An Ultra-low-power Embedded AI Fire Detection and Crowd Counting System for Indoor Areas

ALEXIOS PAPAIOANNOU, CHARALAMPOS S. KOUZINOPOULOS, DIMOSTHENIS IOANNIDIS, and DIMITRIOS TZOVARAS, Information Technologies Institute (ITI), Greece

Fire incidents in residential and industrial areas are often the cause of human casualties and property damage. Although there are existing systems that detect fire and monitor the presence of people in indoor areas, research on their implementation in embedded platforms is limited. This article introduces an ultra-low-power embedded system for fire detection and crowd counting using efficient deep learning methods. For the prediction of fire occurrences, environmental and gas sensor along with multilayer perceptron nodes are used. For crowd counting, a custom lightweight version of YOLOv5 is introduced, using an architecture based on ShuffleNetV2, resulting in a model with low memory requirements, high accuracy predictions, and fast inference on an embedded platform. The accuracy, power consumption, and memory requirements of the proposed system are evaluated using public datasets and datasets acquired by the environmental and image sensors, and its performance is compared to that of existing approaches.

CCS Concepts: • **Hardware**; • **Computing methodologies** → *Neural networks*; • **Computer systems organization** → *Real-time system architecture*;

Additional Key Words and Phrases: Fire detection, crowd counting, embedded system, ultra-low-power system, machine learning for low-power systems, computer vision, Multilayer perceptron (MLP), YOLOv5, LW-YOLOv5

ACM Reference format:
Alexios Papaioannou, Charalampos S. Kouzinopoulos, Dimosthenis Ioannidis, and Dimitrios Tzovaras. 2023. An Ultra-low-power Embedded AI Fire Detection and Crowd Counting System for Indoor Areas. *ACM Trans. Embedd. Comput. Syst.* 22, 4, Article 61 (July 2023), 20 pages.
https://doi.org/10.1145/3582433

This research was carried out as part of the project ≪AE3vAO≫ (Project code: KMP6-0082241) under the framework of the Action ≪Investment Plans of Innovation≫ of the Operational Program ≪Central Macedonia 2014 2020≫, which is co-funded by the European Regional Development Fund and Greece

Authors' address: A. Papaioannou, C. S. Kouzinopoulos, D. Ioannidis, and D. Tzovaras, Information Technologies Institute (ITI), 6th km Charilaou-Thermi Rd, Thessaloniki, Greece; emails: alexiopa@iti.gr, kouzinopoulos@iti.gr, djoannid@iti.gr, Dimitrios.Tzovaras@iti.gr.

1539-9087/2023/07-ART61 $15.00
https://doi.org/10.1145/3582433

1 INTRODUCTION

Every year, several fires occur indoors that cause extensive property damage as well as significant injuries and loss of human life. The European Fire Safety Alliance estimates that over 5,000 people die from residential fires in Europe per year, while the number of people injured is approximately an order of magnitude higher [3]. In surveillance systems, fire monitoring is a significant feature that can be used to detect fire in early stages to avoid its spread. Furthermore, the utilization of crowd counting can provide assistance in the evacuation process by providing to the authorities the total number of people trapped in an indoor space during a fire incident.

Currently, traditional fire detection and crowd counting systems utilize one or more sensors, usually without any memory or power constraints, as the systems that execute the models have enough resources and are equipped with a continuous power supply. Existing fire detection systems due to the need for continuous power supply or high power requirements cannot operate in case of a power outage, which is very common during a fire [10, 50]. Moreover, crowd counting technologies that use commercial **Photoelectric Infrared (PIR)** motion sensors or Infrared beam counters and thermal sensors need continuous power supply and cannot provide the exact number of people accurately [13, 39]. Additionally, **machine learning (ML)** algorithms used for fire detection and crowd counting are generally complex with increased memory requirements. As a result, such systems are not feasible to be implemented on a low-power embedded system that is constrained in terms of resources.

To fill this research gap, this article proposes a low-power embedded system for indoor fire detection and crowd counting during fire incidences using efficient deep learning methods. The embedded system consists of a fire detection sub-system that is based on measurements from environmental and gas sensors and a crowd counting sub-system that is based on an image sensor. The system can inform the fire department and send a notification when a fire incident occurs along with the total number of people occupying an indoor area to manage the rescue of trapped people more effectively. With the use of a battery, or optionally a photovoltaic energy harvester, the system is energy-autonomous and can operate even in the case of a power outage. Moreover, the crowd counting sub-system periodically estimates the total number of people to be available in case of a chaotic situation where it would be difficult for the system to accurately detect the total number of people. Finally, the low-power consumption characterizes the system as environmentally sustainable, reducing energy waste and having minimal negative impacts on the environment.

The method presented for fire detection and the proposed modifications of the original YOLOv5 algorithm, along with its low power consumption and constrained memory footprint, make it an innovative solution for integration into an embedded system.

The major scientific contributions of this article are as follows:

- An ultra-low-power system for fire detection and crowd counting that can operate autonomously for up to two months without any charging;
- An improved version of an existing fire detection system validated using a public and a custom dataset;
- A proposed lightweight modification of the YOLOv5 algorithm for embedded systems based on ShuffleNetV2 architecture;
- An evaluation of the proposed modifications of YOLOv5, based on the original algorithm;
- A thorough sub-system-wise breakup of the energy consumption.

The rest of the article is structured as follows: Section 2 details previous research on fire detection and crowd counting systems. Section 3 presents the overall architecture of the proposed system, details regarding the implemented embedded platform, and the configurations of the MCU. Section 4 provides information for the two sub-systems along with the proposed methods.

In Section 5, experimental results are given on energy consumption and memory requirements, and the accuracy of the proposed methods are compared with existing approaches. Finally, Section 6 presents the conclusions of this research and discusses future work.

2 RELATED WORK

In the past few years, a growing number of papers have been published on fire detection and crowd counting systems, however, research on their implementation on embedded systems is limited. This section details related work on such systems.

2.1 Fire Detection Systems

In the literature, two different approaches for fire detection have been proposed. The first was based on environmental sensing, including temperature, humidity, smoke, and gas sensors, with the main goal being the detection of fire incidents at an early stage without constraints in terms of memory or power consumption. The second was based on image processing, combining image sensing and machine learning algorithms. A system following this approach can detect a fire from direct flames, but usually only after it has spread significantly.

Sowah et al. designed and implemented a multisensor fire detection system based on **Fuzzy Logic (FZ)** and a **Convolutional Neural Network (CNN)** [37]. Multiple fire signatures, such as flames, smoke, and heat, as well as images from surveillance cameras, were used. The system was implemented and tested on a Beaglebone microprocessor, a computing platform with an increased performance compared to the embedded platform used in this article, with an accuracy of 94% for the CNN algorithm and 90% for FZ.

An FZ system, based on an Arduino Uno, was also designed in Reference [36] to identify the existence of fire. Data was collected from flame, temperature, and smoke sensors. The accuracy of the implemented system was 95.83% with an execution time of 5 seconds.

Ouanid et al. developed a fire detection system based on the sensing of smoke density, temperature, and carbon monoxide [52]. The algorithm included an **Artificial Neural Network (ANN)** with a single hidden layer and four neurons and was implemented on an ATmega16 microcontroller. The execution time of the implementation was 30 seconds, and the lowest reported mean squared error was approximately 5.39×10^{-10}.

In Reference [35], a Wireless Sensor Network was designed and implemented using multiple sensors for indoor fire detection. The system was based on a Raspberry PI. Smoke, gas, and temperature sensors were used for the implementation. The system generated an alarm when two or more sensors' values exceeded predefined threshold values. The energy consumption of the deployed sensors was computed, with a minimum of 0.5 mWh and a maximum of 60 mWh reported.

In Reference [38], a low-power sensor node for early detection and monitoring of fire was presented. The authors developed a detection algorithm based on the **Dempster-Shafer theory (DS)**. The system was evaluated on an MSP430 microcontroller using a temperature and humidity sensor. The same algorithm, combined with a threshold method, was utilized in Reference [9]. The system was implemented on a low-power ATmega1281 processor using data collected from temperature, humidity, and light sensors.

A system based on an ATmega microprocessor was presented in Reference [47]. The system utilized a temperature and gas sensor, and an alarm is transmitted via a GSM module in case of fire detection. The detection method proposed by the authors included if-then rules that utilize data from the temperature and gas sensor to detect the occurrence of fire.

A low-power wireless smoke alarm system for home fires is presented in Reference [24]. The system utilizes a combination of smoke, temperature, and CO sensors. Additionally, it includes a mobile application to notify the users in case of fire. According to the authors, the device

consumes 36 Ah, providing a battery life of up to five years. The accuracy of the system was not reported.

