JST: Smart Systems and Devices
Volume 34, Issue 2, May 2024, 001-009

Navistar - A Universal Internet of Things Platform for Smart Systems

Tran Duy Hien[1], Nguyen Van Giang[1], Phan Minh Tan[1], Tran Trung Dung[1], Hoang Van Hiep[2], Nguyen Dinh Thuan[2], Pham Ngoc Hung[2], Ta Hai Tung[2], La The Vinh[2*]

[1] *The Police Department for Administrative Management of Social Order, Ministry of Public Security, Vietnam*
[2] *Hanoi University of Science and Technology, Ha Noi, Vietnam*
[*] *Corresponding author email: vinhlt@soict.hust.edu.vn*

Abstract

Recent years have seen a rapid development of artificial-intelligence-based (AI-based) internet of things (IoT) systems and applications like real-time object detection systems, face identification, AI-based cameras, etc. Many of these utilize a compact-form computing device based on system-on-chip (SoC) embedded modules from different providers like Qualcomm, Renesas, Quectel, etc. One of the most popular and easy-to-access platforms both practically and academically is the Raspberry Pi computing device and other PI-like devices such as: Orange Pi, Banana Pi, Jetson Nano, etc. However, we realize that the existing platforms strongly focus on the computation aspect, while other important features such as communication, location, ease of integration with high-level operating systems are lacking or poorly supported (through the use of so-called HAT). Therefore, in this work we propose a design of a multi-purpose IoT platform with full integration of computation module, IO ports, communications, location and high-level operating system (Android). We also utilize this IoT platform in a variety of applications including data collection, location-based services and biometric identification. Those sample applications will be presented in our work together with the algorithm and testing result to demonstrate the practical use of our platform.

Keywords: IoT, AI, single board computer, smart SoC.

1. Introduction

Recently, artificial internet of things (AIoT), a combination of artificial intelligence technology and internet of things infrastructure, has seen a rapid development and application. AIoT provides more efficient IoT operations, better human-machine interactions and advanced data management and analytics. As the result, a variety of compact-form computing devices (also called single board computer or shortly SBC) are manufactured and provided to the users, to name some: Raspberry Pi, Orange Pi, Jetson Nano (see Fig. 1) [1]. Though, those devices are quite suitable for experimental purposes in laboratory environments with indoor communications like Bluetooth, Wifi, or local ethernet. The deployment of those in the field is complicated because any practical deployment may require a wide-area communication channel like 3G or 4G, a flexible power supply (with changing voltage and current), and other industrial-grade level features like ability to deal with sleeping, electric shock, dusty environment… From technical point of view, the lack of features on single-board computers (SBC) like 3/4G communication, location, serial I/O channels, etc. can be solved by using a variety of the so-called hat (see Fig. 2), a kind of extension module. However, using those external modules significantly decreases the stability of the whole system.

Therefore, in this work our purpose is to introduce our self-designed SBC for IoT, which is capable of:

- Providing all necessary features on-board;

- Working with a high range of input voltage;

- Providing the ability to work with low-level general-purpose input and output (GPIO) pins as well as high-level input and output (I/O) ports;

- Connecting to both short-range communication network (Wifi, Bluetooth) and long-range connections through the use of 3/4G channels;

- Positioning with multiple global navigation satellite systems (GNSS);

- Connecting to external peripherals using I/O ports of different protocol: Serial UART/RS232, I2C, SPI;

- Monitoring on-off signals of highly-voltage-variable input;

- Developing software using high-level software development kit (Android SDK).

