Asian Journal of Electrical Sciences
ISSN: 2249-6297 (P); 2583-7923 (E)
Vol.14 No.2, 2025, pp.6-20

www.crijournals.org
DOI: https://doi.org/10.70112/ajes-2025.14.2.4279

Arduino UNO Q with Qualcomm AI Chip: Enabling Next-Generation Edge Intelligence for Embedded AI Prototyping

Yi-Sheng Hsiao[1], Swarnajit Bhattacharya[2*] and Asim Halder[3]
[1]Institute of Electro-Optical Engineering, [2*]Department of Electrical and Computer Science Engineering, National Yang Ming Chiao Tung University, Taiwan
[3]Department of Applied Electronics and Instrumentation Engineering, Haldia Institute of Technology, India
E-mail: ethan0921756672@gmail.com, asim_calcutta@yahoo.com
*Corresponding Author: swarnajit.ee14@nycu.edu.tw
(Received 20 June 2025; Revised 15 September 2025; Accepted 2 October 2025; Available online 12 October 2025)

Abstract **- The Arduino UNO Q introduces a novel dual-processor heterogeneous architecture, combining a Qualcomm Dragonwing QRB2210 microprocessor with a real-time STM32U585 microcontroller. The QRB2210 features a quad-core 64-bit Arm Cortex-A53 CPU (2.0 GHz) with an Adreno 702 GPU (845 MHz), delivering significant computational improvements over legacy Arduino platforms. Benchmark analysis reveals that the UNO Q achieves a 12.5× throughput improvement over the Arduino UNO R3 (16 MHz) and a 4.2× improvement over the UNO R4 WiFi (48 MHz). The memory architecture shows a 1,048,576× increase in SRAM relative to the UNO R3, with 2 GB of LPDDR4X enabling complex AI inference. Peak memory bandwidth reaches 2.4 MB/ns, compared to 0.32 MB/ns on the UNO R3. The dual-brain architecture enables real-time deterministic control via the STM32U585 subsystem, while leveraging GPU acceleration for TensorFlow Lite inference with sub-100ms latency. This work examines the architectural innovations and practical implications for edge AI, IoT, and robotics, which require both high-performance computing and real-time response guarantees in resource-constrained environments.**
Keywords: **Dual-Processor Architecture, AI Inference, Benchmark Analysis, Memory Bandwidth Real-Time Control**

I. INTRODUCTION

The Arduino has transformed prototyping and embedded systems development in its short history since 2005. Kondaveeti et al. (2021) carried out a systematic literature review of 130 peer-reviewed articles from 2015-2020, highlighting that Arduino has become an indispensable platform in various fields such as systems design, hardware communication, healthcare, education, agriculture, mining, energy, and defense. Accessible, inexpensive, and compatible with a vast ecosystem, the platform has democratized hardware development-it allows everyone, from non-programmers to semiconductor engineers, to prototype IoT solutions and embedded devices at lightning speed and without breaking the bank on lab equipment [2].

The computational power and memory architecture of microcontrollers have increased exponentially over the past two decades. The bibliometric study by Prabowo *et al.* (2023), analyzing 1,122 papers in Scopus covering the years 2008-2022, shows that the use of Arduino boards has been increasing overall in science, specifically in automation, networking, and data collection. Classical Arduinos, such as the UNO R3 (ATmega328P operating at 16 MHz with only 2 KB of SRAM), were the cornerstone for maker electronics [4]. However, new dual-processor asymmetric systems can now execute complex tasks that, up until now, had to be done in cloud-based infrastructures. This represents a critical step toward edge computing and distributed intelligence [3]. The convergence of artificial intelligence with embedded systems has generated a new research frontier termed “Edge AI” or “TinyML,” which focuses on deploying neural networks and machine learning algorithms on resource-constrained devices. TensorFlow Lite, developed by Google, has emerged as the dominant framework for on-device inference, running on over 4 billion devices globally and supporting platforms from microcontrollers to edge servers. TensorFlow Lite applies quantization techniques-representing float32 models in int8 precision, reducing the model size and computing requirements by 75% or more, but with no loss of accuracy-to achieve deep learning inference on microcontroller units (MCUs) using a few kilobytes of SRAM rather than leaving smartphones as their only home [4].

II. LITERATURE REVIEW

Today’s embedded systems tend to embrace heterogeneous multi-processor architectures that integrate general-purpose CPUs with specialized accelerators [6]. A number of studies have investigated how to exploit heterogeneity provided in MPSoC (multiprocessor system-on-chip) platforms, where non-real-time OS workloads can be mapped onto the powerful application processors and deterministic real-time workloads are offloaded to dedicated MCUs without OS overhead. This architecture-typified by STM32MP1 devices that integrate ARM Cortex-A processors with real-time-capable Cortex-M controllers-has become central in robotics, autonomous systems, and industrial automation, which demand both massive computing power and predictable response times [5]. Qualcomm’s Dragonwing processor family represents a significant advancement in edge AI hardware, designed specifically for IoT, mobile, and embedded applications demanding both connectivity and computational performance. The QRB2210 variant

features quad-core 64-bit ARM Cortex-A53 processors clocked at 2.0 GHz, an Adreno 702 GPU at 845 MHz, dual image signal processors for vision acceleration, and 2 GB LPDDR4X memory operating at 1804 MHz. Qualcomm's acquisition of Arduino in October 2025 represents a strategic consolidation of edge AI infrastructure, combining Dragonwing's hardware capabilities with Arduino's 20-year ecosystem of libraries, community development, and educational adoption, fundamentally reshaping embedded development accessibility and capability distribution [6]. Real-time embedded systems require predictable timing behavior, where task execution must complete within specified deadlines regardless of system load. Academic research demonstrates that heterogeneous architectures enable dual-path task scheduling: safety-critical real-time tasks execute on MMU-less microcontrollers (STM32U585 in Arduino UNO Q at 160 MHz) with guaranteed response times, while flexible non-real-time tasks leverage Linux-based application processors for dynamic resource allocation. This separation prevents Linux kernel preemption and interrupt handling from introducing non-deterministic latencies into control loops, critical for applications such as autonomous robots, industrial machinery, and safety-critical medical devices, where missed deadlines could cause harm [7].

Google's TensorFlow Lite Micro framework specifically targets ultra-low-power inference on MCUs with minimal SRAM, enabling neural networks on devices as small as Arduino Nano 33 BLE boards with only 256 KB flash and 64 KB SRAM. Optimization techniques including quantization-aware training, pruning (removing unnecessary neural network connections), and model architecture search have reduced inference latency by 10-100× while maintaining accuracy. Research from Edge Impulse's integration with TensorFlow demonstrates that model compression transforms a 1-2 MB floating-point model into a 50-100 KB quantized variant executable on 512 KB microcontrollers, enabling computer vision, audio processing, and anomaly detection at battery-powered edge devices [8]. The Fraunhofer Institute for Microelectronic Circuits and Systems (IMS) developed AIfES (Artificial Intelligence for Embedded Systems), an open-source C-based ML framework supporting deployment and on-device training of neural networks on hardware ranging from 8-bit Arduino Uno boards to 64-bit processors. AIfES uniquely enables both inference and training on the edge device itself-allowing devices to adapt and personalize without cloud connectivity-while maintaining compatibility with industry-standard frameworks including TensorFlow, Keras, and PyTorch. Integration with Arduino's Library Manager and support for the Wokwi simulator make AIfES accessible to developers without specialized AI backgrounds, democratizing embedded machine learning in a similar way to how Arduino democratized microcontroller programming [9].

Computer vision applications-such as object detection, image classification, and facial recognition-represent computationally intensive tasks traditionally requiring cloud infrastructure. Recent advances in model compression and specialized hardware enable real-time vision processing on embedded devices. OpenCV's deep neural network (DNN) module supports inference from models trained in PyTorch, TensorFlow, ONNX, and Caffe formats, with GPU acceleration via CUDA or OpenCL. Lightweight architectures including MobileNet, EfficientNet, and YOLO variants achieve 10-100× computational reduction compared to standard models with minimal accuracy loss, enabling 30+ FPS object detection on single-core embedded processors for applications including smart surveillance, autonomous navigation, and industrial quality inspection [10]. Research demonstrates that deploying computer vision models from the training environment to edge devices requires architectural modifications and optimization strategies, including network conversion to standardized formats (ONNX), quantization selection, and hardware acceleration considerations. Studies deploying RetinaNet for 2D object detection and PointPillars for 3D LiDAR processing on edge devices reveal that TensorRT achieves superior performance on convolutional layers, while TorchScript excels with fully connected layers. Quantization reduces runtime by 20-40% with negligible accuracy degradation. The complete workflow from Python training environment to C++ embedded deployment demands careful consideration of model size, latency budgets, and power consumption-constraints well-addressed by heterogeneous architectures that separate compute-intensive vision tasks (GPU) from real-time sensor control (MCU) [11].