2.2 Crowd Counting Systems

Various technologies have been used for people counting applications in the literature as well as in commercial systems, including: infrared beams or PIR systems [6, 49], thermal sensors [14, 46], Wi-Fi trackers [19, 48], and image sensors [23, 32].

Infrared beam counters consist of receiver and transmitter units installed side-by-side at the entrances. A count occurs whenever an obstruction to the transmission signal appears. Infrared sensors have the ability to operate in a light-free environment and are able to identify and count people moving in both directions using multiple beam sensors [2]. However, the disadvantages of these systems are their inability to handle multiple people passing simultaneously and the restricted installation specifications, as the system has to be installed beside a narrow door with the accuracy decreasing as the width of the door increases [32]. Additionally, it is unable to identify relatively stationary occupants, the detection range of each sensor is limited, and it is unable to distinguish humans and other moving objects [53].

Thermal sensors are typically installed at gates or entrances to detect body heat [2]. The accuracy of detecting the heat emitted from people may be impacted by the presence of heat sources and external weather conditions. Crowded areas can also reduce accuracy, as thermal counters cannot distinguish between objects due to their different temperature signatures. The main advantages of thermal sensors are that they are unaffected by changes in the illumination and do not require a background subtraction algorithm, increasing the processing time [45]. However, the high cost of acquisition and their limited detection range are significant drawbacks [48].

Wi-Fi trackers are used to locate and count people in a room. They can work as long as the Wi-Fi is enabled and there is access to it. Users must be connected to a Wi-Fi access point to be counted. This can result in inaccurate results, since this sensor type depends on having a Wi-Fi access point where people from different rooms may be connected [48]. Additionally, if a user has multiple devices connected to the same Wi-Fi access point, then the sensor's count values may be inflated [27].

Another type of common commercial device for crowd counting and occupancy detection is image sensors [53]. Accurate occupancy data can be obtained by analyzing the frames that an image sensor has captured. Complex image processing algorithms are used for crowd counting and usually consist of three distinct steps: background subtraction, motion tracking, and occupant recognition [11]. Vision-based sensing systems have the ability to recognize different objects occupying a room, including people and animals, and can achieve highly accurate results [48]. Image sensors often require the utilization of capable systems that can efficiently execute image-processing algorithms as well as the presence of an ambient light source.

Crowd counting implementations that utilize image sensors can be generally classified into two major categories: treating crowd counting as an object detection problem [7, 14] and using the crowd density estimation based on features and regression analysis [1, 44]. Object detection-based methods can be applied by training detectors to locate individual people in images. Thus, certain features are extracted and are subsequently used to train a binary or multi-class classifier. However, regression-based models estimate the crowd count using extracted image features and applying machine learning techniques to perform a regression between the image features and the crowd size.

Conti et al. proposed two different low-power approaches based on CNNs to estimate the occupancy of classrooms [7]. The first CNN consisted of eight layers and was used to classify and count heads. The second method was based on the estimation of crowd density with a

CNN regression model. The system was implemented on a Samsung Exynos 5410, with a **root mean square error (RMSE)** of 6.42 people and an energy cost of 3.97J per image. Despite the low RMSE, this work is not appropriate for a low-power embedded system due to its increased computational requirements.

Gomez et al. designed and developed a low-power crowd counting algorithm using a thermal image sensor and CNN [14]. The CNN consisted of three convolution layers with one pooling layer and the softmax function on the last layer before the output. The system was implemented and tested on a Cortex M4 platform with an error of ±1 person in 84.4% of the test set. The execution time of the algorithm was approximately 2.3 minutes with a power consumption of 0.48 *mAh*.

A lightweight method for crowd counting in indoor rooms, based on motion and size criteria using a histogram, was presented in Reference [44]. The system was evaluated on iMote2k, a low-power platform with high-performance specifications, including an Intel XScale processor. The dataset for the evaluation contained images with one, two, and three people with an accuracy of 89.5%, 82.48%, and 79.8%, respectively.

In Reference [1], a histogram of gradients and the **Support Vector Machine (SVM)** algorithm were used for crowd counting. The system was implemented on an FPGA platform and low-resolution grayscale 320×280 images were used for its evaluation. The authors compared the performance of the algorithm using an FPGA platform and a PC with an i5 processor and concluded that their method was almost five times faster when executed in FPGA compared to a PC processor.

2.3 Fire Detection and Crowd Counting System

There is only limited research on approaches that combine fire detection and crowd counting methods in low-power systems. A vision-based, embedded fire detection system was presented in Reference [5] to detect the occurrence of fire and count the number of people in a building. The authors used a Raspberry Pi and a Kinect sensor to implement two different deep neural network models, MobileNet SSD and ResNet101. The distance of the vision node from people and the light intensity inside a room were evaluated, with a mean accuracy of 87.5% and 75%, respectively.

As can be seen from the analysis of the state-of-the-art research, as presented in the sections above, existing literature works have increased requirements in terms of power consumption and execution time, utilize detection methods with a higher complexity, and generally have to be implemented on platforms with demanding computational specifications, making them challenging to be utilized efficiently in low-power embedded platforms.

3 SYSTEM ARCHITECTURE

This article describes an indoor, ultra-low-power fire detection and crowd counting system using two deep learning models, based on the MLP and YOLO algorithms, respectively. For the fire detection sub-system, a preliminary report with initial findings was published in Reference [31]. A vision-based model is used for the crowd counting sub-system. More details on the two subsystems are provided in Sections 4.1 and 4.2, respectively. In the case of a fire occurence indoors, the proposed system transmits wirelessly a notification of the event as well as a report on the number of people located inside.

Sections 3.1 and 3.2 analyze the hardware components of the embedded platform and the configuration of the MCU. Additionally, Section 3.3 illustrates the methodology of the proposed system for fire detection and crowd counting.

3.1 Embedded Platform

The proposed system is implemented on a low-power embedded platform. It uses the STM32L496VGT6P ARM Cortex-M4 ultra-low-power MCU [42] from ST Microelectronics

for processing. The MCU has a frequency of up to 80 MHz and can operate in the range between $1.71V$ and $3.6V$, with up to $1Mb$ of Flash memory and 320 KB of SRAM.

For temperature and humidity sensing, the BME680 gas sensor from Bosch is used. BME680 is a digital environmental sensor with a voltage range between $1.71V$ and $3.6V$ and a current consumption of $2.1\mu A$. CCS811 is a multi-pixel digital ultra-low-power air quality and gas tool used to monitor eCO_2 and TVOC. The main supply voltage range is between $1.62V$ and $3.6V$, with a maximum current consumption of $30mA$. Additionally, the proposed embedded platform includes an ultra-low-power CMOS image sensor, HM01B0. The image sensor supports a 320×320 pixel resolution and a 320×240 windowed mode, with a power consumption of less than 2 mW.

The embedded platform is designed with an ultra-low-power approach: It is capable of entering the lowest power mode, where most components are switched off to conserve energy and to allow for the elimination of leakage sources. That way, it is possible to totally cut off the microcontroller and most sensors, leaving only the key monitoring elements active. In the lowest power mode, the RV-3028-C7 **Real-Time Clock (RTC)** module is utilized to keep time and trigger system restart regularly.

3.2 MCU Configuration

STM32L496VGT6P includes two different instructions that can be used to switch the power mode into a low-power state: **Wait-For-Event (WFE)** and **Wait-For-Interrupt (WFI)**. WFE uses the value of the event bit from the System Control Block to wake up the MCU while in the WFI, the MCU remains in sleep mode until an interrupt occurs. The implementation is based on hardware interrupts, since they are supported by the sensors used. They allow the MCU to enter low-power modes between processing cycles and thus to minimize the total power consumption.

For the active and inactive/sleep period of the MCU, there are eight different low-power modes. Reference [41] presents a comparison between the modes in terms of wake-up source, enabled components, and power consumption for an $1.8V$ input voltage.

In run mode, the implemented platform supports dynamic voltage scaling to optimize power consumption. The CPU, as well as the Flash and SRAM memory, are enabled at a reduced frequency of 2 MHz. In sleep and low-power sleep mode, the CPU is stopped while the peripherals remain active. An interrupt or event can wake up the CPU. The stop 2 mode is used, where most of the Vcore domain is put into a lower leakage mode to achieve the lowest power consumption while retaining the content of both the SRAM and the registers.