ISSN 2734-9373
https://doi.org/10.51316/jst.174.ssad.2024.34.2.1
Received: Dec 1, 2023; revised: Feb 21, 2024;
accepted: Feb 27, 2024

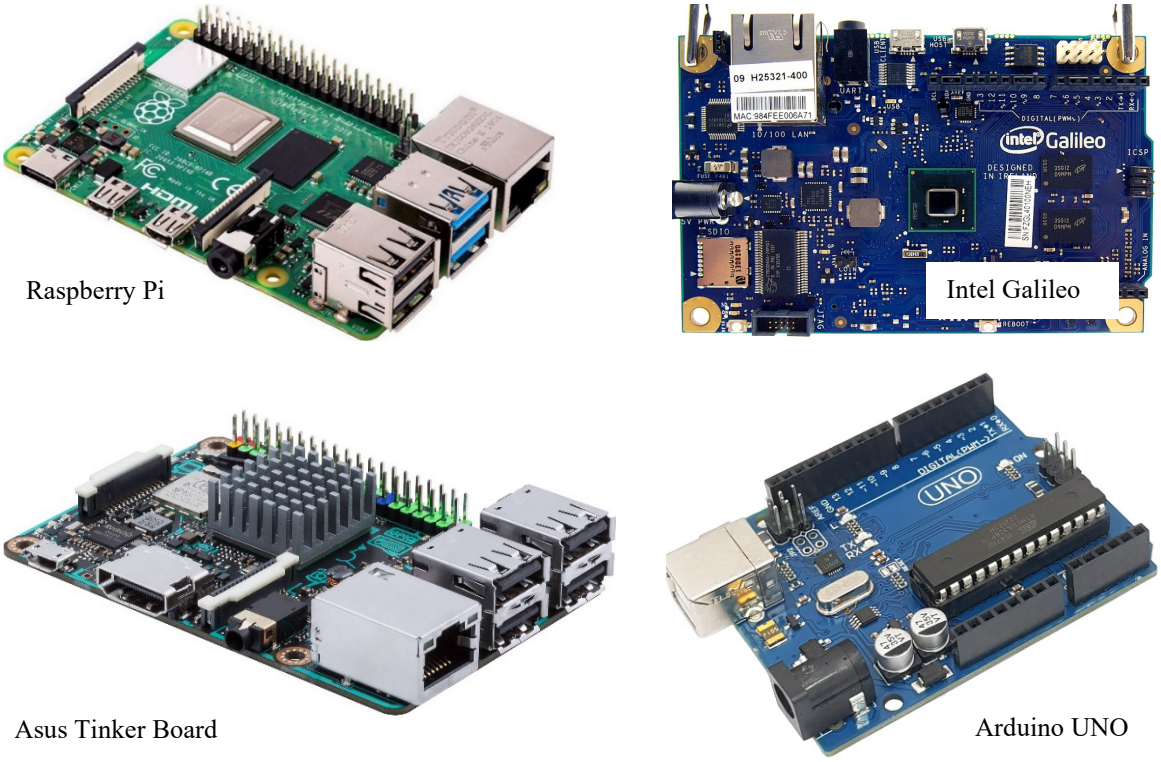

Fig. 1. Single board computers.