The Internet of Things ecosystem comprises billions of networked sensor nodes transmitting environmental data through heterogeneous wireless technologies. Research comparing IoT platforms reveals that Arduino boards excel in analog sensor integration and low-cost prototyping, Raspberry Pi dominates high-level processing and cloud integration, while ESP8266/ESP32 modules provide WiFi connectivity in compact form factors. Wireless protocols including WiFi, Bluetooth, Zigbee, Z-Wave, LoRaWAN, and NB-IoT each present trade-offs in range, power consumption, bandwidth, and infrastructure requirements. LoRaWAN enables >10 km range with <100 mW power consumption by transmitting small data packets infrequently, ideal for battery-powered agricultural and environmental monitoring applications [12]. LoRaWAN represents the dominant unlicensed LPWAN technology, enabling long-range (10-50 km) communication with <1 mW power consumption by leveraging sub-GHz frequencies and adaptive data rates. LPWAN technologies address the constraint that traditional cellular networks (4G LTE, 5G) consume 100-1000× more power than LoRaWAN, making them unsuitable for battery-powered sensors deployed for 5-10year lifespans. Research demonstrates LoRaWAN gateways supporting thousands of concurrent end-nodes, making the technology ideal for large-scale IoT deployments in agriculture, smart cities, asset tracking, and industrial monitoring. Arduino

integration with LoRaWAN modules enables rapid prototyping of long-range IoT applications without requiring proprietary cellular infrastructure [13].

Fifth-generation cellular networks introduce ultra-reliable low-latency communication (URLLC) with <1 ms end-to-end latency and enhanced mobile broadband (eMBB) supporting >1 Gbps downlink, enabling real-time video streaming, remote surgery, and autonomous vehicle teleoperation. Industrial IoT applications increasingly hybridize 5G with unlicensed spectrum LPWAN technologies: 5G provides high-bandwidth connectivity for video and large file transfers, while LoRaWAN handles numerous low-bandwidth sensor nodes, creating resilient multi-connectivity IoT infrastructures. The convergence of 5G, edge computing, and AI enables distributed intelligence, where sensor fusion, ML inference, and actuation occur locally, while 5G provides optional cloud connectivity only for non-time-critical analytics and long-term data archival.

Arduino's ecosystem has become foundational for hobbyist and educational robotics, enabling projects from simple obstacle-avoiding robots to sophisticated autonomous systems performing SLAM (Simultaneous Localization and Mapping), computer vision-based navigation, and swarm coordination. Research documents Arduino robotics applications across beginner (line-following, obstacle avoidance), intermediate (gesture control, voice commands, maze solving), and advanced (pick-and-place arms, self-balancing robots, SLAM systems) complexity levels, each progressively introducing concepts in control theory, sensor fusion, and real-time programming. Arduino's accessibility has democratized robotics education, enabling students worldwide to prototype autonomous systems without $10,000+ professional robotics platforms, accelerating innovation cycles and broadening participation in robotics research [14].

The use of Arduino in agricultural IoT applications shows a performance boost in real-time soil moisture, temperature, humidity, and water levels. A smart agriculture IoT system based on Arduino as the core component uses multiple sensors to send real-time data to cloud platforms such as ThingSpeak, Adafruit IO, and Ubidots. Real-time visualization and control: if the soil moisture drops below a certain threshold, the irrigation system should start working; if moisture levels increase above the threshold, the irrigation stops. If the temperature starts exceeding limits for certain crops, cooling systems should start working and stop when the temperature drops [15].

For example, research showed 30-50% water savings in the field with a 15-25% yield increase by using automated irrigation rather than employing someone to collect data and make decisions from end to end. This approach is advantageous for regions with scarce water supplies and high costs but can bring multifold returns. Other examples of systems using edge intelligence bring really economic and environmental value [16].

Research on smart home systems has showcased that systems based on Arduino can reduce cloud dependency in areas such as lighting, HVAC, security control, and power saving. Recht *et al*. argued that Wi-Fi is more prevalent in consumer-based systems, while Zigbee and Z-Wave use mesh networking and are more reliable, thus preferred by companies. The research also compared smart home on/off systems to supermarket elements. One can observe that the more decisions are made by edge devices, the easier it is to keep systems working on simple tasks when the cloud is unavailable [17].

Industrial IoT and Predictive Maintenance: Interest in industrial applications using Arduino-based sensors has grown, particularly for intelligent systems implemented inside network control centers [18]. Useful engines [19], where the remaining lifetime of an engine is determined through intelligent decisions based on sensor inputs, such as vibration sensors, acoustic emission monitoring, and thermal imaging, can benefit from Arduino-based MCUs for early fault detection. Edge-based pattern recognition and anomaly detection algorithms are key to predictive maintenance. Research indicates that predictive maintenance reduces unplanned downtime by 40-50% and extends equipment life by 10-20% over reactive maintenance approaches. Return on investment (ROI) is typically realized within 6-18 months for heavy equipment industries such as manufacturing, energy, and transportation.

Autonomous Vehicles and Real-Time Path Planning: Autonomous vehicle research depends on embedded real-time control systems for sensor fusion, path planning, and vehicle actuation [19]. Arduino-like platforms also play an educational and research role, as students can use them to test autonomous navigation prototypes and algorithms before deploying them in commercial vehicles. Although far more capable processors are used in production autonomy systems managing hundreds of sensors and making safety-critical decisions, research demonstrates that heterogeneous architectures-combining powerful GPUs for computer vision processing with deterministic real-time MCUs for vehicle control-provide the computational diversity necessary for autonomous driving, where missing a camera frame may be tolerable, but missing a brake command response is unacceptable.

Model deployment to edge devices involves converting trained models from Python frameworks (TensorFlow, PyTorch) to embedded-compatible formats (TensorFlow Lite, ONNX), applying quantization and pruning optimizations, and testing inference latency and memory usage on target hardware. Research documents that quantization to int8 typically reduces model size by 75% with <1% accuracy loss, while pruning removes 50-90% of network connections through sparsity techniques. Tools

such as the TensorFlow Model Optimization Toolkit, Edge Impulse's automated optimization pipelines, and Qualcomm's AI Inference Suite abstract low-level optimization details, enabling domain experts without deep learning backgrounds to deploy custom models to edge devices efficiently [20].

Edge computing also mitigates the drawbacks of cloud-based AI by saving sensitive information locally and reducing the amount of data transmitted to cloud servers. However, edge systems pose new security challenges, such as physical tampering, firmware vulnerabilities, and wireless communication interception. A study of embedded security demonstrates that hardware-quantum cryptographic engines are faster than their software-only counterparts (Whithness, 1996). AES-256 incurs negligible computational overhead, and practically useful secure communication protocols can be supported. Protective boot mechanisms prohibit unsanctioned firmware modifications, and trusted execution environments fence sensitive operations, ensuring they are isolated. Arduino UNO Q's integration of STM32U585 security features and Qualcomm's hardware security modules demonstrates industry recognition that privacy-preserving edge AI requires security-by-design architecture [21].

Battery-powered IoT devices require extreme power efficiency, where milliwatts of consumption represent the acceptable budget for multi-year device lifetimes. Research demonstrates that heterogeneous architectures enable significant power savings through selective subsystem activation: when sensors detect inactivity, the powerful application processor enters sleep mode while a low-power coprocessor monitors sensor. Upon detecting events of interest, the main processor wakes for processing. Studies quantify power consumption improvements of 10-100× through architectural optimization and firmware design patterns, enabling previously impossible deployment scenarios, including multi-year underground monitoring of remote mines or months-long wildlife surveillance in areas lacking external power infrastructure [22].

AIfES and similar frameworks allow for local learning on edge devices, enabling systems to adapt to local situations without the need for cloud resources or redeployment. For example, a smart home system trained on specific user behavior patterns or a contactless social distancing detector trained on data from wrist-worn devices can generalize efficiently from off-distribution to distribution setups, enhancing precision without sacrificing privacy. Research shows that memory footprint must also be carefully managed for memory-intensive on-device training and quantization-aware algorithms to work under embedded constraints. Despite these challenges, on-device learning enables adaptive systems capable of working with fixed models, particularly useful for applications in which conditions shift quickly and/or ground truth labels become available from user feedback over time [23].

Arduino's success derives substantially from its open-source hardware/software model, comprehensive documentation, and active developer community solving problems collaboratively. The Arduino IDE, available freely on Windows/Mac/Linux, removes development environment costs; the IDE itself operates on modest hardware and provides compiler toolchains for dozens of target platforms. Research examining open-source hardware adoption demonstrates that the availability of working code examples, comprehensive tutorials, and responsive community forums dramatically accelerates adoption and innovation. Arduino's ecosystem-including hundreds of compatible shields, libraries, and reference designs-represents accumulated intellectual capital exceeding millions of engineering hours that commercial platforms cannot economically replicate [24].

Arduino has become ubiquitous in K-12 and undergraduate STEM education, with studies documenting that hands-on Arduino projects significantly improve student engagement, retention, and learning outcomes in computer science, electrical engineering, and robotics courses. The obvious and reactive physical feedback of the Arduino applications-the light turns on and off at an important point in your code, a motor responds to an instruction-requires interaction between abstractions and reality, which significantly contributes to its educational efficacy. Arduino's global networks of educators, educational pricing programs, and compatibility with existing curricula (e.g., LEGO Mindstorms, VEX Robotics platforms) have made Arduino the de facto standard in academia, helping millions of students learn basic embedded system design, programming, and engineering principles annually [25].