When reading data from Flash, latency can be introduced by the system in the form of wait states, depending on the frequency of the CPU clock and the internal voltage range of the device. Also, the main regulator output voltage (Vcore) supports two modes that affect the total number of wait states and the total power consumption. Range 1, or high-performance range, supports a clock frequency of up to 80 MHz with a minimum Flash access time for reading. Range 2, or low-power range, has a maximum clock frequency of 26 MHz and a longer reading time from the Flash memory than Range 1. The correspondence between wait states and CPU clock frequency is presented in Reference [41].

The Flash memory interface of the MCU includes a $256B$ data cache memory with 8 cache lines of 4×64 bits each. When data is requested by the CPU, frequently used data lines can be stored in the cache to accelerate code execution by enabling a data cache enable bit in the Flash access control register. Moreover, the use of the cache memory for accessing the Flash memory has no effect on the performance of the proposed implementation when there are no wait states. However, according to Reference [40], the cache should lower the power consumption by up to 20%, since accesses to the cache require significantly less current compared to accesses to the Flash memory.

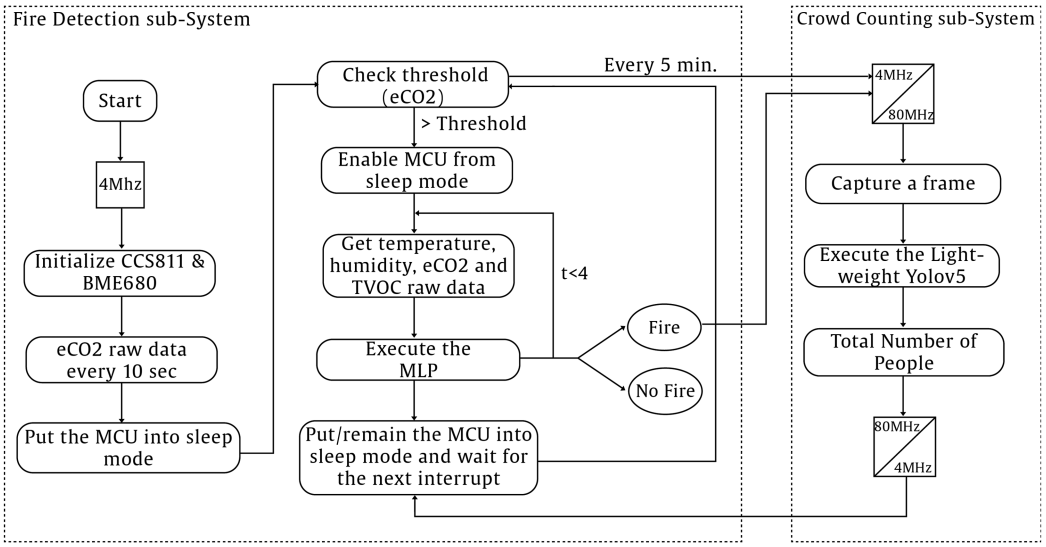

Fig. 1. Flow diagram of the methodology for fire detection and crowd counting.

STM32L496VGT6P can be supplied by an external Switched Mode Power Supply, bypassing the internal regulator of the MCU and extending that way the power efficiency in run modes. More specifically, a step-down DC-DC converter is used in the proposed embedded platform. The step-down converter deactivates the integrated low-dropout regulator by generating a voltage for the core that exceeds the internal voltage by 50 mV or more.

3.3 Methodology

Figure 1 presents a flow diagram of the proposed system for fire detection and crowd counting. The procedure begins with the **microcontroller unit (MCU)** and the two environmental sensors being initialized and the static RAM being restored. The operating frequency of the MCU is 4 MHz. After initialization, the gas sensor acquires **equivalent calculated carbon dioxide (eCO_2)** concentration measurements every 10 seconds, while the MCU enters sleep mode with its interrupts enabled. In sleep mode, the main regulator that supplies the core of the MCU is disabled and the Flash memory is powered down while retaining the content of both SRAM and registers.

The system remains in sleep mode with the lowest power consumption until the concentration of eCO_2 exceeds a set threshold. In that case, an interrupt from CCS811 wakes up the MCU, while input data for the temperature, humidity, eCO_2, and **total volatile organic compounds (TVOC)** sensors become available.

Since different gases are emitted during a fire (e.g., eCO_2, **carbon monoxide (CO)**, **hydrogen cyanide (HCN)**, O_2, H_2), a specific type of gas could be used as first indicator for reliable indoor fire detection systems [12, 20]. The eCO_2 levels were selected as a first indicator in case of fire, since it is the primary gas emitted from complete combustion in smoldering fires, based on most recent research [29]. Smoldering is a slow, flameless form of combustion that is started by heat sources such as cigarettes, coal, or wires, and it is usually the first stage of a residential or office fire. It can cause slow combustion of household items such as furniture, linens, and paper-based materials that are found in a home or office space and it is difficult to be quickly detected by photoelectric smoke detectors and fire alarms [29].

However, higher values in eCO_2 than the set threshold could be an indication of poor air ventilation in the room and may lead to false alarms. Based on References [29] and [31], measurements

of TVOC level can enhance the accuracy of a fire detection system and limit the false alarm produced by the eCO_2 measurements. Additionally, based on Reference [15], high values of TVOC are observed in case of fire with the gases influencing the TVOC value not including the eCO_2.

During the execution of the fire detection process, interrupts are disabled to prevent the program from being interrupted unexpectedly. When the execution of the fire detection process is completed, the frequency of the MCU scales up to 80 MHz to execute the crowd counting method.

Clock sources can be changed safely on the fly in run mode through certain configuration registers. Initially, the system uses the multi-speed internal RC oscillator, which is trimmable by software and is able to generate frequencies from 100 kHz up to 48 MHz. To utilize the maximum clock speed of 80 MHz, a phase-locked loop of the system is required. The transition between the two clock sources is performed by adjusting the flash latency and configuring the prescalers and the multipliers of the clock tree accordingly.

After increasing the operating frequency, the MCU is ready to capture an image frame from the area via the image sensor and subsequently to execute the proposed lightweight version of the YOLOv5 algorithm. The frequency remains at the highest level until the crowd counting process finishes. At that point, it is scaled back to 4 MHz. When a prediction of the total number of people is available, the MCU returns to sleep mode, while the interrupts are enabled back.

The MCU wakes up periodically to execute the crowd counting sub-system on 5-minute intervals and also when receiving a fire alert. The image sensor requires approximately 12 $msec$ to capture a frame, a short enough time to acquire a clear frame before everyone starts to panic and run to the exit. However, a chaotic situation can occur before the detection process has started; in that case, images with significant motion blur can be potentially captured and thus the crowd counting subsystem will not be able to detect the total number of people with high enough accuracy. In that case, the system will report the total number of people occupying the room as retained from the previous execution of the model.

4 PROPOSED SYSTEM AND METHODS

4.1 Fire Monitoring and Detection Sub-system

A low-complexity and high-efficiency model based on the MLP neural network was implemented for fire detection. The model is called "Chain of MLP models" and was introduced in Reference [31]. It consists of five inputs: temperature, humidity, eCO_2, TVOC, as well as the output from the previous execution of the MLP model. Different numbers of hidden layers and neurons have been evaluated for the implementation, as presented in Section 5.2. In the hidden layers, the rectified linear unit was used as an activation function, as it had the best convergence performance. For the output layer, the sigmoid function was used, as it was a supervised classification problem with only two values: zero for no fire and one for fire.

Thus, the chain of MLP models uses as a feature the output from the previous state together with the environmental sensors' data from the current state. The process is described by the following equation:

$$OutMLP_t = MLP(OutMLP_{t-1}, T_t, H_t, (CO_2)_t, TVOC_t), t \geq 1, \quad (1)$$

where T is the temperature, H the humidity, and t the number of the previous states or timesteps.

The length of the MLP chain is proportional to t and is defined by the timesteps. Figure 2 illustrates the **partial autocorrelation function (PACF)** plot for the final decision of the MLP model for the first 15 lags. The PACF is a description of the relationship between an observation in a time series and observations at previous timesteps, with any intervening observations being eliminated. As observed, until lag 3 there is highly significant with the highest values being observed on lag 1.

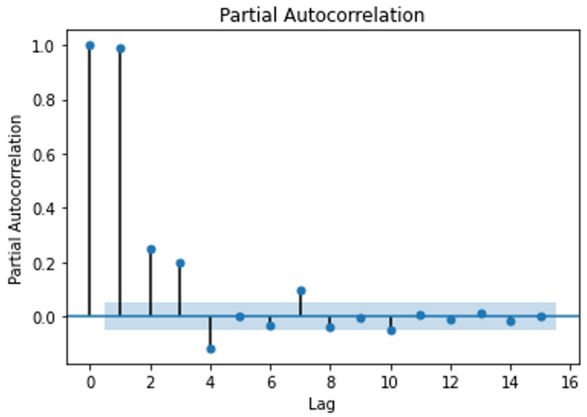

Fig. 2. PACF plot with 15 lags.

