nuertey_odzeyem@hotmail.com ● **Buffalo Grove**, Illinois ● github.com/nuertey

Nuertey Odzeyem – Seeking Flex, Contract, or Full-Time STRICTLY REMOTE Work And Willing to Relocate

SUMMARY:

- Loves solving problems generally. And specifically, loves to architect, design and code software, especially in the **Embedded** space and particularly for **Linux-like** OS targets. Highly experienced in writing **well-optimized** code in **C/C++,** especially in **C++11, C++17** and **C++20**.
- **Extensive Software Design Experience: Designed real-time** robotic software, computationally intensive software for minimization of Boolean functions and application development. Have designed using **Event-Driven** Architectures, **Multithreaded** Architectures, **Object Modeling Technique** (OMT), Modular Specification and other techniques. Experienced in all phases of software design.
- **Architectural Skills:** Successful record of scrutinizing and redefining vague problem definitions to obtain robust system designs. Possesses an unassuming and questioning nature combined with an expert knowledge base. **Analytical**, **adaptable**, able to **think independently** and **loves to learn**.

SKILLS:

O/S-Libraries:	ARM **Mbed OS** for **ARM Cortex-M Microcontrollers**, Embedded **Linux**, **Raspbian**, **VxWorks**, OpenWRT, **OpenRTOS**/**FreeRTOS**, OSE, pSOS, Solaris, Windows, UNIX, Ubuntu, Cygwin; **POSIX, STL, BOOST, C++11, C++17, C++20. PostgreSQL, sqlpp11, sqlpp11-connector-postgresql.**
Languages:	**C/C++, Python**, Perl, Shell Scripting, **Qt**, qmake, cmake, **Matlab, R,** VHDL, Assembly Languages, Lattice ABEL-HDL, XML, HTML
Engineering:	**NEB1DX** Future Nebula Board for IoT ecosystems prototyping, **Keil** µVision, Cypress **WICED**™ SDK, **RIOT** (The friendly OS for IoT); STM32 **Nucleo Dev Board Hobbyist**; FMEA, DFMEA, Iterative Development, Agile, Scrum, TDD, Object-Oriented Design
Software:	**TouchGFX Graphics Designer** for **STM32 MCUs**, SW4STM32 (**System Workbench** IDE for STM32), VirtualBox, Eclipse-based IDEs, Wind River Workbench, **GIT**, Subversion, Perforce, JIRA, TestTrack, Code Collaborator, Visual Studio, **CPPUnit, Log4cxx**, **fmtlib**, **spdlog**, Chainsaw GUI Log Viewer, SDEdit, Enterprise Architect, ScrumWorks, CruiseControl, Tomcat, Rose RealTime, Telelogic DOORS, CodeWarrior, OSE Illuminator, ClearCase, DDTS/ClearQuest, Coverity, Rational Rose, STP, Altera Maxplus, Maple, Matlab, PSpice, **Wireshark/Ethereal, Omnipeek/Etherpeek, strace, valgrind,** ProcommPlus, **Kermit**, HyperTerminal, **Boost.Process** (**to enable rapid product development cycles as it is portable between OSes**), OpenOffice, MS Office, **CAN, CANOpen, SocketCAN,** Modbus TCP/IP**, Cap'n Proto,** ProtocolBuffers, **AZMQ**, **ZeroMQ (zmq, czmq, zmqpp), MQTT, Yocto, OpenEmbedded,** bitbake, **SPI** and **I2C** Userspace encapsulations.

92% Select2Perform C++ Test (PreVisor Talent Measurement)
Scored at the Advanced (Master) Level of the Brainbench Advanced C++ Test

EXPERIENCE:

Beacon EmbeddedWorks 12/2021 – 4/2022
Senior Embedded Software Engineer - Contractor

- Tasked with implementing Root File System **Encryption** for an **i.MX 8M Mini** Gateway development board via the **LUKS2** (Linux Unified Key Setup) scheme.

- Tasked with implementing Root File System **Authentication** for an i.MX 8M Mini Gateway development board.

To achieve the above tasks, **rebuilt Linux Kernel images** as needed, **wrote custom bash scripts**, issued relevant Linux kernel commands such as **cryptsetup** and **veritysetup** and their various variants, customized **crypttab** and **fstab** as needed, **rebuilt custom intramfs images** as needed, and exercised the onboard **TPM2.0 chip** automagically and manually.