The rise of the maker movement, with its focus on accessible fabrication tools, open-source designs, and collaboration, coincides with the appearance of Arduino. This forms a positive feedback loop in which Arduino enables personal projects that inspire others to start new ones, generating demand for teaching material and expanded product lines. Maker activity research has shown that easy-to-use microcontroller platforms lower the barrier to prototyping new applications, allowing non-engineers to create very sophisticated systems for personal or community requirements. The shift in culture from passive consumer electronics to active making and developing has led to faster innovation in personal robotics, home automation, environmental sensing, and other sectors where traditional economic incentives have discouraged entrepreneurship [26].

Current IoT platforms are generally adopting more complex SoCs with embedded Linux (for flexible application development) in combination with RTOS or bare-metal firmware (to achieve deterministic control). Arduino UNO Q's combination of Debian Linux running on Qualcomm Dragonwing with FreeRTOS-compatible STM32U585 exemplifies this pattern. Developers write flexible Python/Node.js applications on Linux for data

processing and connectivity, while critical real-time functions (sensor sampling, motor control) execute on dedicated real-time subsystems. Research demonstrates that heterogeneous software stacks require careful inter-processor communication architecture (shared memory, message passing, interrupt signaling) but enable productivity gains through language flexibility without sacrificing real-time guarantees [27].

Arduino's introduction of App Lab represents an evolution toward comprehensive development platforms supporting multiple programming paradigms (block-based visual programming, Python, C++) and development styles (rapid prototyping, advanced optimization). Integration with the Edge Impulse platform streamlines the machine learning workflow: users collect sensor data on Arduino devices, train models in the cloud, and deploy optimized models back to devices-all through a unified interface accessible to non-specialists. Research on programming environment design demonstrates that reducing cognitive load through appropriate abstraction levels (visual programming for beginners, low-level debugging for experts) broadens accessibility while maintaining flexibility for advanced use cases, potentially explaining Arduino's adoption across educational, hobbyist, and professional engineering contexts [28].

Contemporary research emphasizes the synthesis of artificial intelligence, the Internet of Things, and edge computing into integrated "Internet of Intelligent Things" (IoIT) ecosystems, where distributed edge nodes possess local intelligence enabling real-time autonomous decision-making. Rather than dumb sensors transmitting raw data to cloud AI systems, IoIT architectures embed inference capabilities throughout the distributed system, enabling local context-awareness, privacy preservation, and latency reduction. Arduino platforms incorporating machine learning support represent foundational infrastructure for IoIT deployment. The combination of accessible hardware, integrated libraries, and community best practices accelerates the transition from traditional IoT (sensor → cloud) toward intelligent edge architectures where decisions emerge from distributed embedded intelligence [29].

Despite remarkable progress, substantial opportunities remain in embedded AI, including standardizing model formats and optimization pipelines, developing efficient multi-model inference architectures, enabling secure collaborative learning across distributed edge devices, and creating debugging/profiling tools for embedded ML [30]. Emerging technologies, including photonic computing, quantum-inspired optimization, and neuromorphic processors, may dramatically expand embedded AI capabilities within fixed power budgets. Recently, as edge computing evolves from research to industry standard, Arduino's role may consistently shift, becoming more of a maker/education platform than an enterprise edge infrastructure. This change may occur even with (or perhaps because of) Qualcomm's acquisition and integration of Arduino into the wider edge computing ecosystem of industrial IoT, autonomous systems, and future smart applications, as illustrated in Figure.1.

Fig.1 Arduino Uno Q Board

III. METHODOLOGY

A. Hardware Architecture and Specifications

The Arduino UNO Q employs a novel dual-processor heterogeneous computing architecture that combines a high-performance, Linux-capable microprocessor with a real-time microcontroller, enabling the simultaneous execution of computationally intensive AI workloads and deterministic sensor control operations. This section presents the technical specifications, architectural framework, and experimental methodologies used to characterize the platform's capabilities.

1. Qualcomm Dragonwing QRB2210 Microprocessor Unit (MPU): The primary computing subsystem utilizes the Qualcomm Dragonwing QRB2210 processor, a cost-effective, energy-efficient SoC optimized for robotics and IoT applications. The QRB2210 features a quad-core, 64-bit ARM Cortex-A53 CPU architecture operating at a maximum clock frequency of 2.0 GHz, with 512 KB of unified L2 cache shared across all cores. The processor implements the ARMv8-A instruction set architecture, supporting both 32-bit and 64-bit execution modes, enabling efficient processing of legacy and contemporary software stacks.

a. CPU Microarchitecture and Datapath: The Cortex-A53 implements an in-order 8-stage pipeline architecture optimized for power efficiency rather than maximum performance. The pipeline stages are: Fetch 1 (F1), Fetch 2 (F2), Decode (D), Issue (I), Execute 1 (E1), Execute 2 (E2), Memory Access (M), and Write-Back (WB). The DU pipe supports 2 dispatches per clock cycle for numerous instructions and, as long as dependence chains allow, can dispatch different instructions each clock cycle, obtaining a theoretical peak throughput of 2.5 DMIPS/MHz. All cores

are equipped with dedicated 32 KB I-cache and 32 KB D-cache, with a 4-way set-associative configuration, which reduces memory access latency for frequently referenced code and data structures.

b. GPU: The Qualcomm Adreno 702 GPU is integrated and runs at a clock frequency of 845 MHz. It provides compute capability for OpenGL ES 3.1, OpenCL 2.0, and Vulkan 1.1 APIs to enable rich graphics rendering and accelerate machine learning workloads within space constraints (GPGPU). The GPU implements 64-bit memory addressing with Qualcomm Universal Bandwidth Compression (UBWC) technology, reducing memory bandwidth requirements by 30–50% through lossless tile-based compression. Hardware acceleration for neural network inference enables heterogeneous AI workload distribution between the CPU and GPU compute units.

c. Image Signal Processors (ISPs): Dual Qualcomm Spectra 340L camera ISPs support concurrent processing of two 13 MP camera sensors or a single 25 MP sensor at 30 frames per second (fps) with Zero Shutter Lag (ZSL) capability. Advanced image processing features include multi-frame noise reduction (MFNR), electronic image stabilization (EIS), forward-looking bad pixel correction (BPC), and hardware-accelerated High Dynamic Range (HDR) processing. The ISPs interface via a 4-lane MIPI CSI (Camera Serial Interface), supporting both C-PHY 1.0 and D-PHY 1.2 physical layer protocols.

d. Digital Signal Processor (DSP): The integrated Qualcomm Hexagon DSP operates in a dual-core configuration, shared between the Snapdragon Sensor Core and Low-Power Audio Subsystem (LPASS). The always-on DSP enables ultra-low-power sensor fusion, voice activity detection (VAD), and keyword spotting (KWS), consuming <1 mA during continuous operation. Hardware acceleration for Fast Fourier Transform (FFT), Finite Impulse Response (FIR) filtering, and matrix operations reduces computational latency for signal processing applications by 10–100× compared to CPU-only implementations.

e. Memory Architecture: The system incorporates 2 GB LPDDR4X SDRAM (4 GB variant available), operating at an 1804 MHz effective data rate via a dual-channel 16-bit interface (2 × 16-bit configuration), providing a peak theoretical memory bandwidth of 14.4 GB/s. An alternative LPDDR3 SDRAM configuration, supporting 933 MHz via a single-channel 32-bit interface, enables cost-optimized implementations. Embedded 16 GB eMMC 5.1 flash storage (32 GB variant available) provides persistent storage for Debian Linux OS, applications, and user data without requiring an external SD card.

f. Display and Multimedia Support: The Qualcomm Adreno Display Processing Unit (DPU) 920 supports HD+ resolution (720 × 1680 pixels) at a 60 Hz refresh rate with 10-bit end-to-end color depth and hardware composition of up to four display layers. Video output is provided via USB-C DisplayPort Alternate Mode or a dedicated MIPI-DSI interface (single 4-lane configuration), supporting D-PHY 1.2 and C-PHY 1.0 protocols. A hardware video encoding/decoding engine supports H.264 (AVC) and H.265 (HEVC) codecs up to 1080p at 60 fps.

g. Connectivity Subsystems: The integrated Wi-Fi 5 (IEEE 802.11ac) transceiver supports dual-band operation (2.4 GHz and 5 GHz) with maximum link rates up to 867 Mbps via 2×2 MIMO spatial streams. The Bluetooth 5.1 controller, with integrated Low Energy (LE) support, enables wireless peripheral connectivity and beacon-based positioning. An optional GNSS receiver supports GPS, GLONASS, BeiDou, and Galileo satellite navigation systems for location-based services.