The relationship between accuracy and the number of timesteps is detailed in Reference [31]. Four timesteps have been selected for the proposed system, since in that case the highest change of value was achieved in terms of accuracy. Furthermore, based on the PACF, the previous three values appear to have the biggest impact on the final decision. The model becomes slightly more accurate, as additional MLP models are used, but with a significantly longer execution time.

Furthermore, using the prediction of the previous state as an input to the next state could reduce the overall performance of the model, as a prediction error is entered. Nonetheless, false predictions from the previous state in combination with data from environmental sensors of the current state do not appear to affect the overall performance of the model.

4.2 Crowd Counting Sub-system

For the crowd counting sub-system, a custom lightweight version of the original YOLOv5 object detection algorithm was developed, aiming for real-time results using a low-power embedded platform.

4.2.1 YOLO. The YOLO algorithm is commonly used in object detection problems [33]. It is utilized as an alternative to popular two-stage approaches, and it handles object detection as a regression problem rather than a classification problem by using a single neural network. YOLOv5 utilizes convolutional neural networks as a backbone, with the Darknet architecture being replaced with PyTorch [21]. It uses PANet as neck and multi-scale features, which consist of several bottom-up and top-down layers. The main differences between YOLOv5 and previous versions are the significantly smaller model size, faster inference, as well as better accuracy.

4.2.2 YOLOv5 Modifications. This section summarizes the modifications applied to the original version of YOLOv5. The aim of the modifications was to reduce the total memory footprint of the algorithm while maintaining its accuracy.

- The CSPDarknet53 network was replaced by lightweight architectures based on ShuffleNetV2, as detailed in Section 4.2.3.
- The total number of detection layers was reduced to minimize the total size of the output vector, as detailed in Section 4.2.4.
- The total size of PANet was reduced, while the activation function was replaced from **Sigmoid Linear Unit (SiLU)** to **Rectified Linear Unit (ReLU)**, as detailed in Section 4.2.5.

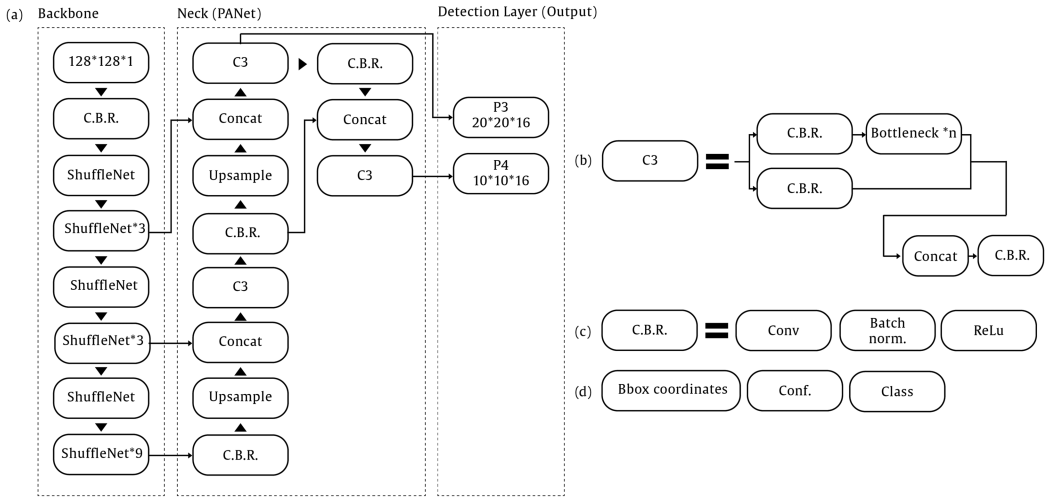

Fig. 3. The proposed LW-YOLOv5 network architecture.

- Grayscale images were used to train the custom YOLOv5, reducing the total input size from three channels to a single one and subsequently the algorithm's overall size. More details are given in Section 5.1.2.

4.2.3 ShuffleNetV2. ShuffleNet is an efficient convolutional neural network architecture that focuses on a lightweight architecture for embedded devices [51]. It utilizes pointwise group convolution layers and channel shuffle operations to reduce computation costs while maintaining accuracy [25]. In pointwise convolution, a 1×1 kernel is used, which means that the kernel iterates through every single point [16].

ShuffleNetV2 is an improvement over V1 in terms of execution time and computational complexity [26]. The most significant differences between the two algorithms are: the replacement of group convolutions with a new channel split operation, which splits the channels into two groups, the replacement of the group convolution layers with convolution layers, the replacement of the addition operation with concatenation, since addition is relatively expensive operation with a significant number of memory accesses. The channel shuffle remains the same in the second version of the algorithm, but it has been moved to the end of the architecture.

4.2.4 Anchor Boxes and Detection Layer. YOLO can work very well on multiple-object detection problems even if the objects are contained in the same grid using anchor boxes. Anchor boxes are a set of predefined boxes with a specific height and width that are used to capture the scale and aspect ratio of objects to be detected. The k-means clustering algorithm is used to define the best-fit anchor boxes. In the proposed implementation of this article, 3 anchor boxes were used based on the number selected by the authors of the original YOLOv5.

Detection layers are used to perform dense predictions consisting of a vector containing the predicted bounding box coordinates (center, height, width) along with the confidence value and the probability classes. Three detection layers were included in the initial model presented by the authors of YOLOv5. To reduce the total memory requirements, two different variations with one and two detection layers were used in the proposed implementation.

4.2.5 Network Architecture. In Figure 3(a), the proposed **lightweight YOLOv5 (LW-YOLOv5)** network architecture is presented. The main parts are the Backbone, Neck, and Output layer, as

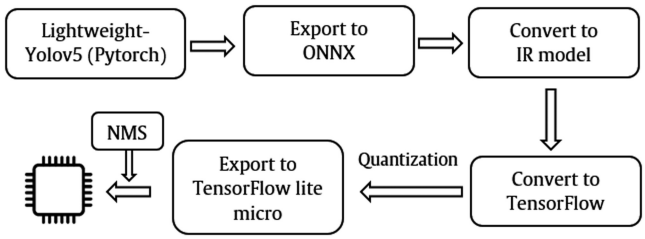

Fig. 4. Converting a PyTorch model to TFLM.

in the original algorithm. Additionally, the algorithm includes the C.B.R. block, which consists of a Convolution layer, Batch normalization layer, and a ReLu function. This block can be found in the Backbone as well as in the Neck layer. In Backbone, it is used to replace the original Focus layer in YOLOv5, reducing the execution time and the total size in RAM, while in Neck it is used before the final output. Finally, the SiLU function of the original YOLOV5 algorithm is replaced with ReLU, as SiLU is not quantization-friendly and not well-supported in embedded devices.

In Figure 3(c), a C3 block is presented. It is inspired by DenseNet [17] but instead of combining the entire input and output after CNN layers, the input is split into two parts. The first part is passed through a C.B.R. block as well as a number of Bottleneck blocks, while the second only through a C.B.R. block. Then, the two outputs from the two parts are concatenated, followed by another C.B.R. block.

As mentioned in Section 4.2.4, the total number of the detection layers was two and one in the proposed LW-YOLOv5. The size for each detection layer is displayed in Figure 3(d). Additionally, the output dimensions for each detection layer were $20 \times 20 \times 16$ for the first (P1) and $10 \times 10 \times 16$ for the second (P2).

4.3 PyTorch Model Conversion

TensorFlow Lite Micro (TFLM) is an open-source machine learning inference framework that can be used to run deep learning models on embedded platforms [8]. The proposed model was developed in the PyTorch framework, similar to the original YOLOv5, and was subsequently converted to a TFLM model before loaded to the embedded platform.

Figure 4 presents the conversion process from PyTorch to TFLM. Initially, the PyTorch model was exported in the Open Neural Network Exchange format. hen, the exported model was converted to **Intermediate representation (IR)** using **Open Visual Inference and Neural network Optimization (OpenVINO)**. Finally, the exported OpenVINO IR model was converted to TensorFlow using a script from Reference [18], a quantization method was applied, and then it converted to TFLM. The script converts the exported IR model to TensorFlow representation without converting the NMS block, as there are no operations in TFLM for this block. Thus, a custom version of the NMS algorithm was developed in C without any machine-dependent optimizations or special instructions.