AST & Science LLC./Upwork® Global Inc. (Freelancer/Consultant) 7/2020 - 12/2020
Embedded Software Engineer / Flight Software (FSW) Contractor - FSW Team

- Tasked with importing third-party libraries such as libcsp, libparam, etc. into the client's embedded software ecosystem and its attendant build system in Keil µVision IDE. In addition to creating and logically structuring the requisite **git submodule repositories**, it also involved migrating compilers from ARMCC to **GCC arm-none-eabi** (GNU Arm Embedded Toolchain\9 2020-q2-update). Many code changes had to be made across the client's existing embedded software ecosystem/applications targetted for **STM32H743ZI** hardware, in order to make this migration work.

- Designed and coded up a **driver** for the BAT-P3 **Satellite Module** that will run on an ARM **Cortex-A7**-based Flight Computer. The BAT-P3 is a 6-8 cell Lithium-Ion battery system designed for battery life-time, easy integration, and safety. With 3 different battery configurations and up to 92 Wh in nominal capacity, the BAT-P3 is both flexible enough and sufficiently powerful for most **nano-** and **small-satellite missions**. The driver was designed to communicate from the Flight Computer to the BAT-P3 Module over **CSP/CAN link** and was primarily concerned with issuing parameter gets/sets and performing **telemetry monitoring**.

- Designed and implemented a **Software Emulator** for the BAT-P3 Satellite Module. For this software emulation, advantage was taken of the expressiveness of Boost.MSM State Machine functionality, and the nimbleness of various software timers, to mimic exactly how the BAT-P3 behaves in hardware. Also, an elaborate and well-documented parallel build system was constructed with the Meson/Ninja build frameworks targetted for Ubuntu and Raspbian (Raspberry Pi) Linux. These were then subsequently tested and deployed on both target OSes. Eventually, the completed Emulator itself was deployed on a Raspberry Pi 4 to successfully communicate with the driver outlined in the previous accomplishment.

Some Technologies Used:
ARM Cortex-based Microprocessors; **Ubuntu** Linux; **Raspbian (Raspberry Pi) Linux**; CubeSat Space Protocol (CSP); USB-CANmodul1 devices (connection of **CAN-bus** with PC via USB) and bridge software supporting it, such as tucCspBridge and nanoMCS; **Meson** build frontend; **Ninja** build backend; **PlantUML** (State Machine diagramming tool); **Boost.MSM**; C++11, C++17, **C++20**; **Template MetaProgramming**, **SFINAE** programming techniques.

Personal Projects 08/2019 – 12/2020

Implementing numerous **C++** and **Python** coding projects on personal **git repositiories**. Some of these projects include:

[1] Designed an Ubuntu Linux C++ application that implements a **volume-weighted average price (VWAP) trading algorithm** via a self-designed very fast type-driven trading message protocol. Messages were **serialized** and **deserialized** via self-designed **C++ template mechanisms** whilst sidestepping **network to host byte order issues**. Also, trades and quotes received, and the subsequently generated customer stock orders were written on-the-fly to a **PostgreSQL database** from within the C++ application. Two versions of the software were written, one utilizing C++17 and the **Boost ASIO** library for networking, and the other utilizing C++11 with my own self-designed encapsulations and abstractions for the networking segment.

[2] Demonstrated asynchronous **ZeroMQ TCP** and **IPC** socket communications integrated with **Boost.ASIO io_context/worker** thread between a server and client pair. Also, the complex continuous data exchanged between the client and server applications was transmitted with **protobuf** and **capn proto** serialization (I used **Nanopb** for an equivalent application on an embedded platform).

[3] Designed and coded a **C++ IOT project** which involved capturing the outputs of my cellphone's **accelerometer** and **gyroscope** sensors in realtime, and continuously plotting that device's position and inclination (**sensor fusion algorithm**) as it is being moved around. The capture application I wrote in **elegant C++17** with its protocol component in self-designed asynchronous **MQTT** (no reliance on external libraries such as Mosquitto, RabbitMQ or Paho MQTT) and its GUI plotting component in **OpenGL**. With the sensors being sampled in the **lower milliseconds range** and the code so optimized that elegantly, moving the phone updates its plot in **true realtime** with no discernable delays.

[4] Designed and coded a C++ encapsulation and abstraction of USB devices that enables userspace programs talk directly to attached USB gadgets without having to go through kernel USB device drivers. I used this abstraction to enumerate and display information for all the USB devices attached to my system and to communicate with my Verizon MiFi device.