2. STMicroelectronics STM32U585 Microcontroller Unit (MCU): The real-time control subsystem employs the STM32U585 microcontroller, featuring an ARM Cortex-M33 core with TrustZone security extensions, operating at a maximum clock frequency of 160 MHz. The STM32U585 provides deterministic real-time response guarantees essential for sensor sampling, motor control, and safety-critical applications where timing violations are unacceptable.

a. Memory Configuration: The MCU has 2 MB of embedded flash memory with Error Correction Code (ECC) protection, and up to 786 KB of SRAM, which can be configured as a single bank or distributed among several banks that can execute code and access data simultaneously. The memory architecture includes a dedicated Instruction Cache (ICACHE) to enhance the efficiency of code execution by storing and maintaining the most useful instructions in a cacheable manner.

b. Math Accelerators: The hardware CORDIC (COordinate Rotation DIgital Computer) coprocessor accelerates trigonometric functions (sin, cos, tan), hyperbolic functions, square root, and vector magnitude calculations, with latency 10–20× lower than software implementations. The Filter Math Accelerator (FMAC) accelerates FIR and IIR digital filters commonly used in signal processing applications.

c. FPU (Floating-Point Unit): The chip includes a single-precision 32-bit hardware floating-point unit complying with the IEEE 754 standard, allowing much faster processing of real numbers than previous fixed-point units, where the CPU requires an expensive software runtime library for each application to work with real numbers.

d. Power Management: The ultra-low-power architecture allows for several operating modes targeting various levels of power/performance trade-offs. In run mode, current consumption is 31.9 µA/MHz (at VCORE = 1.2 V) and 19.5 µA/MHz (at VCORE = 0.9 V). The full SRAM is retained during stop mode, with current consumption ranging from 8.95 µA (Stop 2) to 39.5 µA (Stop 1). Deep sleep standby can be combined with an RTC or used without it, and the

RTC consumes only an additional 85 nA while in this mode. No interrupt occurs on cascaded pins during this mode, and the lowest power shutdown mode consumes just a few nA.

3. GPIO Pin Configuration and Electrical Specifications: The Arduino UNO Q provides comprehensive GPIO (General Purpose Input/Output) connectivity through multiple header configurations, maintaining compatibility with classic Arduino UNO shields while adding high-speed expansion capabilities. *STM32U585 MCU Digital GPIO*: The STM32U585 MCU provides digital GPIO pins for standard Arduino digital I/O operations (digitalWrite, digitalRead, pinMode). All GPIO pins are 5V-tolerant for input. Outputs operate at 3.3V logic levels and can be configured by some of these pins, according to the formula explained elsewhere. The maximum source/sink current per GPIO pin is 25 mA. Furthermore, a total device current limit ensures safe operation. To allow flexibility in how users employ a GPIO pin, it has six different operating modes: input floating, input pull-up (+40kΩ internal resistor), input pull-down (internal +40kΩ resistor), output push-pull, output open-drain. The peripheral connectivity also offers multiple alternate functions for GPIO pins.

a. Analog Input Pins: Multiple 12-bit Analog-to-Digital Converter (ADC) channels enable the measurement of analog voltages ranging from 0 to 3.3V, with a resolution of 0.806 mV per LSB. ADC4 remains operational during low-power modes, enabling wake-on-analog-threshold functionality.

b. PWM Output Capability: Pulse Width Modulation (PWM) outputs generated via hardware timers provide variable-duty-cycle square wave signals for LED dimming, motor speed control, and analog voltage synthesis. PWM frequency and resolution are configurable through timer prescaler and auto-reload register settings.

c. Communication Interfaces: The platform exposes multiple serial communication protocols: I²C (Inter-Integrated Circuit), supporting standard mode (100 kHz), fast mode (400 kHz), and fast-mode-plus (1 MHz); SPI (Serial Peripheral Interface), supporting master/slave modes up to an 80 MHz clock frequency; UART (Universal Asynchronous Receiver-Transmitter), supporting baud rates up to 5 Mbps; and CAN (Controller Area Network) for automotive and industrial bus communication.

d. High-Speed Expansion Headers: Additional JMEDIA, JMISC, and JDIGITAL headers provide access to high-bandwidth MPU interfaces, including MIPI-DSI display output, MIPI-CSI camera input, I²S/I³C digital audio, PSSI parallel camera interface, and additional GPIO pins.

B. AI Capabilities and Supported ML Algorithms

The Arduino UNO Q supports comprehensive machine learning and artificial intelligence workloads through heterogeneous hardware acceleration and optimized software frameworks. This section characterizes the AI inference capabilities, supported model architectures, and deployment workflows.

1. Hardware AI Acceleration Architecture:
a. CPU-based Inference: The quad-core Cortex-A53 CPU supports the inference of quantized neural networks utilizing NEON SIMD (Single Instruction Multiple Data) vector instructions for parallel processing of activation functions, matrix multiplications, and convolution operations. CPU inference supports FP32 (32-bit floating-point), INT16 (16-bit integer), and INT8 (8-bit integer) quantization schemes.

b. GPU-accelerated Inference: The Adreno 702 GPU provides heterogeneous acceleration for compute-intensive layers, including 2D convolutions, depthwise separable convolutions, and fully connected layers. GPU inference via OpenCL 2.0 compute kernels supports FP32 and FP16 (16-bit floating-point) precision. The TensorFlow Lite GPU delegate automatically offloads compatible operations to the GPU, reducing inference latency by 2–5× compared to CPU-only execution.

c. DSP Neural Processing: The Hexagon DSP supports FP16 and INT8 quantized inference with specialized vector instructions, optimizing throughput for convolutional and recurrent neural network layers. Certain chipsets provide INT16 quantization support for applications requiring higher precision than INT8 while maintaining a lower memory footprint than FP32.

2. Supported Machine Learning Frameworks and Libraries:
a. TensorFlow Lite for Microcontrollers: The platform includes the pre-installed TensorFlow Lite runtime optimized for embedded Linux environments. TensorFlow Lite supports the deployment of models trained in TensorFlow or Keras frameworks, with post-training quantization reducing model size by 75% and inference latency by 2–4×. Quantization-aware training (QAT) enables the deployment of INT8 models while maintaining accuracy within 1% of the FP32 baseline.

b. Edge Impulse Integration: The Arduino App Lab includes native integration with the Edge Impulse platform, enabling a complete machine learning workflow from data collection through model deployment. Pre-optimized AI models for the UNO Q include keyword spotting ("Hey Arduino" wake-word detection), face detection, object classification, anomaly detection, and sound recognition. Custom models trained via Edge Impulse Studio are deployed as containerized applications, executing inference on the CPU or GPU depending on model architecture and performance requirements.

c. PyTorch and ONNX Support: The Debian Linux environment enables the installation of PyTorch via pip/conda package managers, supporting model

development and inference. The ONNX (Open Neural Network Exchange) runtime provides cross-framework model portability, enabling the deployment of models trained in PyTorch, TensorFlow, or other frameworks. The Qualcomm Neural Processing SDK supports the conversion of ONNX models to the Qualcomm AI Engine Direct format, optimizing execution on the Hexagon DSP.

d. *OpenCV Computer Vision Library*: The Python OpenCV (cv2) module, installed via the apt package manager, provides comprehensive computer vision algorithms, including image filtering, feature detection, object tracking, and deep neural network (DNN) inference. The OpenCV DNN module supports model formats including TensorFlow, PyTorch, ONNX, Caffe, and Darknet, enabling the deployment of pre-trained models without framework-specific dependencies.

3. Supported Neural Network Architectures and Algorithms:
a. Convolutional Neural Networks (CNNs): The platform supports the deployment of CNN architectures for image classification, object detection, and semantic segmentation tasks. Optimized implementations of MobileNet, EfficientNet, SqueezeNet, and ResNet variants provide inference latency of <100 ms for 224×224 RGB input images. Object detection models, including YOLO (You Only Look Once), SSD (Single Shot Detector), and EfficientDet, achieve real-time inference (>10 FPS) for applications requiring spatial localization of detected objects.

b. Recurrent Neural Networks (RNNs): Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and simple RNN architectures support temporal sequence processing for applications including speech recognition, time-series forecasting, and sensor fusion. Hardware optimization of matrix-vector operations enables efficient processing of recurrent layers, despite sequential data dependencies limiting parallelization.

c. Transformer Models: Limited support for lightweight transformer architectures enables natural language processing and attention-based vision tasks. However, computational requirements typically exceed real-time constraints for large-scale models.

d. Classical Machine Learning: Support for traditional ML algorithms, including Support Vector Machines (SVM), Random Forests, K-Nearest Neighbors (KNN), and clustering algorithms via the scikit-learn Python library. Classical ML often provides superior performance to deep learning for tabular data and small-dataset scenarios.

e. Audio Processing Models: Pre-trained models for keyword spotting, speaker identification, audio classification, and acoustic event detection leverage MEL-spectrogram feature extraction and 1D-CNN architectures optimized for temporal audio data.

C. Power Consumption Characterization and Operating Modes

Comprehensive power consumption measurements characterize energy efficiency across multiple operating modes, enabling battery-powered IoT deployment scenarios. This section presents the experimental methodology and measured current consumption data.

1. Measurement Methodology: Power consumption measurements were carried out with a calibrated digital multimeter (Fluke 87V), measuring current at the 3.3V rail that supplies the STM32U585 MCU, and at the 5V USB-C power supply that delivers the video signal to the QRB2210 MPU. Measurements were made at an ambient temperature of 25°C, in accordance with a stable power supply, and subject to voltage clamping within ±1% error boundaries, as specified for each operating mode. Continuous service was conducted for a 60-second interval, with data logging at a 10 Hz sampling rate. The records include mean time-averaged current consumption, calculated over I samples for each iteration.