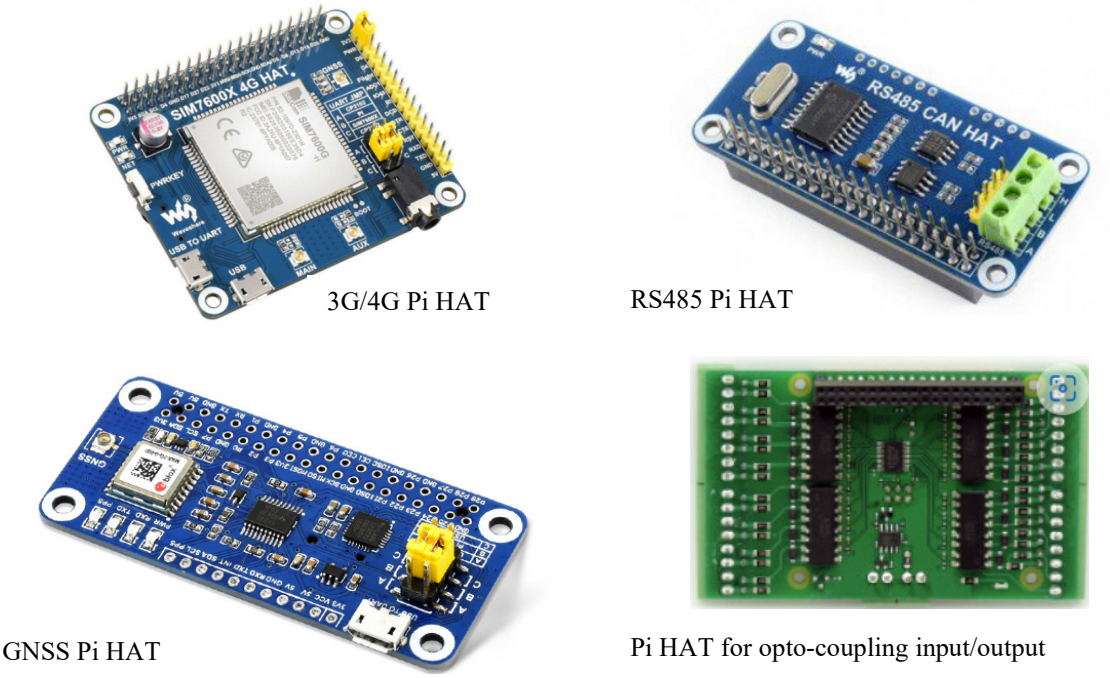

Fig. 2. Different kind of HATs (external modules) for SBC.

We provide our reference schematic design publicly to make it easily accessible and extensible for the community. Besides, we also describe some of our existing applications which are already built and proven to work well on the proposed design. With such an open approach, we hopefully expect the proposed design may get improved gradually and become a mutual platform for IoT applications.

Our work is presented in the following structure: the first section introduces the motivation, the next section gives some related work in this topic, we detail our design in the third section with some algorithms, applications are described in the fourth, and finally the paper is concluded in the last section.

2. Related Work

The concept of a computing facility with different kind of peripherals has been developed for many years. In the last decade, a large number of such facilities are available on the market, for example: Arduino UNO, Arduino Mega, ESP 32; these are examples of low-level computing devices that can be used for IoT applications. Recently, even more powerful devices, which support high-level operating systems (Linux, Android or Windows), are getting popular in the IoT community, and often called single board computers (SBC - see Table 1) [1].

Together with the high availability of those computing devices, the research community realizes the potential of using SBC for rapid prototyping, testing and deploying IoT systems. In [2] the author investigates the use of Raspberry Pi 3 Model B+ for implementing a weather station. Also in this work, the author points out that Wifi connection is a barrier preventing their prototype to be deployed in real-life scenarios. Authors of [3] utilize SBC in the education area, wherein SBC are used in different purpose: the core of a smart classroom projection, teaching materials, and rapid prototyping of embedded systems like mobile robot, unmanned aerial vehicle (UAV), spherical robot. SBCs are not only simply used as a data collection/transmission mechanism like the mentioned work above but also are deployed as a kind of mini server as in [4] and powerful edge-computing devices as in [5].

The author of [6] perform an evaluation of executing a support vector machine (SVM) classifier on a Raspberry Pi and compare the result with that on a laptop PC. It is shown that the classification accuracy is similar but the training time on SBC is surely significant longer. They also evaluate other kinds of machine learning models like multi-layer perceptron network (MLP) and convolution neural network (CNN), and the result indicates that SBC is practically acceptable regardless of the longer execution time. In [7], the researchers apply CNN model to recognize face image emotion on a Raspberry Pi 3. They conclude that their solution shows satisfactory results in proper tasks with a cost-effective hardware platform.

Another AI-related research for IoT is [8], this work proposes a combination of OpenCV and Raspberry Pi 3 for face classification tasks. So far, it can be seen that SBC is quite a good choice for a variety of applications, however those applications have just utilized the computing capability and not yet really make use of the low-level general purpose input output (GPIO) pins, which are one of the important features on SBC for embedded systems.