[5] Wrote **Python scripts** (**Python 3**) to query and visualize data of the moment such as world-wide COVID-19 statistics distribution, market data time-series trends for certain raw materials, **Google Search** trending topics in any topic category world-wide, and news headline topics for most newspapers from most countries around the world. In addition to packages such as **Pandas**, **Numpy**, TensorFlow, Keras, **Requests**, **Json, Psycopg2**, and **Plotly**, exercised third-party APIs such as Google Drive Python API, Google Trends Python API, **Quandl** Python API, **Alpha Vantage** Python API, News API and **MapBox** API. Also experienced with using the **Socrata Open Data API (SODA)** to programmatically access, filter, query, and aggregate municipal datasets. Currently delving into Neural Networks and Deep Learning with Numpy, **TensorFlow** and **Keras** Python packages and experimenting with **Flask** and **Dash** Python **web development frameworks** whilst deploying and hosting resultant **web applications** on **Heroku cloud-based platform** and serving them out with **Gunicorn Python web server**.

[6] Familiar with the use of **Matlab** (and its cousins **GNU Octave** and **R**) for signal processing, plotting, and various filter designs. Examples include designing **Kalman** filters.

MP Consulting, IL   4/2019 – 7/2019

Contractor, Software Engineer

Employing C and **C++11**, implemented customer features such as WiFi, Ethernet, Bluetooth, MQTT, drivers for EPaper displays, custom waveform generation with PWMs to drive LED circuitry, and audio waveform playback on dev board modded speakers with customized DACs. Development took place on various STM32 flavors and on dev boards such as the NEB1DX Future Nebula Board, an IoT cloud ready board which allows users to quickly prototype and deploy their IoT ecosystems. Amongst the SDKs utilized were Cypress WICED Studio and Keil µVision.

Personal Projects   01/2018 – 04/2019

In the process of seeking jobs, and programming self-tasked **C++17** coding projects for a **Ubuntu 16.04 Linux target**, I have also been creating and coding applications for a bare-metal STMicroelectronics development board (**also C++17** but without exceptions and dynamic memory allocations). The applications and protocols exercised for the **STM32 NUCLEO F767ZI** board include:

- Coding for and synchronizing the onboard RTC's timestamps to a remote NTP server according to RFC 4330.

- Dispatching periodic events to asynchronuously monitor for Network Status changes and Disconnected events and reacting in some appropriate fashion.

- Performing asynchronuous HTTP POST and GET requests by means of non-blocking TCP sockets, stateful well-composed lambdas, and the ARM Mbed SIGIO interrupt notification method.

- Performing HTTPS requests (both blocking and non-blocking, i.e. synchronuous and asynchronuous) with appropriate CA level certificates to fetch predefined pages.

- Encapsulating MQTT Client functionality in a **C++17 Object-Oriented class** and using it to demonstrate MQTT communication with several differing IOT message types alongst with data serialization and deserialization via Nanopb.

- Encapsulating WebSocket Client functionality in a C++17 Object-Oriented class and using it to demonstrate periodic WebSocket random messaging with a hosted WebSocket server. Random WebSocket message streaming alongst with Control Frame exchanging are also simultaneously demonstrated with a second locally-hosted WebSocket server.

- Simultaneously mimicing IOT sensor acquisition and publishing via MQTT by generating and publishing (and receiving) several random dictionary well-composed sentences every few seconds forever.

- Invoking composed operations to communicate with Amazon **AWS IOT Cloud**, **Google IOT Cloud**, and **IBM Watson IOT Cloud** in sequence. Here, use is made of a well-structured templatized **C++17 MQTTS** class that I designed.

- And simultaneously with all these invoking primality-testing every few seconds by generating large random numbers and testing if they are prime by means of a well-composed algorithm. Essentially, the goal in this step is to employ some hardcore mathematics to spin the '**ARM 32-bit Cortex-M7** + DPFPU + Chrom-ART Accelerator' CPU a little even as it performs the afore-listed tasks also periodically.