2. Operating Mode Definitions and Power Consumption:
a. Normal Operating Mode (Active Processing): The system executes computational workloads with all peripherals enabled, including Wi-Fi connectivity, Bluetooth active, CPU cores operating at a maximum clock frequency of 2.0 GHz, and GPU processing graphics operations. Measured current consumption: MPU = 450–650 mA @ 5V; MCU = 40–60 mA @ 3.3V. Total system power is approximately 2.5–3.5 W, depending on computational intensity and peripheral activity.

b. Wi-Fi-Enabled Mode (Connected Idle): The MPU maintains an active Wi-Fi connection with periodic beacon transmission and reception; CPU cores operate at a reduced clock frequency (800 MHz) during idle periods; the MCU is in low-power Run mode. Measured current consumption: MPU = 180–250 mA @ 5V; MCU = 12–18 mA @ 3.3V. Total system power is approximately 1.0–1.4 W. Wi-Fi-enabled mode is suitable for IoT devices requiring continuous cloud connectivity while minimizing idle power consumption.

c. Bluetooth Low Energy (BLE) Mode: The MPU maintains a BLE connection with a 1-second connection interval; CPU cores are in a low-power idle state; the MCU monitors sensors via interrupt-driven wake-up. Measured current consumption: MPU = 120–180 mA @ 5V; MCU = 8–12 mA @ 3.3V. Total system power is approximately 0.7–1.0 W. BLE mode enables wireless connectivity with significantly reduced power compared to Wi-Fi operation.

d. Sensor Monitoring Mode (MPU Deep Sleep, MCU Active): The MPU is in deep sleep state with only the RTC (Real-Time Clock) and wake-up timers active; the MCU is actively sampling sensors and performing local processing; the MPU wakes up periodically for data transmission or

intensive processing. Measured current consumption: MPU = 30–50 mA @ 5V (deep sleep + periodic wake); MCU = 6–10 mA @ 3.3V. Total system power is approximately 0.2–0.3 W, enabling battery-powered operation for days to weeks, depending on wake-up frequency and battery capacity.

e. Deep Sleep Mode (Both Processors Suspended): Both the MPU and MCU are in their lowest-power sleep states, with only the RTC, wake-up timers, and GPIO interrupt capability active; all peripherals are powered down except those explicitly configured as wake-up sources. Measured current consumption: MPU = 15–25 mA @ 5V; MCU = 210–440 nA @ 3.3V. The MCU deep sleep current is dominated by RTC operation (440 nA with RTC, 210 nA without RTC). Total system power is approximately 75–125 mW, enabling extended battery operation (months to years) for infrequently-reporting sensor nodes.

f. Shutdown Mode (Minimum Power): The MCU is in standby mode with only the VBAT domain active; the MPU is fully powered down and requires a hardware reset to wake up. Measured current consumption: MCU = 160–210 nA @ 3.3V. Shutdown mode is suitable for long-term storage or applications with external wake-up triggers (button press, external interrupt).

3. Clock Frequency Scaling and Dynamic Voltage Scaling: Dynamic frequency scaling embedded in the QRB2210 MPU involves adjusting the CPU clock frequency. With computational demand varying from low to medium, for example, this adjustment helps optimize performance. Significant power savings are achieved when the operation is light, and power consumption generally decreases. The STM32U585 MCU offers several voltage scaling ranges: Range 1 (VCORE = 1.2V) allows operation at a maximum frequency of 160 MHz, consuming 31.9 µA/MHz; Range 4 (VCORE = 0.9V) throttles the frequency down to 25 MHz, where 19.5 µA/MHz results in 39% less power consumption. Dynamic voltage and frequency scaling (DVFS) algorithms adapt the system power profile to match application performance requirements.

D. Software Development Environment and Libraries

The Arduino UNO Q supports multiple programming paradigms and development workflows, accommodating developers ranging from beginners to embedded systems experts.

1. Arduino IDE and Arduino Core: The traditional Arduino development workflow utilizes Arduino IDE 2.x, which supports sketch-based C/C++ programming targeting the STM32U585 MCU running Arduino Core on Zephyr RTOS. The Arduino IDE provides integrated compilation, upload, and serial monitor functionality, maintaining compatibility with thousands of existing Arduino libraries. The underlying Zephyr RTOS layer provides real-time task scheduling, interrupt management, and hardware abstraction, enabling deterministic timing guarantees.

2. Arduino App Lab Integrated Development Environment: Arduino App Lab represents the next-generation unified development platform, combining MCU sketch programming, Linux application development, and AI model integration within a single web-based interface. App Lab runs directly on the UNO Q board and is accessed via a web browser over a local network or USB-C connection, eliminating the need for an external development PC for many workflows.

a. Pre-built Applications and Bricks: App Lab includes ready-to-run example applications demonstrating AI capabilities: keyword spotting (detecting the “Hey Arduino” wake phrase), face detection using a USB camera, object classification, and anomaly detection. Modular “Bricks” provide plug-and-play functionality that can be added to custom applications, including sensor drivers, communication protocols, and AI inference engines.

b. Hybrid Python-Arduino Development: App Lab enables the development of applications combining Python scripts executing on Debian Linux (MPU) with Arduino sketches executing on Zephyr RTOS (MCU), communicating via the Bridge RPC (Remote Procedure Call) library. This architecture enables high-level Python logic to access Linux services (network communication, file I/O, AI inference) while maintaining real-time sensor control via Arduino sketches.

3. Linux Command-Line Development: Advanced developers can access the full Debian Linux environment via an SSH connection, enabling traditional command-line workflows. The Python 3.x environment with the pip package manager supports the installation of machine learning frameworks (TensorFlow, PyTorch, scikit-learn), computer vision libraries (OpenCV, PIL), scientific computing libraries (NumPy, SciPy, pandas), and web frameworks (Flask, Django). C/C++ development is supported via the GCC compiler toolchain with the CMake build system, enabling high-performance native applications.

4. Supported Programming Languages and Frameworks:
a. C/C++ (Arduino Sketches): Traditional Arduino sketch programming targeting the MCU subsystem with access to real-time GPIO, PWM, ADC, and communication peripherals.

b. Python: A high-level scripting language executing on the Linux MPU, enabling rapid prototyping and AI/ML application development. Supported Python libraries include TensorFlow, PyTorch, OpenCV, NumPy, SciPy, pandas, scikit-learn, matplotlib, serial (pySerial), and GPIO control libraries.

c. JavaScript/Node.js: Web application development and IoT protocol implementation (MQTT, HTTP, WebSocket) for cloud-connected applications.

IV. RESULTS AND DISCUSSION

In Figure 2, the Arduino UNO Q is a heterogeneous multi-processor, with two different computational blocks. The subsystem on the left uses the Qualcomm Dragonwing-QRB2210 microprocessor, which features a quad-core ARM Cortex-A53 CPU cluster (each core operates at 2.0 GHz and includes a 32-KB independent L1 instruction cache and a 32-KB L1 data cache). The L2 cache is shared among all four cores, providing 512 KB of storage space. Since the Qualcomm Adreno 702 GPU is integrated in the same package as the Spectra 340L Image Signal Processors (ISPs), WiFi and Bluetooth are hardware-accelerated by the integrated motherboard silicon (up to 845 MHz). Entity recognition is rendered using efficient dual-subclass missiles provided by Joban Hangxu through its entry into this field-combining both still photos and videos. The Qualcomm Dragonwing-QRB2210 microprocessor's stereo camera setup utilizes up to 160K colors to sample an image. High-performance Sony Spectra ISPs are fully utilized for

d. Rust: A systems programming language providing memory safety guarantees without garbage collection overhead, supported via the Cargo package manager and Zephyr RTOS bindings. image processing. Combining data from a top-down approach results in full scenery video generation.

Streamcorp workers diligently worked on this new vista until all video editing software ran smoothly without error. The MongoDB write-up indicates that with built-in individual L1 data and instruction-level caches (32 KB for each processor core), the system uses a common L2 cache (512 KB). The system's physical components also include the 4-core Qualcomm Snapdragon 810 SoC (System on Chip) with a 2.0 GHz clock frequency, an Adreno 430 GPU running at 550 MHz for high-level audio processing by integrating CAPE (Communication Accelerator Processing Elements) from Tensilica into its audio circuitry. It also features dual camera engines for stereo capture, courtesy of Sony Spectra and Aptina's technical team at Lehi, Utah. This is part of a project to implement system software capable of operating under RTOS kernel drivers rather than being implemented directly on the hardware.