4.4 Quantization

Quantization reduces the number of bits used to store the values of weight and activation functions of ANNs, converting floating-point values to fixed-point integers. The operations between weights are faster but at the expense of the model's accuracy. Thus, there is a tradeoff between the number of bits that represent weights and the accuracy of the model, and some information may be lost [34].

There are two different methods for neural network quantization, **Post-Training Quantization (PTQ)** and **Quantization-Aware Training (QAT)** [28]. The main difference between PTA and QAT is the stage where the scale is computed. In PTQ, the quantized model is computed after the network has been trained and typically restricted to FP16 or INT8 quantization. In QAT, the quantized model is computed during the training phase, keeping significantly more accuracy in the results than PTQ [22]. QAT methods have been tested on classification models using 4 bits per neural network weight and 8 bits per activation [4]. However, low-bit quantization is much more difficult to apply to object detection problems and achieve accurate results.

QAT was applied in this article, in the TensorFlow file. Currently, TFLM supports computation on both floating-point in binary32 format and fixed-point on 8-bit integers [30]. Thus, 8-bit was selected for quantization, as it was the lowest supported value. However, experimentation with different quantization sizes can also be performed. Finally, symmetric quantizers were used for the weights and asymmetric for the activation function of the model.

5 EXPERIMENTAL RESULTS

This section includes the different datasets used during the training and evaluation phases as well as the model performance analysis of the two proposed models and their overall energy requirements.

5.1 Datasets and Evaluation Metrics

5.1.1 Fire Detection. Two different datasets were used to evaluate the performance of the proposed algorithm. The first dataset was created by the authors of this article. It contains measurements using two environmental sensors (eCO_2, TVOC, temperature, humidity) at regular 10-second intervals. The experiments were conducted in a naturally ventilated room, with dimensions of $10m^2$, a temperature of approximately $23^{\circ}C$, and a relative humidity of approximately 32%. Different materials were used, which can be easily found in residential homes and office spaces, including paper stacks, cardboard boxes, plastic wraps, wood from office desks, and electrical cables. In each experiment, an electric heating device was used to start the fire. Approximately 8,500 data points were collected in total, for temperature, humidity, eCO_2, and TVOC and were divided into training, validation, and test sets with a ratio of 70%, 10%, and 20%, respectively, within each event class. The dataset was balanced as it contained equal instances from the fire and no fire classes.

The second dataset from Reference [29] contains data from eCO_2, TVOC, temperature, and humidity sensors, using as burning materials paperboard cardboards (paper with a plastic layer), clothing (cotton fabric, denim fabric, a mixture of cotton and polyester), commercial bed sheet and pillow sets. Two methods were used by the authors to start fire in each experiment; charcoals and electric heating devices. A re-sample of the data was applied to a 10-sec interval to fit in the presented method.

5.1.2 Crowd Counting. A set of images has been collected and annotated from two different working areas at the **Centre for Research and Technology Hellas (CERTH)**. In the first room, containing between 0 to 4 people, the camera covered approximately $10m^2$. The second room was part of the CERTH/ITI nZEB Smart Home, a real domestic building that is used to implement and evaluate smart IoT-based technologies in different areas including energy, health, big data, robotics, and artificial intelligence. The total covered area was approximately $15m^2$. In both datasets, there were images that represent different actions of the employees (not wearing masks, wearing masks, covering heads) and different lighting conditions (low-light conditions, normal conditions).

The input data from the image sensor had 320×240 resolution and was subsequently cropped to 128×128, as Yolo is more efficient with square images. In total, 3,680 gray-scale images were

(a) Working area (b) nZEB Smart Home

Fig. 5. Images from two different indoor rooms.

used for training, 1,051 for testing, and 525 for evaluation. The total number of people in each set was 11,040, 2,102, and 1,050, respectively. In Figure 5, two examples from the different rooms are illustrated.

5.1.3 Evaluation Metrics. To evaluate the performance of the different fire detection models and existing works, the accuracy, $F1$ score, and sensitivity metrics were used. $F1$ combines the precision and recall of a classifier into a single metric. Sensitivity measures the ability of the model to predict true positives of each class.

For the evaluation of the effect of the crowd counting model, three different metrics were used: precision, recall, and mAP, with a confidence level of 0.5 and a range between 0.5 and 0.95. Average precision is a method of condensing the precision-recall curve into a single number that represents the average of all precisions. The mAP is the average of AP. Additionally, mAP(0.5) represents the mean average precision for **Intersection over Union (IoU)** = 0.5, while mAP(0.5 : 0.95) is the averaged mAP for increasing IoU threshold values, from 0.5 to 0.95 by 0.05. The formulas of previous metrics are displayed in Equations (2) and (3) below. **TP (true positive)** implies that a head is detected, and heads exist in the actual image, **FP (false positive)** means that heads are detected when there is no head in the actual image and **FN (false negative)** means that no head is detected, although heads exist in the actual image.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

5.2 Models Performance Analysis

In this section, the performance of the fire detection and crowd counting sub-system is presented. The comparative analysis for the first component of the system was made using the following evaluation metrics: accuracy, F1 score, and sensitivity, and for the second component: precision, recall and mAP, as presented in Section 5.1.3.

5.2.1 Fire Detection Sub-system. The different experiments were conducted using the custom dataset for the fire detection system, as well as the results from previous studies implemented for fire detection, are presented in Table 1. Different model architectures were used, including 2 hidden layers with 60 and 30 nodes, called 1st architecture, 2 hidden layers with 120 and 60 layers, called 2nd architecture, and 3 hidden layers with 120, 60, and 30 layers, called 3rd architecture. Furthermore, three different model representation types were used: 32-bit, 16-bit, and 8-bit.

Regarding the 32-bit models, the best overall results in terms of accuracy, F1 score, and sensitivity were observed in the 2nd and 3rd architectures with values reaching 99%. The sensitivity

Table 1. Comparison of Models Implementation for Fire Detection

Model Architecture	Accuracy	F1 score	Sensitivity	Model Type	Exec. time (msec)	Model size (KB)
60 + 30 layers	0.993	0.987	0.985	FP32	123.7	10.341
60 + 30 layers	0.985	0.981	0.976	FP16	100.5	5.325
60 + 30 layers	0.931	0.916	0.848	INT8	6.68	4.745
120 + 60 layers	0.992	0.991	0.992	FP32	394.8	32.657
120 + 60 layers	0.989	0.990	0.990	FP16	256.8	12.222
120 + 60 layers	0.972	0.964	0.931	INT8	6.70	10.87
120 + 60 + 30 layers	0.993	0.996	0.996	FP32	479.6	39.802
120 + 60 + 30 layers	0.997	0.992	0.992	FP16	340.4	14.282
120 + 60 + 30 layers	**0.974**	**0.975**	**0.942**	**INT8**	**6.80**	**13.191**

Table 2. Comparison of the Best-proposed Model for Fire Detection Using Public Dataset and Different Materials

Model Architecture	Accuracy	F1 score	Sensitivity	Model Type
electric fire	0.998	0.998	0.999	FP32
electric fire	0.996	0.997	0.996	FP16
electric fire	0.981	0.982	0.967	INT8
carton	0.995	0.996	0.998	FP32
carton	0.994	0.995	0.996	FP16
carton	0.983	0.982	0.978	INT8
clothing	0.997	0.997	0.999	FP32
clothing	0.996	0.996	0.994	FP16
clothing	0.979	0.982	0.981	INT8

was affected by an incorrect classification of class 1 (fire occurrences) values to class 0. Thus, a high value in this metric is particularly important for system efficiency. The execution time of the MLP chain for the 2nd and 3rd architecture was 394.8 *msec* and 479.6 *msec*, respectively. Similar results were observed in all metrics, in the quantized models with 16-bit representation, with a small reduction in execution time and model size, as expected.

In terms of the 8-bit models, the best overall results were observed in the 3rd architecture using 3 layers in each MLP model, with an accuracy of 98% and a sensitivity of 94%. As expected, the variation with the fewest bits had a lower ability to separate the two classes, but with a smaller size of the weights' file. The execution time was 6.8 *msec*, with a total model size of 13.191 *KB*.

Experiments were also conducted using the dataset described in Section 5.1.1. The performance of the best model that came up from the above experiments was tested in three sources: an electric fire source, a paperboard cardboards (carton) source, and a clothing source. Table 2 presents the results of the proposed method. As can be observed, the results for each material and for model types FP32 and FP16 were close to 0.99. Additionally, the performance of the proposed model using INT8 as a model type was close to 0.9 in all metrics. Finally, since the models were identical to those in Table 1, the model size and the execution time remained constant.