Life Fitness, Brunswick Corporation, IL **2/2016 - 12/2017**
Embedded Software Engineer III / Contractor

- Utilized Boost.MetaStateMachine, Boost.ASIO, personally-crafted asynchronous timers, and other **C++11 techniques** such as **std::future** and **std::async** to create a GUI application representing a treadmill. Within the GUI's main context, care was taken to transform all asynchronous events into synchronous actions so as to sidestep **multi-threading** issues inherent in GUI development.
- Utilized **Boost.Fusion**, **Lambdas**, and general **Template Metaprogramming techniques** (including **SFINAE**) to create a well-structured type-driven protocol around a **SPI device** that was deployed to communicate to a third-party Display Board.
- Exercised profilers such as ***perf***, ***ftrace***, ***strace***, ***uftrace*** in order to detect and identify performance bottlenecks in the company's legacy C++ software implementations. Consequently resolved those performance bottlenecks by rewriting and improving upon those code segments, and, in a few cases, redesigning the required functionality.
- Wrote code to polymorphically encapsulate **MQTT infrastructure**. Extensively debugged said infrastructure to resolve legacy issues such as spurious disconnections, non-existent reconnection logic, and the proper handling of asynchronous callbacks.
- Debugged and revamped how a log4cxx deployment is configured and provisioned in order to support uniform application logging.

IHI Corporation, Energy Storage Department, IL **7/2015 - 10/2015**
Contractor - Senior Software Engineer

- Within two weeks of being hired, and on my own initiative, wrote a suite of applications to mimic the core of the company's software applications, and to illustrate several flaws in the latter's design. Techniques illustrated as part of these improvements include:
  → Working with (as opposed to against) the inherent asynchronicity of outward-facing protocols such as CAN, Modbus and ZMQ by employing event-based loops to better model the interaction between devices (eg. CANOpen message flows). Using event-based loops in such

a scenario streamlines the communications between each application in the ecosystem and the outside world. It thus, vastly improves application response times.
- → Separating different concerns into different threads to take advantage of concurrency. For example, ZMQ processing should occur in a separate thread of concern whereas CAN processing should occur within its own thread of concern.
- → Adopting an object-oriented approach to model device and OS Abstraction relationships, and ensuring all the newly created event-based artifacts also fit within this class-based polymorphic hierarchy. This ensues that going forward, a simple and elegant framework exists that can be extended, easily maintained and reused in other projects.

- Leveraged various techniques including the **Linux SocketCAN** implementation of CAN protocols and its userspace utilities and tools (**libsocketcan-dev** and **can-utils** packages) to successfully debug and resolve a CAN-based x86 Battery Rack Controller application. Prior to my efforts, this application had not been working in about a year.

- Leveraged Yocto, OpenEmbedded, bitbake, bash scripting, and other techniques to successfully debug and resolve a CAN-, Modbus-, and ZMQ-based distributed ARM Battery Rack Controller application. Prior to my efforts, this application had never worked. The application was eventually deployed on an **ARM Cortex A9** Dual 1 GHz **Freescale i.MX6** series ultra low power SoC.

- Created and documented procedures by which the company's engineers can bundle release software into deb packages, and deploy these along with a customized version of a bootable Ubuntu via Live Linux USB. This ensured that the same particular version of the Ubuntu OS/distro, its attendant set of programs and packages, its working settings and utilities, and a snapshot of the company's software could now be easily deployed out to customers in the field.

Shure Incorporated, IL **9/2013 – 6/2015**
Consultant, Embedded Software Group

- Participated in the development of a WiFi-enabled Conferencing and Discussion Access Point.
  - → Took primary ownership of the Conferencing Subsystem and was instrumental in the architecture and design of the WiFi, Network, BSP, and Software Download Subsystems.
  - → Invented a suite of event-based applications that leveraged **epoll** to listen on multiple file descriptors (such as that for sockets, **POSIX** message queues, **GPIOs** and **asynchronous timers**) and then reacted accordingly to achieve some desired system effect.
  - → Consequently leveraged the above exercise to create a standard framework encompassing the various event-based OS abstraction objects that can be reused across projects.

- Implemented an **OS Abstraction Layer** for the **C++11**, **POSIX**, and **OpenRTOS/FreeRTOS** platforms.
  - → This layer comprised objects such as **mutexes**, **semaphores**, **message queues**, **sockets**, **timers**, **Active Objects**, **Active Queues** and **Concurrent Queues**.
  - → As part of this effort, improved upon the formal interface specification itself and identified and resolved issues in the legacy **VxWorks** implementation.
  - → Also wrote custom CPPUnit platform-agnostic test scripts to demonstrate the conformance, functional, and stress characteristics of these OS abstraction objects.