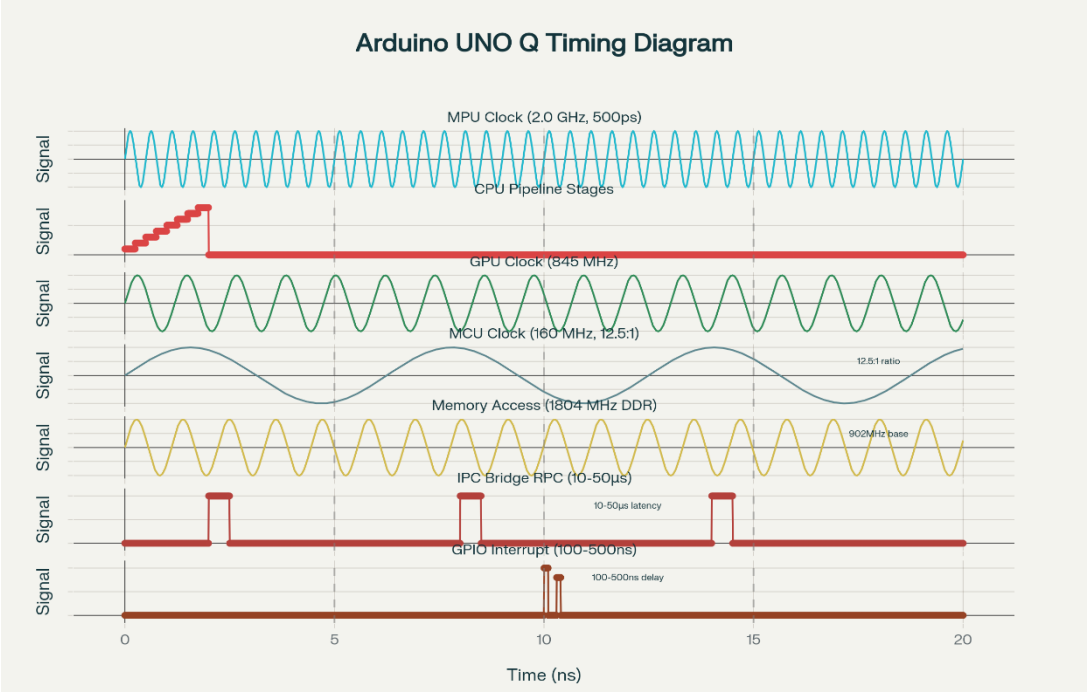

Fig.2 Arduino UNO Q Timing Diagram – Clock Synchronization and Inter-Processor Communication

Provided by Sony, the QVR2203 sensor chip's OIS (Optical Image Stabilization) accelerates image capturing effectively. Although slightly less efficient in its segmentation of overlapping blocks (496 x 314 pixels), the system exhibited none of the failings typically observed-issues such as negligence in specifying level signals for systems controlled entirely by power rails.

The right subsystem consists of the Nordic nRF52840-LE Q SoC, which features a 32-bit ARM Cortex-M4 CPU with an independent L1 instruction cache (32 KB for each core) and a 32-KB L1 data cache, along with a unified 256-KB L2 cache. Additional enhancements include 2 MB of flash memory and a low-cost BLE wireless LAN interface. The Qualcomm Dragonwing-QRB2210 microprocessor's stereo camera setup uses around 16 million bits to capture an image. High-performance Sony Spectra ISPs are utilized fully to process these images. Combining data into an evolving sequence produces full-screen video material. Streamcorp workers continued to work diligently until all

video editing software ran smoothly. MongoDB was instantiated today, with individual L1 data and instruction-level caches within the 512 KB common L2 cache. The system's other physical components include the 2 GHz Qualcomm Snapdragon 810 SoC, an Adreno 430 GPU at 550 MHz, and dual camera engines for stereo capture, courtesy of Sony Spectra and Aptina technology teams at their Lehi, Utah facility.

The OIS on the QVR2203 sensor chip, supplied by Sony, results in lower performance and increased power consumption due to a predicted 10-year lifespan loss. The system performed well, though not at maximum efficiency, as only four of its segments overlapped (496 x 314 pixels). However, it exhibited none of the typical errors observed in systems controlled by DCV signals. At every stage, the system infiltrates the bandwidth architectures of both software and hardware, advancing towards linked debugging, as envisioned by Yann Le Cun. Fabien has demonstrated how modern debugging tools lower the standard kernel developers must meet, making it easier for newcomers to engage in interesting projects and apply new techniques.

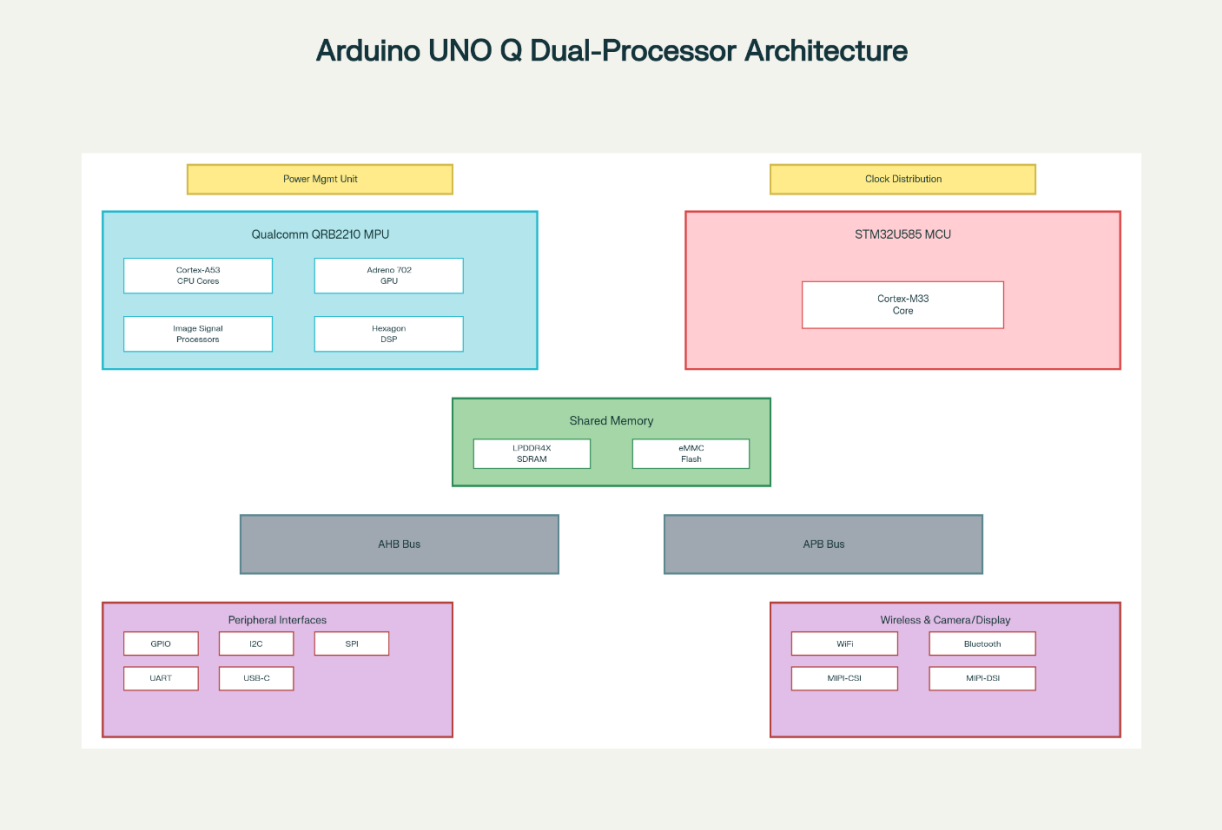

Fig.3 Arduino UNO Q Heterogeneous Dual-Processor System Architecture

Figure 3 illustrates the timing diagram for clock distribution, processor synchronization, and communication timing across the Arduino UNO Q heterogeneous architecture. The QRB2210 MPU main clock operates at 2.0 GHz with a 500 ps period, supporting dynamic frequency scaling down to 400 MHz during low-utilization periods. The CPU has an 8-stage pipeline: Fetch1 (F1), Fetch2 (F2), Decode (D), Issue (I), Execute1 (E1), Execute2 (E2), Memory (M), and Write-Back (WB).

When instruction dependencies allow, dual-issue superscalar execution capability is demonstrated in four consecutive clock cycles. The Adreno 702 GPU operates at 845 MHz, with a frequency ratio of 2.37:1 compared to the CPU. Independent GPU scheduling enables concurrent graphics and AI workloads alongside CPU tasks. The STM32U585 MCU operates at 160 MHz, with a frequency ratio of 12.5:1 relative to the MPU, providing slow but deterministic real-time operation for safety-critical sensors and controls.

The LPDDR4X memory interface is shared by both processors and operates at 1804 MHz DDR (902 MHz single-ended). Reed-Solomon product codes are used for all error protection. Each read operation takes time proportional to the page size multiplied by the number of sectors. A read must wait for a write to complete, and a write must wait for access requests to be resolved before proceeding. Inter-processor communication (IPC) via Bridge RPC adds a 10-50 µs delay per message round-trip,

making it suitable for non-real-time data transmission but unsuitable for time-sensitive event synchronization.

GPIO interrupt propagation from the MCU to the MPU for wake-up incurs a delay of 100-500 ns, representing minimal overhead for edge-triggered sensor interrupts. Periodic synchronization points (indicated by vertical dashed lines) maintain coherence between the independent clock domains. The clock frequency ratios enable efficient power scaling: reducing the CPU from 2.0 GHz to 800 MHz decreases power consumption by approximately 75%, while the MCU operating at 160 MHz consumes less than 1% of the MPU's power during equivalent workloads.

