedded linux systems with yocto/open-embedded (Board: raspberry pi, minnowboard max, dragonboard 410c, beaglebone black..)
  + Creating layers, adding a new machine, writing and extending recipes
  + Creating custom images, integrating the board in a board support package-BSP, application development with the poky SDK
- MySQL database and high performance technology
  + Build MySQL community server
  + Build MySQL connector/C++ (X DevAPI, X DevAPI for C, legacy JDBC4-based API)
- Anaconda (data science toolkit working with thousands of open-source packages and libraries)
  + The PyCharm (Python IDE)
  + The fundamentals (jupyter, pandas, scipy, numpy,...)
  + Machine learning (keras, tensorflow, pytorch, scikit-learn,...)
  + Data visualization (matplotlib, bokeh, plotly, holoviz,...)
  + Image processing (pillow, scikit-image, opencv,...)
  + Natural Language Processing (NLTK, gensim, spacy,...)
- Machine learning with Matlab/Python (discover patterns and build predictive models with engineering, manufacturing, and financial data)
  + Accessing and loading data, preprocessing data, deriving features, and training and refining models
  + Unsupervised learning-finds hidden patterns or intrinsic structures in input data (hard and soft clustering algorithms, common dimensionality-reduction techniques for improving model performance)
  + Supervised learning-trains a model on known input and output data so that it can predict future outputs (classification and regression algorithms, techniques for model improvement,

including feature selection, feature transformation, and hyperparameter tuning)

- Big data with Matlab/Python
  + Use matlab datastores to access data, datastores support a variety of data types and storage systems (images, spreadsheets, tabular data files, custom files, databases-SQL/NoSQL, Hadoop/HDFS/Spark…)
  + Use matlab tall arrays and distributed arrays to explore, process, and analyze data. Tall arrays for statistics and machine learning, visualization. Distributed arrays for math and matrix manipulation
  + Use advanced mathematics and machine learning algorithms in matlab to develop predictive models with big data
- Deep learning with Matlab/Python (training a network from scratch, using transfer learning to train an existing network, adapting a pretrained network for semantic segmentation)
  + Access the latest pretrained models (GoogLeNet, VGG-16, VGG-19, AlexNet, ResNet-50, ResNet-101, and Inception-v3...)
  + Create and configure network layers
  + Adapt network architectures, including convolutional neural network (CNN), directed acyclic graph (DAG), and long short term memory (LSTM)
  + Select the best training options and algorithms
  + Use data augmentation and Bayesian optimization to improve training accuracy
  + Incorporate spectrograms for speech recognition
  + Automate ground-truth labeling using apps
  + Use NVIDIA GPUs for GPU programming: Accelerate training using multiple GPUs, the cloud, or clusters
  + Use functions and tools to visualize intermediate results and debug deep learning models
  + Work with models from other frameworks (Caffe, TensorFlow-Keras...)
  + Build model with google colab, kaggle
- Natural Language Processing-NLP with Matlab/Python (data analytics with human language data)
  + Automating the classification of reviews based on sentiment, whether positive or negative
  + Counting the frequency of words or phrases in documents and performing topic modeling
  + Developing predictive equipment maintenance schedules based on sensor and text log data
  + Automating labeling and tagging of speech recordings

Sept 2012 – Mar 2015 **Metran Vietnam Co., Ltd -** Japan

Full time job: **Embedded Software Engineer**

Project 1: **Continuous Positive Airway Pressure-CPAP Machine**
Subject of thesis (master): **Control of a self-balancing Robot on a ball**

Achievements:

- Unix/Linux operating system
    + Linux kernel architecture (hardware, kernel space, system call, user space…)
    + Process and thread programming in linux, how process and thread communicate with each other, IPC-interprocess communication (modes: pipe, fifo, message queue, shared memory, semaphore, socket...)
    + Process scheduling in linux (scheduling classes such as completely fair scheduling class, real-time scheduling class… Scheduling policies such as round robin-RR, first come first served-FCFS, shortest job first-SJF, priority-PRI scheduling…with preemptive/non-preemptive)
    + Multi-threading programming and models of multi-threaded applications (boss/worker model, peer model, pipeline model, producer-consumer model)
    + Main issues with multi-threading applications (deadlock, race condition) and synchronization mechanisms
    + POSIX thread libraries
    + Producer – consumer with pthreads (synchronization: queue, state variable, busy waiting, mutex, condition variable ...)
    + Dining philosophers with pthreads (solutions: asymmetric, waiter…without deadlocks and starvation)
    + Network system (OSI, TCP/IP reference model), socket programming (client/server model, communicate locally or across a network, TCP sockets, UDP sockets,…)
    + Writing shell scripts (.sh)
    + Signals (signal handlers, system calls: alarm(), pause()…)
    + Memory mapped I/O (instead of using read/write system calls)
- Design (using starUML software), design patterns (singleton, factory, adapter...)
- Data structure and algorithms (linked list, stack, queue, search, sort, graph, tree, recursion, dynamic programming,...)
- Building embedded linux systems with buildroot (cross-compilation toolchain, root filesystem generation, kernel image compilation and bootloader compilation)
    + Internal toolchain, external toolchain
    + Managing the linux kernel configuration
    + Integrating new packages(components) in buildroot
    + Root filesystem customization
- Object oriented programming (OOP) in C++ (abstraction, encapsulation, inheritance, polymorphism, overloading)
    + Class and object, constructor and destructor, copy constructor, friend function, friend class, inline function, this pointer, static member…
    + Base class, derived class, virtual base class, virtual function, pure virtual function, virtual destructor…
    + Function overloading, operator overloading, overloadable/non-overloadable operator, constructor overloading…
    + Array, string, pointer, reference and dynamic memory allocation, namespace, exception handling, template…

+ STL- standard template library (vector, list, map, multimap, bitset, algorithm, iterator, array...), C++ standard library
+ How to write makefile effectively (preprocessor, compiler, object file, linker, static lib, shared lib,...)
+ Debug logging (glog) and unit tests (googletest, cppunit,...), gdb to debug the C/C++ program
+ Detect memory leak with valgrind
+ Smart pointer (unique_ptr, shared_ptr, weak_ptr,...)

- C/C++, multi-threading programming, compiler, debugger (IDE: Qt creator, eclipse) on linux (Board: friendlyARM, AT91SAM9x5, i.MX6, STM32F4...)
- Building Qt for embedded linux (Qt libraries and Qt creator), Qt for non-graphical applications
+ QtCore-event loop with an original signal/slot mechanism, data structures, threads, regular expressions
+ QtNetwork-networking (TCP, UDP clients and servers made easy, HTTP, FTP support)
+ QtXml-SAX/DOM parsing of XML files
+ QtXmlPatterns-XPath, XQuery, XSLT and XML schema support
+ QtGui-GUI widgets
+ QtMultimedia-low-level multimedia functionality
+ QtSQL-query various databases
+ QtOpenGL, QtOpenVG, QtSvg...

- Qt Quick-QML programming for graphical user interface-GUI applications
+ QML object attributes
+ Integrating QML and C++ (exposing attributes of C++ classes to QML, embedding C++ objects into QML with context properties, interacting with QML objects from C++...)
+ Model/view programming, using C++ models with Qt Quick views
+ Qt Quick - states, transitions and animations

- Real-time for linux (RTAI, RT-PREEMPT, xenomai)
- Inertial measurement unit – IMU (gyroscope, accelerometer, magnetometer sensor), GPS/INS and kalman filter
- System modeling and identification (constructing mathematical models or create linear and nonlinear dynamic system models from measured input-output data)
- Control algorithm design and simulation using matlab/simulink/stateflow, generating and integrating C/C++ code from matlab/simulink/stateflow for deployment on an embedded device (PID, intelligent control, nonlinear control, adaptive control, optimal control, multivariable control, fuzzy logic, neural network, genetic algorithm...)
- Optimal state estimation using kalman filter (optimally estimate the internal states of a system in the presence of uncertain and indirect measurements)
+ Guidance, navigation and control applications (combining GPS and IMU measurements to synthesize position and velocity signals)

+ Computer vision (object tracking to predict an object's future location)
+ Together with LQR algorithm for LQG control
+ Nonlinear state estimators (extended kalman filters, unscented kalman filters, and particle filters)