5.2.2 Crowd Counting Sub-system. Table 3 presents the results of the experiments, which were conducted using variations from the proposed model and the literature. Different parameters were used in terms of backbone, the size of Depth and Width of the backbone layer, and the number of detection layers. The CSPDarknet was used in the literature models, whereas the custom ShuffleNetV2 was used in the proposed models. Since models based on literature could not be imported into the embedded platform, the PANet layer was reduced while the rest of

Table 3. Comparison of the Proposed LW-YOLOv5 and Existing YOLOv5 Implementations (All Models Are in an INT8 Format)

Detector	Backbone	Depth/Width	P	R	mAP (0.5)	mAP (0.95)	Exec. time (sec)	Model size (KB)
YOLOv5s (1 head) [21]	CSPDarknet	0.15/0.15	0.969	0.979	0.975	0.723	187.5	595.4
YOLOv5s (2 heads) [21]	CSPDarknet	0.15/0.15	0.978	0.969	0.995	0.759	295.2	904.1
YOLOv5s (1 head)(0.5 Focus) [21]	CSPDarknet	0.15/0.15	0.973	0.979	0.975	0.716	138.2	528.2
YOLOv5s (1 head)(0.5 Focus) [21]	CSPDarknet	0.20/0.20	0.973	0.979	0.975	0.722	259.2	842.5
YOLOv5s (2 heads)(0.5 Focus) [21]	CSPDarknet	0.15/0.15	0.973	0.969	0.975	0.739	154.4	548.2
YOLOv5s (2 heads)(0.5 Focus) [21]	CSPDarknet	0.20/0.20	0.973	0.979	0.975	0.755	278.1	854.6
our LW-YOLOv5 (1 head)	ShuffleNetV2	0.15/0.15	0.951	0.969	0.970	0.585	16.8	88.47
our LW-YOLOv5 (1 head)	ShuffleNetV2	0.20/0.30	0.957	0.965	0.966	0.602	43.5	171.2
our LW-YOLOv5 (1 head)	ShuffleNetV2	0.50/0.50	0.969	0.976	0.975	0.632	92.2	439.5
our LW-YOLOv5 (2 heads)	**ShuffleNetV2**	**0.15/0.15**	**0.954**	**0.969**	**0.974**	**0.619**	**21.3**	**113.3**
our LW-YOLOv5 (2 heads)	ShuffleNetV2	0.20/0.30	0.969	0.979	0.976	0.634	55.2	219.5
our LW-YOLOv5 (2 heads)	ShuffleNetV2	0.50/0.50	0.969	0.976	0.975	0.671	115.2	539.3

the architecture remained unchanged. The initial learning rate was $1e^{-2}$ and the final was $1e^{-4}$. Additionally, all models were trained for 300 epochs and a batch size of 32.

As can be observed in Table 3, the models with two detection layers had better performance in mAP(0.5) and mAP(0.5 : 0.95) in all cases, when compared to models with a single layer. The addition of a second detection layer improves the model's performance by allowing it to detect objects of variable sizes. Regarding the proposed LW-YOLOv5, the model with two detection layers and with depth and width values of 0.50 had the best overall results in terms of mAP, precision, and recall. However, for a low-power system, the execution time was excessively long. Thus, the model with two detectors and a 0.15 value for both depth and width (bold text) was chosen for the crowd counting sub-system, as the mAP(0.5 : 0.95) value was reduced by 18% compared to the best model, while the execution time was reduced by 92.78%.

Figure 6 presents two different types of loss: box loss and objectness loss. The box loss measures how accurately the algorithm can locate the center of an object and how well the predicted bounding box covers an object. Objectness is a measure of the probability that an object exists in a proposed region of interest. A high level of objectivity indicates a high probability of an object being present in the image window. As can be observed, the model improved significantly in terms of precision, recall, and mAP after 170 epochs and became stable after 200 epochs. The accuracy of the model can be improved by adding more images from different indoor areas.

In comparison to the proposed method, the original YOLOv5 models had better performance in terms of the mAP metric, as can be observed in Table 3. However, the most important aspect of the work of this article is the quick inference of the object detection algorithms and the low memory requirements. Thus, the original YOLOv5 model with the quickest execution time requires 138.2 seconds, which is roughly 18.15% longer than the slowest of all proposed models and 84.5% longer than the fastest. The total memory requirements of the aforementioned original YOLOv5 model is 528.2 *KB*, which is nearly equal to the largest of all proposed models but 79% larger than the smallest. Additionally, the accuracy of the smallest model derived from the literature was 14% better for mAP(0.5 : 0.95) and only 0.1% for mAP(0.5) compared to the selected model.

5.3 Energy Requirements

The total power consumption can be defined as the sum of the power expenditure during the active mode (data acquisition, processing, and inference of ML algorithms) and standby or sleep mode. The load current and voltage were measured using a digital oscilloscope (RIGOL DS1074Z) [43]. Separate jumpers for each component (MCU, sensors) were used to allow for the measurement of the supply currents. Moreover, shunt resistors have been imported in each component with values

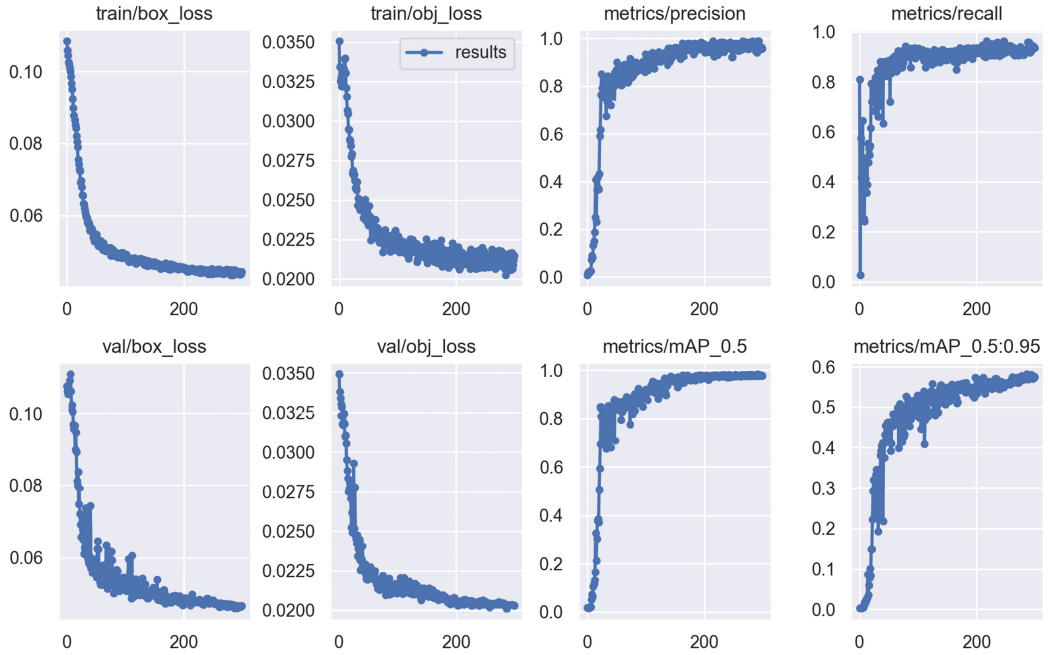

Fig. 6. Plots of box loss, objectness loss, precision, recall, and mean average precision (mAP) over the training epochs and over the training and validation set of the proposed LW-YOLOv5 (bold line of Table 3).

such that the voltage does not fall below 1% of the supply voltage in all cases and therefore presents an acceptably small perturbation. For the measurements, a 2.2V supply voltage was used.

The power consumption of each part in active mode is shown in Table 4. The MCU, the two environmental sensors, and the image sensor are switched on, having a power consumption of 51.36 mW on average with a total energy cost of 1.13J. The most power-intensive process is the execution of the LW-YOLOv5, with C3 layers being the most demanding part of it. Additionally, the execution of the MLP chain, including in the fire detection sub-system, requires only 3.38e^{-5} J and only 6.8 msec.

In standby mode, the MCU operates on 13.7 μA, while the CCS811 sensor consumes only 4.6mA until the value of the eCO_2 concentration crosses the threshold that has been set or the 10-second interval has elapsed. In total, the power consumption in standby mode is 15.2 mW with a peak value of 51.4 mW every 10 seconds.

Figure 7, presents the aggregated power consumption of each component including the power consumption of the MCU, the gas and environmental sensor, as well as the image sensor for 6 min (360.000 msec). Using a 3.7 V Li-Ion battery with 7,000 mAh (25.9 Wh) the system can operate approximately for two months without any human intervention. In contrast, if the system operated continuously at a maximum frequency of 80 MHz, without interrupts and using the fastest model of the original YoloV5 (YOLOv5s (1 head)(0.5 Focus)), then the operation time would be reduced to approximately 20 days.