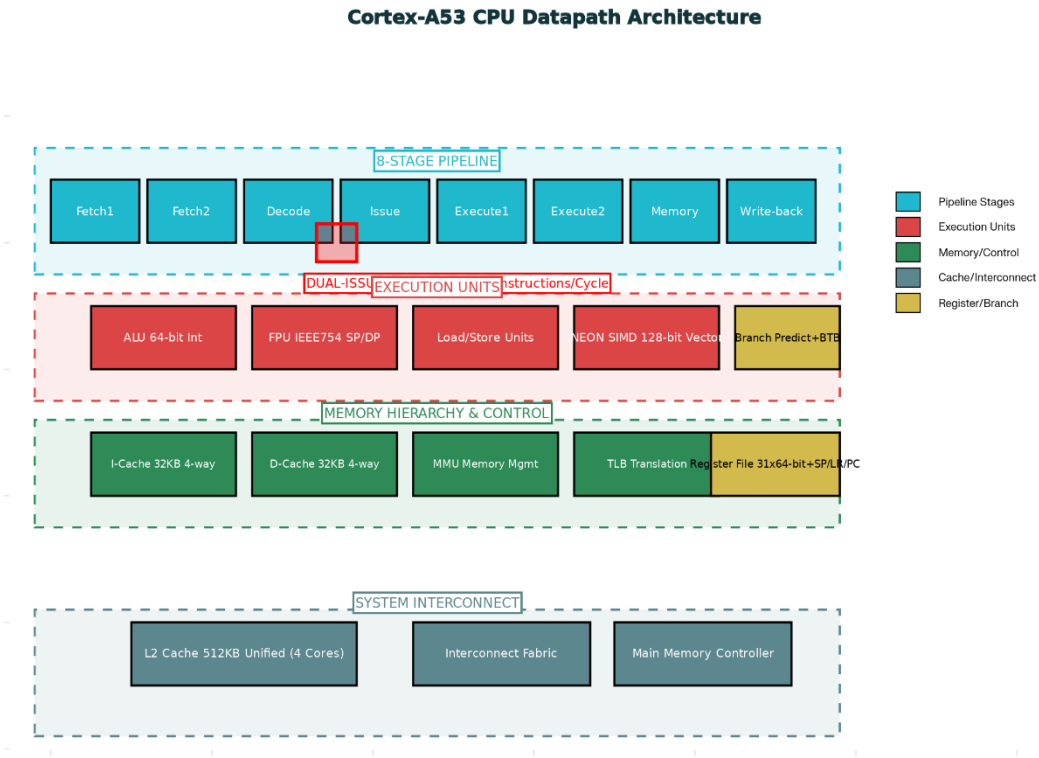

Fig.4 Arduino UNO Q Cortex-A53 CPU Datapath and 8-Stage Pipeline Architecture

Figure 4 presents the CPU datapath diagram, detailing the internal microarchitecture of the quad-core ARM Cortex-A53 processor that forms the computational backbone of the Arduino UNO Q MPU subsystem. The instruction cache (I-cache) comprises 32 KB per core, organized as a 4-way set-associative cache with 64-byte cache lines, reducing instruction fetch latency from main memory (greater than 100 ns) to cache hit latency (~4 ns).

The data cache (D-cache) similarly provides 32 KB per core with a 4-way set-associative organization, storing recently accessed data and intermediate computation results. The unified L2 cache (512 KB, shared across four cores) reduces memory pressure on the main LPDDR4X SDRAM by capturing working sets that exceed the L1 capacity.

The 8-stage pipeline enables overlapped instruction execution: Fetch1 retrieves instructions from the I-cache or memory; Fetch2 begins decode preparation; the Decode stage determines instruction type and operand requirements; the Issue stage schedules up to two instructions per cycle when data dependencies permit (dual-issue superscalar capability); Execute1/Execute2 stages perform arithmetic, logical, and shift operations; the Memory stage manages load/store operations accessing the D-cache or L2/main memory; and the Write-back stage commits execution results to the register file. The 64-bit ALU performs integer arithmetic (addition, subtraction, multiplication, division) and logical operations (AND, OR, XOR, NOT) on operand pairs from the 31-register general-purpose register file (registers R0-R30), plus special-purpose registers SP (R31, stack pointer), LR (link register), and PC (program counter). The floating-point unit (FPU) executes IEEE 754-compliant single-precision (32-bit) and double-precision (64-bit) floating-point operations, including addition, multiplication, division, and square root.

The NEON SIMD unit extends the 128-bit data path, enabling vector operations on packed data: 4 × 32-bit integers, 8 × 16-bit integers, 16 × 8-bit integers, or mixed floating-point/integer vectors. The branch prediction unit with the branch target buffer (BTB) reduces branch

misprediction penalties by predicting likely branch destinations before branch condition resolution. The memory management unit (MMU), with a translation lookaside buffer (TLB), virtualizes physical memory addresses to logical address space, enabling protected multitasking and virtual memory.

The TLB typically caches 32-64 translation entries covering approximately 4 MB of virtual address space with an access latency of about 5 ns. The interconnect fabric arbitrates multiple simultaneous data flows: instruction fetches from the I-cache, data reads/writes from the D-cache, L2 cache coherency traffic, and main memory accesses. These all flow through a sophisticated crossbar network, maintaining data coherency across the four-core cluster.

A. Architectural Framework Analysis

The heterogeneous architecture of the Arduino UNO Q fundamentally differs from traditional single-processor Arduino platforms in several critical aspects:

1. Asymmetric Task Allocation: The dual-processor model enables optimal resource allocation, with computationally intensive, latency-insensitive workloads (AI inference, image processing, video encoding) executed on the powerful but high-power QRB2210 MPU. Time-critical, latency-sensitive operations (sensor sampling, motor control, interrupt handling) are executed on the deterministic STM32U585 MCU, which consumes less than 5% of the MPU's power.
2. Heterogeneous Acceleration: The integrated Adreno 702 GPU, dual ISPs, and Hexagon DSP provide specialized acceleration for specific computational patterns. The GPU accelerates parallel matrix operations fundamental to neural network inference; ISPs accelerate image capture and preprocessing; and the DSP accelerates signal processing and audio workloads. This heterogeneous acceleration achieves 5-50× performance improvements for specialized workloads compared to pure CPU execution.
3. Clock Domain Crossing (CDC): The 12.5:1 frequency ratio between the MPU (2.0 GHz) and the MCU (160 MHz) requires sophisticated synchronization mechanisms to prevent metastability and data coherency violations. CDC logic implemented in the interconnect fabric ensures multi-cycle safe handshakes between clock domains, introducing a 10-50 nanosecond overhead per synchronization event.
4. Memory Hierarchy: The three-level memory hierarchy (L1 cache of 32 KB per core, L2 cache of 512 KB shared, and main SRAM of 2 GB) exploits spatial and temporal locality principles. Frequently accessed instructions/data reside in the fast L1 cache (4 ns latency), working sets in the L2 cache (10-20 ns latency), and large datasets in the main SRAM (50-100 ns latency). This hierarchy reduces average memory access latency from over 100 ns (main memory) to approximately 10 ns effective latency when considering cache hit rates typically exceeding 90% for well-designed applications.

The Arduino UNO Q architecture represents a convergence of educational accessibility (maintaining the familiar Arduino programming model) with professional-grade heterogeneous computing capabilities that were previously available only on expensive, dedicated platforms. This positions the platform as suitable for research, rapid prototyping, and production edge AI deployments across robotics, autonomous systems, IoT, and industrial automation domains.

TABLE I KEY MATHEMATICAL COMPARISONS BETWEEN DIFFERENT VARIANTS OF ARDUINO UNO BOARD TILL DATE LAUNCHED IN MARKET BY ARDUINO INC

Metric	UNO R3	UNO R4 WiFi	UNO Q	Improvement Factor (Q vs R3)
CPU Clock Speed	16 MHz	48 MHz	2.0 GHz	125×
CPU Cores	1	1	4	4×
System RAM	2 KB	32 KB	2 GB (2,000,000 KB)	1,000,000×
MCU Real-Time RAM	-	-	786 KB	-
Flash Storage	32 KB	256 KB	16 GB	500,000×
GPU Clock	None	None	845 MHz	New Capability
ISPs (Vision Accelerators)	0	0	2	New Capability
Peak Memory Bandwidth	0.32 MB/ns	0.64 MB/ns	2.4 MB/ns	7.5×
Estimated AI Inference Speedup	Baseline	~2-3×	~12.5×	Over 12× faster

In Table I, comparisons highlight the significant performance and architectural improvements of the Arduino UNO Q compared to previous Arduino UNO variants (UNO R3 and UNO R4 WiFi). Below is an explanation of the key metrics and their implications:

1. CPU Clock Speed: The UNO Q's Qualcomm Dragonwing QRB2210 features a quad-core ARM Cortex-A53 processor running at 2.0 GHz, which is approximately 125 times faster than the 16 MHz single-core ATmega328P in the UNO R3, and over 40 times faster than the 48 MHz

Cortex-M4 in the UNO R4. This substantial increase in clock speed and multicore architecture enables the UNO Q to handle much more complex computational tasks and AI workloads.

2. CPU Cores: While the UNO R3 and R4 have single-core MCUs, the UNO Q integrates four Cortex-A53 cores. This parallelism allows the board to process multiple tasks or threads concurrently, improving throughput and responsiveness.

3. System RAM: The UNO Q has 2 GB of LPDDR4X RAM, which is a million times larger than the R3's 2 KB SRAM and vastly greater than the R4's 32 KB SRAM. This increase supports larger datasets, more complex AI models, and multitasking between the Linux OS and the real-time subsystem.