- Robotics and Autonomous Systems
  + Sensors: camera, lidar, radar, ultrasonic, GPS, IMU, encoder and calibration of the sensors' coordinate systems, time synchronization using ROS/ROS2, filter using Kalman filter
  + Perception: object detection with deep learning (YOLO), object detection by point cloud processing (2D/3D Lidar), tracking detected objects, build and visualize 2D and 3D maps using Lidar SLAM or visual SLAM using orb-slam3 algorithm, localization state estimation methods such as Monte Carlo localization algorithm use a particle filter to estimate, fuse IMU and GPS sensor reading for accurate pose estimation
  + Planing and decision making: route planing, path planing, motion planing...typical planning algorithms include sampling-based methods such as rapidly exploring random tree (RRT) and RRT*, A*
  + Control: pure pursuit controller for path tracking, model predictive control, obstacle avoidance using reinforcement learning
  + ROS/ROS2
- Software management and source code control (svn, mercurial, git)
- Software development life cycle (V-Model)

Sept 2011 - July 2012 **3T Robotics Group -** Vietnam

Full time job: **Electrical - Electronics Engineer**

Project: **Humanoid Robot**

Achievements:

- C/C++ programming for Microchip PIC, STMicroelectronics ARM micro-controller
  + Data type, array, string, pointer, storage class, function, dynamic memory, preprocessor, type casting…
  + Data type: struct, union, enum, bit field…
  + File I/O, input/output, error handling and standard C library…
  + Recursion, variable argument, memory management
  + Void pointer, function pointer and state machine…
- Analyze a hardware schematic, PCB
- Microchip PIC, STMicroelectronics ARM micro-controller development tools, compiler, debugger (IDE: MPLAB, MDK-ARM) and architecture
- Peripheral communication programming (GPIO, I2C, SPI, UART, ADC, DAC, DMA, WDG, TIM, PWM, EXTI, USB Interface...)
- C and assembly programming for Atmel 8051 micro-controller
- Radio frequency (RF) remote control system

- Atmel 8051 micro-controller development tools, compiler, debugger (IDE: Keil C51) and architecture
- C/C++ programming for Texas Instrument DSP micro-controller
- Layout a printed circuit board (PCB), design and simulate an electronic circuit using proteus, orCAD software
- Computer - based measurement and control system
- Texas Instrument DSP micro-controller development tools, compiler, debugger (IDE: CCStudio) and architecture

SKILLS

Professional knowledge - Good

- MicroControllers and embedded systems (DSP, ARM, PIC, 8051..)
- Unix/Linux/Android operating systems
- Real time operating system
- Advanced programming (C/C++, javascript, nodejs, python, golang, rust, java, PHP, solidity, matlab,...)
- Full-Stack Web Development (REST, GraphQL, gRPC, Websocket,...)
- Blockchain and Web 3.0
- Microservices Patterns
- Cloud, DevOps/DevSecOps and System Administrator
- Big data/data lake/data warehouse, predictive analytics, AI (machine learning, deep learning)
- Image processing, Computer vision
- Natural language processing
- Robotics and Autonomous Systems
- Model based design (Matlab/simulink/stateflow)
- System modeling and identification
- Control system design and simulation
- Optimal state estimation (Kalman filter)
- Methodology of scientific research
- Digital system design
- Electronics engineering
- Dynamic control of dynamic system
- Robot dynamics and control
- Intelligent control systems
- SCADA: design and analysis
- PLC
- Soft computing (fuzzy system, neural network, genetic algorithm)
- Nonlinear control
- Adaptive and robust control
- Optimal control theory
- Multivariable control

Independent studying and researching skills - Good

- Collected information from many sources: library, research books and internet to enrich my knowledge
- Well reading comprehension of english documents

English – Good

- Good in listening, speaking, reading and writing skills

Communication and presentation skills - Good

- Acquired experience about effective presentation methods
- Realized the importance of communication skills
- Took the course of communication and presentation skills

Team work, leading skills - Good

- Participated the green summer volunteer campaign as a team leader
- Worked as a member of youth union related to organizing social activities

Computer - Good

- Good at windows, word, excel, power point and technical software...
- Had knowledge of mail, internet and antivirus...

INTERESTS

- Listening to music, playing football, learning foreign languages, surfing the internet, swimming and meeting new people