Table 1. Some typical SBCs

	Arduino Uno	**Intel Galileo**	**Intel Edison**	**ESP8266**	**Beagle Bone**	**BPI P2 Zero**	**Raspberry Pi 4B**
CPU	ATMega 328P	Intel Quark SoC	Intel Quark SoC	RISC L106 32 bit	ARM Coretex A15	ARM Coretex A7	Broadcom SoC BCM2711
GPU	NO	NO	NO	NO	PowerVR SGX544	Mali 400 MP2	Broadcom VideoCore
Clock	16 MHz	400 MHz	100 MHz	80 MHz	800 MHz	800 MHz	800 MHz
RAM	2 KB	256 MB	1 GB	32 KB	512 MB	512 MB	4 GB
Storage	32 KB	8 MB	4 GB	80 KB	4 GB	8 GB	4 GB
Communication	Wifi, Bluetooth, Ethernet, Serial	Wifi, Bluetooth, Ethernet, Serial	Wifi, Bluetooth, Ethernet, Serial	Wifi, Bluetooth	Wifi, Bluetooth, Ethernet, Serial	Wifi, Bluetooth, Ethernet, Serial	Wifi, Bluetooth, Ethernet, Serial
I/O Ports	SPI, I2C, UART, GPIO	SPI, I2C, UART, GPIO	SPI, I2C, I2S, UART, GPIO	SPI, I2C, UART, GPIO	SPI, I2C, I2S, CAN, UART, GPIO	SPI, I2C, I2S, UART, GPIO	SPI, DSI, SDIO, CSI, UART, GPIO
Programming Language	Wiring	Wiring, Wyliodrin	Wiring, C/C++, HTML5	C/C++, Python, Ruby	C/C++, Python, Ruby, Java, Shell	C/C++, Python, Java	C/C++, Python, Java, Scratch
Estimated Cost	$20	$70	$50	$4	$270	$30	$35

In [9], the authors integrate an SBC with a programmable logic controller (PLC) to build an embedded control system. [10] is another research work that makes use of both the high-level operating system (to run a Python application) and low-level gpio pins (to control a gas flow) on a Raspberry Pi-Based embedded system.

3. Navistar Design

From our above analysis of the trend and related research it can be seen that the motivation of utilizing SBC in IoT applications are increasing gradually. It, however, can also be noted that the existing SBCs are designed mostly for laboratory environments without much capability of handling: variable input power, input isolation, wide-area communication, electric shocks. Raspberry Pi, one of the most popular SBC, can be a typical example of our analysis above; it uses DA9090 as its power management IC (PMIC), which is quite easy to be dead and impossible to be repaired. Besides, Raspberry Pi requires external modules (called HAT) to provide features like positioning, mobile communication, opto-coupler input, RS232/485 communication, etc.

Therefore, we propose our open-source design of a single board computer to complement the existing disadvantages; our design is divided into four main blocks (see Fig. 3):

- Input power management;

- Low-level microprocessor;

- High-level central processing unit (CPU);

- Input and output.

Detail schematics and PCB are also provided in the following paragraphs.

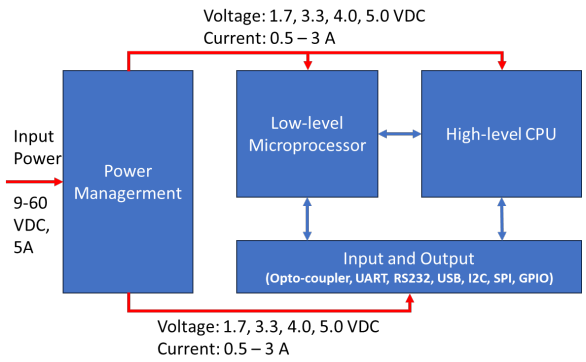

Fig. 3. Block diagram of the proposed design

Power Management (see Fig. 4): to make the designed SBC capable of working with a variety of input voltage values, which is the usual case in practice, we utilize the Texas Instruments LM2576 HVS ADJ power converter. We separated input power for the low-level microprocessor and the high-level CPU to optimize the operation of the whole system in different scenarios in which a sleep monitoring mode with very low power consumption is necessary. In such a situation, only the low-level microprocessor lives to wait for an external trigger signal, power of the main CPU is driven off by Q_EN_IN pin of the microprocessor. Besides, MIC29302WU-TR from Microchip is used to generate a 4-VDC input for the main CPU (also controlled by Q_EN_CTL in sleeping mode). MIC29302 is a LDO regulator, which can provide more stable output voltage than that of the switching regulator. However, the heat released by MIC29302 increases as the difference between input and output voltage increases. Therefore, we need a switching regulator before it. Table 2 and 3 describe some important attributes of both LM2576 and MIC2930 ICs.