The crowd counting sub-system will be compared to existing literature works, as there is no similar low-power system in the literature as the proposed whole system. Compared to the existing work [14] from the literature, the proposed method for crowd counting has better results in terms of power consumption and execution time, with a slight decrease in accuracy. Thus, a reduction in power consumption by 78.95% and 84.56% in execution time was observed.

Table 4. Energy Breakdown in Run Mode

Task	Energy (J)	Exec. Time (msec)
Start-up, data acquisition eCO_2, TVOC,T,H, & capture a frame	**0.097**	**≈600**
Execute chain of MLP models	**$3.38e^{-5}$**	**Total: 6.8**
• 4*MLP	$4*8.471e^{-6}$	4*1.7
Execute LW-YOLOv5	**1.03**	**Total: 2,1271.88**
• C.B.R.	0.053	1,108.82
• ShuffleNet	0.071	1,478.43
• ShuffleNet*3	0.062	1,293.63
• ShuffleNet	0.062	1,293.63
• ShuffleNet*3	0.088	1,663.24
• ShuffleNet	0.125	2,587.26
• ShuffleNet*9	0.071	1,478.43
• C.B.R	0.017	369.61
• C3	0.197	4,065.69
• C.B.R	0.008	184.80
• C3	0.125	2,587.26
• C.B.R	0.017	369.61
• C3	0.116	2,402.45
• Detect	0.017	369.61
• NMS	0.001	19.42

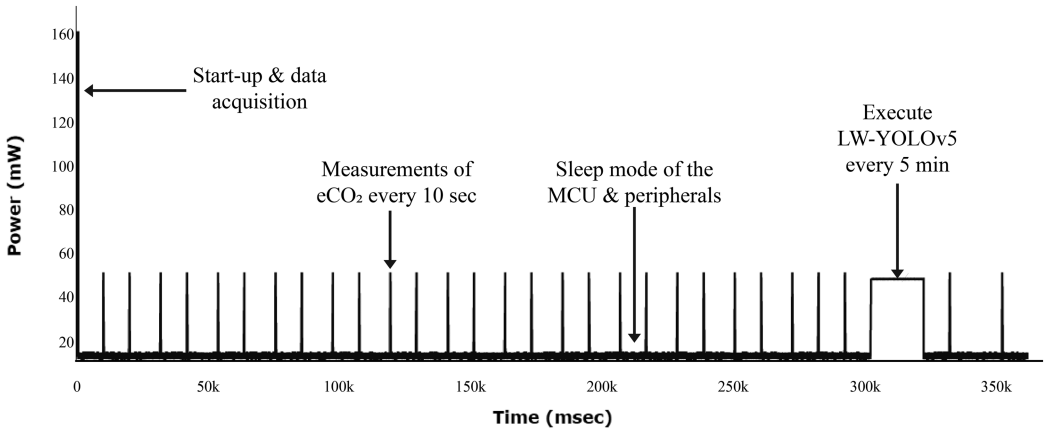

Fig. 7. Aggregated power consumption of the system for 6 mins.

6 CONCLUSIONS

This article presents an ultra-low-power fire detection and crowd counting system. For the fire detection sub-system, environmental and gas sensor along with multilayer perceptron nodes were used. For crowd counting, a custom lightweight version of YOLOv5 was introduced, using

an architecture based on ShuffleNetV2, resulting in a model with low memory requirements, high accuracy predictions, and fast inference on an embedded platform. When a fire incident occurs, the system can send a notification to the fire department along with the total number of people occupying an indoor area to manage the rescue of the trapped people more effectively. Additionally, it can operate with the use of a battery autonomously for up to two months without any charging and even in the case of a power outage.

The evaluation of the two sub-systems was made using custom and public datasets. Specifically, the fire detection method was evaluated in two different datasets. The authors of this article created the first dataset, and the second was based on a public dataset. The crowd counting method was evaluated on a custom dataset created by the authors of this article and included data from two different working areas. In the case of the fire detection sub-system, accuracy close to 97.4% was achieved using the custom dataset and approximately 98% using the public dataset. The model size of the best model (bold line of Table 1) was 13.191 KB and the execution time 6.80 msec. For the crowd counting sub-system, the selected model (bold line of Table 3) had an mAP(0.5 : 0.95) close to 61.9%, a model size 113.3 KB, and an execution time of 21.3 seconds. Finally, comparative analyses with previous studies related to the crowd counting sub-systems were presented.

In terms of future work, regarding the fire detection sub-system, it will be interesting to integrate a low-power CO sensor and compare the results with the approach suggested in this article. Additionally, the development and use of a new gas sensor with reduced power consumption would contribute to a better result. Regarding the crowd counting sub-system, more images for the training phase from different indoor areas would be useful to collect and use. Moreover, it would be interesting to use images from various scenarios during the training phase, such as chaotic situations where people start running around when the alarm is sounded.

REFERENCES

[1] Ibtihaj Ahmad, Zain U. I. Islam, Fahim Ullah, Muhammad Abbas Hussain, and Shahid Nabi. 2019. An FPGA based approach for people counting using image processing techniques. In *11th International Conference on Knowledge and Smart Technology (KST)*. 148–152. DOI: https://doi.org/10.1109/KST.2019.8687568

[2] Zeyad Al-Zaydi, Branislav Vuksanovic, and Imad Habeeb. 2018. Image processing based ambient context-aware people detection and counting. *Int. J. Mach. Learn. Comput.* 8, 3 (2018), 268–273.

[3] European Fire Safety Alliance. 2022. EU-Wide data on residential fires. Retrieved from https://www.europeanfiresafetyalliance.org/.

[4] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. 2018. Post-training 4-bit quantization of convolution networks for rapid-deployment. DOI: https://doi.org/10.48550/ARXIV.1810.05723

[5] Gajanand S. Birajdar, Mohammed Baz, Rajesh Singh, Mamoon Rashid, Anita Gehlot, Shaik Vaseem Akram, Sultan S. Alshamrani, and Ahmed Saeed AlGhamdi. 2021. Realization of people density and smoke flow in buildings during fire accidents using Raspberry and OpenCV. *Sustainability* 13, 19 (2021). DOI: https://doi.org/10.3390/su131911082

[6] Jeong Woo Choi, Xuanjun Quan, and Sung Ho Cho. 2018. Bi-directional passing people counting system based on IR-UWB radar sensors. *IEEE Internet Things J.* 5, 2 (2018), 512–522. DOI: https://doi.org/10.1109/JIOT.2017.2714181

[7] Francesco Conti, Antonio Pullini, and Luca Benini. 2014. Brain-Inspired classroom occupancy monitoring on a low-power mobile platform. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 624–629. DOI: https://doi.org/10.1109/CVPRW.2014.95

[8] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. 2020. TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems. *Proceedings of Machine Learning and Systems* 3 (2021), 800–811.

[9] Arnoldo Diaz-Ramirez, Luis Tafoya, Jorge Atempa, and Pedro Mejia Alvarez. 2012. Wireless sensor networks and fusion information methods for forest fire detection. *Proced. Technol.* 3 (12 2012), 69–79. DOI: https://doi.org/10.1016/j.protcy.2012.03.008

[10] Hussam Elbehiery. 2012. Developed intelligent fire alarm system. *J. Amer. Sci.* 8 (10 2012), 1016.

[11] Varick L. Erickson, Miguel Á. Carreira-Perpiñán, and Alberto E. Cerpa. 2011. OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE, 258–269.