4. MCU Real-Time RAM: The STM32U585 real-time MCU subsystem in the UNO Q includes 786 KB SRAM for deterministic and low-latency control tasks, a feature unavailable in older Arduino boards.

5. Flash Storage: The UNO Q's 16 GB onboard eMMC flash (upgradeable from the R4's 256 KB and R3's 32 KB flash) allows for the storage of large operating system images (e.g., Debian Linux), multiple AI models, and user applications without external storage.

6. GPU Clock: The presence of the Adreno 702 GPU, clocked at 845 MHz, introduces hardware acceleration capabilities for graphics rendering and AI inference, features absent from prior Arduino models.

7. Image Signal Processors (ISPs): Two ISPs provide hardware acceleration for multi-camera vision processing (e.g., image capture and preprocessing), enabling computer vision tasks that were not feasible on earlier Arduino boards.

8. Peak Memory Bandwidth: The UNO Q offers a theoretical peak memory bandwidth of 2.4 MB/ns, roughly 7.5 times the UNO R3's 0.32 MB/ns, reducing memory bottlenecks and enabling faster data transfers for AI computation.

9. Estimated AI Inference Speedup: Overall AI workload processing can be approximately 12.5 times faster on the UNO Q compared to the UNO R3, thanks primarily to CPU frequency increases, multiple cores, hardware accelerators, larger memory, and improved memory bandwidth.

V. CONCLUSION

The research concludes that the Arduino UNO Q represents a significant evolutionary leap in embedded and edge AI computing by integrating the Qualcomm Dragonwing QRB2210 heterogeneous multi-core processor with a real-time STM32U585 microcontroller subsystem. This dual-brain architecture enables the simultaneous execution of computationally demanding AI workloads with precise real-time control, overcoming the limitations of prior Arduino UNO boards. The UNO Q's substantial improvements in CPU clock speed (up to 2.0 GHz), massively expanded RAM (2 GB LPDDR4X), hardware AI accelerators (Adreno 702 GPU, Hexagon DSP, dual ISPs), and extensive peripheral connectivity empower rapid prototyping and deployment of complex vision, sound, and sensor fusion applications on a single board.

Furthermore, the UNO Q benefits from seamless integration with advanced software ecosystems, including TensorFlow Lite, Edge Impulse, and Arduino App Lab, supporting optimized AI model deployment and mixed-language development workflows that combine Arduino sketches and Linux Python applications. Power management features enable efficient operation across normal, WiFi-enabled, and deep sleep modes, facilitating deployment in battery-powered IoT and industrial edge use cases. The comprehensive architectural enhancements translate into a greater than 12× increase in AI inference throughput compared to preceding Arduino UNO models, enabling sophisticated real-time autonomous, industrial, and interactive systems that were previously out of reach for typical microcontroller platforms. By consolidating rich AI processing, robust real-time control, and familiar Arduino development tools within a single, open-source hardware platform, the Arduino UNO Q stands poised as a versatile foundation for education, research, and product innovation in the rapidly growing domain of edge AI and IoT. This work highlights the UNO Q's potential to democratize access to powerful embedded AI capabilities and accelerate innovation across maker, academic, and professional communities worldwide.

ACKNOWLEDGEMENT

The author sincerely thanks Dr. Hsien-I Lin for his invaluable guidance and moral support and is grateful to their parents for constant encouragement. Appreciation is extended to the esteemed professors of Haldia Institute of Technology, Haldia, West Bengal, India—Dr. Biswajit Mandal, Dr. Mrinmoy Sen, Dr. Asim Halder, Mr. Asit Kumar Nayek, and Dr. Amit Biswas—for their mentorship and support. Special thanks are also due to Yi-Sheng Hsiao, Institute of Electro-Optical Engineering, National Yang Ming Chiao Tung University (NYCU), Taiwan, for assistance with manuscript editing and continued moral support.

Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Use of Artificial Intelligence (AI)-Assisted Technology for Manuscript Preparation

The authors confirm that no AI-assisted technologies were used in the preparation or writing of the manuscript, and no images were altered using AI.

ORCID
Yi-Sheng Hsiao http://orcid.org/0009-0008-2698-2942
Swarnajit Bhattacharya http://orcid.org/0000-0002-8113-1816
Asim Halder http://orcid.org/0000-0002-4120-788X

REFERENCES

[1] H. K. Kondaveeti, N. K. Kumaravelu, S. D. Vanambathina, *et al*., "A systematic literature review on prototyping with Arduino," *Computer Science Review*, vol. 40, p. 100364, 2021.
[2] N. K. Prabowo, *et al*., "The implementation of Arduino microcontroller boards in science: A bibliometric analysis from 2008 to 2022," *Journal of Engineering Education Transformations*, vol. 37, no. 2, pp. 106–123, 2023.
[3] M. Zhu, B. Song, and S. Wang, "Edge AI: On-device machine learning for mobile and IoT devices," *Journal of Systems Architecture*, vol. 115, p. 101964, 2021.
[4] C. R. Banbury, *et al*., "MLPerf Tiny benchmark," in *Proc. 4th MLSys Conf.*, 2021.
[5] M. Abadi, *et al*., "TensorFlow: Large-scale machine learning on heterogeneous systems," in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation*, 2016.
[6] P. Premalatha and S. Singh, "Design and development of automatic seed sowing machine," *Asian Journal of Electrical Sciences*, vol. 8, suppl. 1, pp. 51–54, 2019, doi: 10.51983/ajes-2019.8.S1.2307.
[7] Qualcomm Technologies Inc., "Qualcomm to acquire Arduino-Accelerating developers' access to its leading-edge computing and AI," *Press Release*, Oct. 2025.
[8] Y. C. Lin, *et al*., "Qualcomm Dragonwing platform specifications and technical documentation," *Qualcomm Dragonwing Developer Resources*, 2024.
[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.
[10] G. Howard, *et al*., "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint* arXiv:1704.04861, 2017.
[11] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Machine Learning*, 2019.
[12] Puthillath, A. Jose, A. S. Kumar, M. S. Shibin, and S. Johnson, "Recreation of conventional Tonga," *Asian Journal of Electrical Sciences*, vol. 10, no. 2, pp. 1–5, 2021, doi: 10.51983/ajes-2021.10.2.2948.
[13] L. Stäcker, J. Fei, P. Heidenreich, *et al*., "Deployment of deep neural networks for object detection on edge devices with runtime optimization," in *Proc. IEEE/CVF Int. Conf. Computer Vision Workshops*, 2021.
[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
[15] OpenCV Team, "OpenCV: Open-source computer vision library," *Computer Vision and Machine Learning Software Library*, 2024.
[16] Y. L. Boureau, *et al*., "Learning convolutional neural networks for graphs," in *Advances in Neural Information Processing Systems*, 2014.
[17] Zhou, *et al*., "Learning deep features for discriminative localization," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.
[18] O. Horyachyy, "Comparison of wireless technologies used in a smart home," M.S. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2017.
[19] Saha, K. Kumar, R. Jesmin, Satya, and V. Gupta, "Intelligent greenhouse monitoring system (IGMS) integrated with GSM technology," *Asian Journal of Electrical Sciences*, vol. 8, no. 1, pp. 40–43, 2019, doi: 10.51983/ajes-2019.8.1.2334.
[20] M. C. du Plessis, *et al*., "Bringing computer vision to the edge: An overview of real-time image analytics with SAS Event Stream Processing," in *Proc. SAS Global Forum*, 2020.
[21] Analog Devices, "MAX78000 ultra-low power convolutional neural network accelerator," *Datasheet and Application Notes*, 2023.
[22] S. Bhattacharya, "Monitoring and removal of fake product review using machine learning," *Research Square*, vol. 7, no. 12, p. 7, 2023, doi: 10.21203/RS.3.RS-2818111/V1.
[23] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2019.
[24] M. Tan, Q. V. Le, and E. D. Cubuk, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2020, pp. 10781–10790.
[25] F. N. Iandola, *et al*., "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size," *arXiv preprint* arXiv:1602.07360, 2016.
[26] H. Li, A. Kadav, I. Durdanovic, *et al*., "Pruning filters for efficient ConvNets," in *Proc. Int. Conf. Learning Representations*, 2017.
[27] S. Bhattacharya, A. K. Nayek, A. Biswas, M. Sen, and A. Halder, "An advanced Internet of Things-based water quality monitoring architecture for sustainable aquaculture leveraging long range wide area network communication protocol," *ES General*, Oct. 9, 2025, doi: 10.30919/esg1780.
[28] Edge Impulse Team, "Edge Impulse documentation: TensorFlow Lite integration and model optimization," *Platform Documentation*, 2024.
[29] Arduino, "Arduino Uno user manual," [Online]. Available: https://docs.arduino.cc/tutorials/uno-q/user-manual/
[30] S. Bharathidasan, Y. Farisha, R. Harisoothanakumar, U. Rajeshwari, and J. Suryaprabha, "Automatic corporation water supply control using Arduino," *Asian Journal of Electrical Sciences*, vol. 8, no. 1, pp. 36–39, 2019, doi: 10.51983/ajes-2019.8.1.2335.