- Other positive impact on the Product Development team.
  - → Took the initiative to instruct the team in key Linux Kernel device driver development principles, and also developed Linux specific artifacts such as a generic Shared Memory implementation to be used across projects.
  - → Took the initiative to implement a generic platform-agnostic caching algorithm to be used across teams and projects.

→ Ported a legacy **VxWorks** and **OpenRTOS** application to **Linux**. Demonstrated the success of the effort and its performance improvements by running a demo on a Ubuntu 13.10 Saucy Salamander virtual machine.
→ Documented legacy software and in the process, pointed out its flaws and shortcomings. Throughout the process, also exposed the team to useful tools such as SDEdit.

NEC Sphere Communications Inc., IL **12/2011 – 5/2013**
Consultant - Senior Software Engineer, Media Endpoint Group

- **HTTPS** module design, implementation and test for a SIP endpoint.
  → Conforming to RFCs 2818 and 2616. And utilizing **OpenSSL**.
  → Support for large file retrievals, session resumption, renegotiation and other advanced features.
  → Validated the implementation with a Microsoft IIS Testbench.

- Implemented an RTCP-XR VoIP metrics reporting module for a SIP endpoint.
  → Installed and setup a Ubuntu 12.04 LTS virtual machine to act as the Collector server.
  → Installed and configured OpenSIPs to serve as the Collector engine.
  → Wrote custom scripts to aid in VoIP metrics collection and display via mysql.

- Responsible for Port Hardening a SIP endpoint.
  → Familiar with Industry security tools such as **NMAP**, NESSUS, Metasploit, etc.
  → Activities involved exposing and identifying vulnerabilities, and then writing code in a systemic fashion to patch such vulnerabilities.

- Developed other SIP endpoint features such as system Call History and Boot Loading firmware.

- Participated in requirement analysis, testcase creation, testcase execution, and defect creation and tracking for a High End SIP phone.

Tellabs, Inc., IL **4/2010 – 12/2011**
Contractor, Network Elements (NE) Software

- Worked on new Embedded development for the Tellabs 7100 Optical Transport Network (OTN). The 7100 OTN deploys all service types (SONET/SDH, OTN, Ethernet, etc.) and comprises modules such as the SPMH, SPMC, UFABO, UFABC, FGSM, and various switching shelves.

- Took primary ownership for the Card Status Monitoring, Software Download, managed Optical Pluggable modules, and LED Control features.

- Designed, implemented, and tested the CXP Optical Pluggable module used to provide optical inter-connectivity between shelves.

- Designed, implemented and tested the Card Status Monitoring and Software Download features for the various modules within the 7100 OTN. This effort also involved implementations in the Alarm Management and Fault Reporting features.

- Designed, implemented and tested the LED Management and Control feature for the various modules within the 7100 OTN. This effort required interfacing with hardware engineers to ensure that the design aligned faultlessly with the pertinent hardware.

- Was instrumental in implementing changes needed to rebrand the 7100 OTN for Ericsson.

Motorola Inc., IL/Canada **6/2000 – 7/2009**
Senior Software Engineer, Commercial Government Industrial Services Sector (CGISS)

- **9 years of developing software in C/C++** primarily for APCO Base Stations, a **multi-threaded** trunked digital voice and data systems application embedded in a real-time platform. As a node in the Infrastructure side of an Astro IP network, its design required knowledge of network technologies such as IP, UDP, Ethernet, SNMP, NTP, and several other standard and proprietary protocols.
- Designed software features for **APCO Project 25 Internet Protocol-Based, Mission Critical Communications Systems and Radios**.

EDUCATION:

McMaster University – Hamilton, ON
Bachelor of Engineering (Cum Laude), Computer Engineering **2000**

- Performed research, under the supervision of Dr. Simon Haykin, on employing higher dimensional chaos for secure communications. Consequently wrote a dissertation with primary emphasis on the synchronization of chaotic systems.
- Designed and simulated a high-speed digital modem with variable data rates for a target radio link. Implemented the design in hardware using Altera VHDL.
- Developed Software for minimization of Boolean Functions with emphasis on the Object Modelling Technique (OMT). Also developed a hybrid method and **algorithm** for minimization of Boolean functions.
- Developed Software for a maze-tracing robot with emphasis on software documentation, testing, integration and system verifications.