[12] Jordi Fonollosa, Ana Solórzano, and Santiago Marco. 2018. Chemical sensor systems and associated algorithms for fire detection: A review. *Sensors* 18, 2 (2018). DOI: https://doi.org/10.3390/s18020553
[13] Hemant Ghayvat, S. C. Mukhopadhyay, Xiang Gui, and Nagender Suryadevara. 2015. WSN- and IOT-Based smart homes and their extension to smart buildings. *Sensors* 15 (05 2015), 10350–79. DOI: https://doi.org/10.3390/s150510350
[14] Andres Gomez, Francesco Conti, and Luca Benini. 2018. Thermal image-based CNN's for ultra-low power people recognition. In *15th ACM International Conference on Computing Frontiers (CF'18)*. Association for Computing Machinery, New York, NY, 326–331. DOI: https://doi.org/10.1145/3203217.3204465
[15] Fiona Hewitt, Antony Christou, Kathryn Dickens, Richard Walker, and Anna Stec. 2016. Release of volatile and semi-volatile toxicants during house fires. *Chemosphere* 173 (12 2016). DOI: https://doi.org/10.1016/j.chemosphere.2016.12.079
[16] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. 2018. Pointwise Convolutional Neural Networks. arXiv:cs.CV/1712.05245.
[17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Densely Connected Convolutional Networks. arXiv:cs.CV/1608.06993.
[18] Katsuya Hyodo. 2013. Openvino2tensorflow. Retrieved from https://github.com/PINTO0309/openvino2tensorflow.
[19] Osama Talaat Ibrahim, Walid Gomaa, and Moustafa Youssef. 2019. CrossCount: A deep learning system for device-free human counting using WiFi. *IEEE Sensors J.* 19, 21 (2019), 9921–9928.
[20] M. A. Jackson and I. Robins. 1994. Gas sensing for fire detection: Measurements of CO, CO2, H2, O2, and smoke density in European standard fire tests. *Fire Safet. J.* 22, 2 (1994), 181–205. DOI: https://doi.org/10.1016/0379-7112(94)90072-8
[21] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. 2020. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements. DOI: https://zenodo.org/record/4154370/export/hx#.ZAcBTnZBxPY
[22] Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *ArXiv* abs/1806.08342.
[23] Ashish Lalchandani and Samir Patel. 2021. Smart IoT based people counting system. In *International Conference on Artificial Intelligence and Machine Vision (AIMV)*. IEEE, 1–6.
[24] Juan Luis, J. A. Galan, and Javier Espigado. 2015. Low power wireless smoke alarm system in home fires. *Sensors* 15 (8 2015), 20717–20729. DOI: https://doi.org/10.3390/s150820717
[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *European Conference on Computer Vision (ECCV)*.
[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. arXiv:cs.CV/1807.11164.
[27] Kyle Madden, Elaine Ramsey, Sharon Loane, and Joan Condell. 2021. Trailgazers: A scoping study of footfall sensors to aid tourist trail management in Ireland and other Atlantic areas of Europe. *Sensors* 21, 6 (2021). DOI: https://doi.org/10.3390/s21062038
[28] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. 2021. A White Paper on Neural Network Quantization. DOI: https://doi.org/10.48550/ARXIV.2106.08295
[29] Amril Nazir, Husam Mosleh, Maen Takruri, Abdul-Halim Jallad, and Hamad Alhebsi. 2022. Early fire detection: A new indoor laboratory dataset and data distribution analysis. *Fire* 5, 1 (2022). DOI: https://doi.org/10.3390/fire5010011
[30] Pierre-Emmanuel Novac, Ghouthi Boukli Hacene, Alain Pegatoquet, Benoît Miramond, and Vincent Gripon. 2021. Quantization and deployment of deep neural networks on microcontrollers. *Sensors* 21, 9 (Apr. 2021), 2984. DOI: https://doi.org/10.3390/s21092984
[31] Alexios Papaioannou, Panagiotis Verikios, Charalampos S. Kouzinopoulos, Dimosthenis Ioannidis, and Dimitrios Tzovaras. 2021. A low-power embedded system for fire monitoring and detection using a multilayer perceptron. In *IEEE Sensors Applications Symposium (SAS)*. 1–6. DOI: https://doi.org/10.1109/SAS51076.2021.9530090
[32] Wahyu Rahmaniar, W. J. Wang, Chi-Wei Ethan Chiu, and Noorkholis Luthfil Hakim. 2021. Real-time bi-directional people counting using an RGB-D camera. *Sensor Rev.* 41, 4 (2021), 341–349.
[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv:cs.CV/1506.02640.
[34] Wolfgang Roth, Günther Schindler, Matthias Zöhrer, Lukas Pfeifenberger, Robert Peharz, Sebastian Tschiatschek, Holger Fröning, Franz Pernkopf, and Zoubin Ghahramani. 2020. Resource-efficient Neural Networks for Embedded Systems. arXiv:stat.ML/2001.03048.
[35] Faisal Saeed, Anand Paul, Abdul Rehman, Won Hwa Hong, and Hyuncheol Seo. 2018. IoT-Based intelligent modeling of smart home environment for fire prevention and safety. *J. Sensor Actuat. Netw.* 7, 1 (2018). DOI: https://doi.org/10.3390/jsan7010011

[36] Barera Sarwar, Imran Sarwar Bajwa, Shabana Ramzan, Bushra Ramzan, and Mubeen Kausar. 2018. Design and application of fuzzy logic based fire monitoring and warning systems for smart buildings. *Symmetry* 10, 11 (2018). DOI: https://doi.org/10.3390/sym10110615

[37] Robert Adjetey Sowah, Kwaku O. Apeadu, Francis Gatsi, Kwame O. Ampadu, and Baffour S. Mensah. 2020. Hardware module design and software implementation of multisensor fire detection and notification system using fuzzy logic and convolutional neural networks (CNNs). *J. Eng.* 2020 (2020), 1–16.

[38] S. R. Vijayalakshmi and Shanmugam Muruganand. 2016. Real time monitoring of wireless fire detection node. *Proced. Technol.* 24 (12 2016), 1113–1119. DOI: https://doi.org/10.1016/j.protcy.2016.05.244

[39] M. S. Sruthi. 2019. IoT based real time people counting system for smart buildings. *Int. J. Emerg. Technol. Innov. Eng.* 5, 2 (2019).

[40] STMicroelectronics. 2019. STM32 power mode examples - Application note. Retrieved from https://www.st.com/resource/en/application_note/dm00237631-stm32-power-mode-examples-stmicroelectronics.pdf.

[41] STMicroelectronics. 2021. Ultra-low-power Arm®Cortex®-M4 32-bit MCU+FPU, 100 DMIPS, up to 1 MB Flash, 320 KB SRAM, USB OTG FS, audio, external SMPS. Retrieved from https://www.st.com/resource/en/datasheet/stm32l496qe.pdf.

[42] STMicroelectronics. 2021. Ultra-low-power with FPU Arm Cortex-M4 MCU 80 MHz with 1 Mbyte of Flash memory, USB OTG, LCD, DFSDM. Retrieved from https://www.st.com/en/microcontrollers-microprocessors/stm32l496rg.html.

[43] RIGOL Technologies, Inc. 2014. *DS1000Z Series Digital Oscilloscope*. RIGOL Technologies.

[44] Thiago Teixeira and Andreas Savvides. 2007. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *1st ACM/IEEE International Conference on Distributed Smart Cameras*. 36–43. DOI: https://doi.org/10.1109/ICDSC.2007.4357503

[45] T. Tikkanen. 2014. People detection and tracking using a network of low-cost depth cameras. *Aalto University* (2014).

[46] Rita Tse, Tianchen Wang, Marcus Im, and Giovanni Pau. 2020. Privacy aware crowd-counting using thermal cameras. In *12th International Conference on Digital Image Processing (ICDIP'20)*. SPIE, 323–333.

[47] Rishika Yadav and Poonam Rani. 2020. Sensor based smart fire detection and fire alarm system. In *Proceedings of the International Conference on Advances in Chemical Engineering (AdChE'20)*. DOI: https://doi.org/10.2139/ssrn.3724291

[48] Ooi Boon Yaik, Kong Zan Wai, Ian K. T. Tan, and Ooi Boon Sheng. 2016. Measuring the accuracy of crowd counting using Wi-Fi probe-request-frame counting technique. *J. Telecommun. Electron. Comput. Eng.* 8, 2 (2016), 79–81.

[49] Xiuzhu Yang, Wenfeng Yin, Lei Li, and Lin Zhang. 2019. Dense people counting using IR-UWB radar with a hybrid feature extraction method. *IEEE Geosci. Rem. Sens. Lett.* 16, 1 (2019), 30–34. DOI: https://doi.org/10.1109/LGRS.2018.2869287

[50] Norsuzila Ya'acob, Yasser Asrul Ahmad, and Fadhila Nasir Al Mashhoor. 2022. PCB design for IoT based fire detection and alarm system. *J. Posit. School Psychol.* 6, 3 (2022), 8359–8372.

[51] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2017. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv:cs.CV/1707.01083.

[52] Mingyi Zhu and Jiamin Zhang. 2011. Design of fire detection and alarm system based on intelligent neural network. In *2011 3rd International Conference on Computer Research and Development*, Vol. 3. IEEE, 139–142.

[53] Han Zou, Yuxun Zhou, Jianfei Yang, and Costas J. Spanos. 2018. Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT. *Energy Build.* 174 (2018), 309–322. DOI: https://doi.org/10.1016/j.enbuild.2018.06.040

Received 13 July 2022; revised 5 January 2023; accepted 12 January 2023