Table 2. LM2576 attributes

Attributes	Value
Manufacturer	Texas Instruments
Output voltage	1.23 V to 57 V
Output current	3 A
Input voltage	4 - 63 V
Switching frequency	52 KHz
Number of outputs	1

Table 3. MIC29302WU-TR attributes

Attributes	Value
Manufacturer	Microchip
Output voltage	1.25 V to 25 V
Output current	3 A
Input voltage	2.3 - 26 V
Type	LDO Voltage Regulators
Number of outputs	1

In addition to the two main power ICs, we also use AP7365-ADJ IC from Diodes Incorporated to generate low-current I/O level voltages (1.7, 2.8, 3.3 VDC - 0.5 mA).

In our solution, we integrate a ATMega 8 microprocessor. This microprocessor serves two main purposes: (1) in sleeping mode, it drives the power of the main CPU and other power-consuming peripherals off to save the battery; an external trigger signal will turn on the whole system again. (2) in the normal operation, it acts as a hardware watchdog timer to prevent the main CPU from being stuck in a suspension. The microprocessor communicates with

the main CPU via two general purpose pins PD0 and PD1; any application running on the main CPU has to keep a clock signal of 2-Hz frequency on PD0 so that the reset counter inside the microprocessor does not reboot the CPU; otherwise, any kind of suspension occurs corrupting the clock signal, the microprocessor will reboot the CPU after a desired period of time. This specific design is necessary since the high-level application and operating system running on the main CPU always have some probability of being suspended, especially when they keep running for a long period of time.

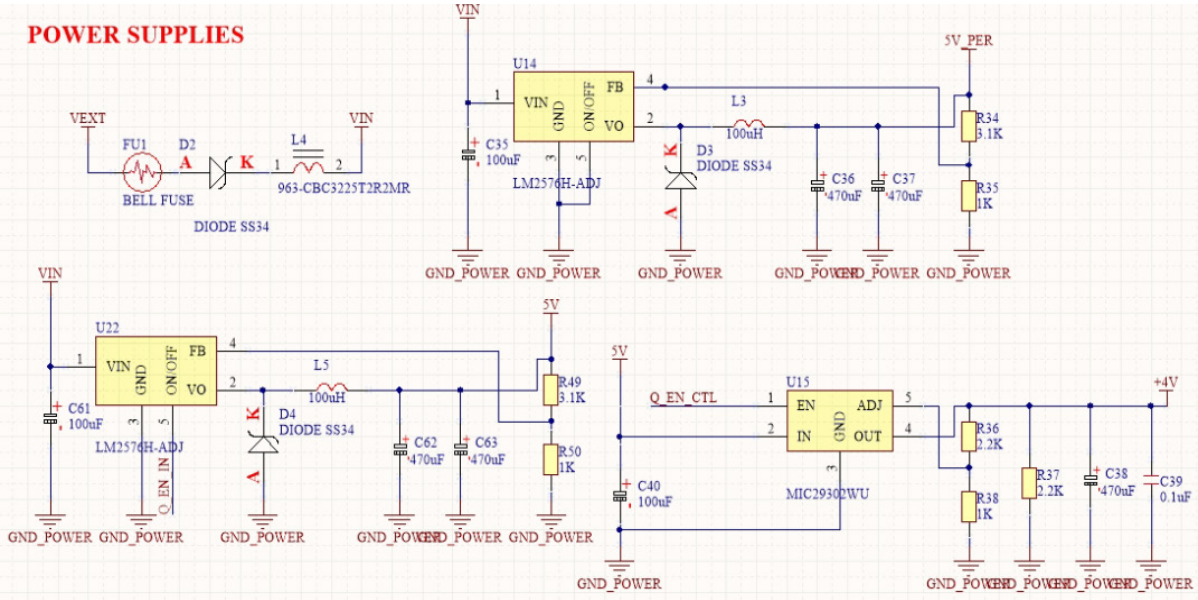

Fig. 4. Power management schematic

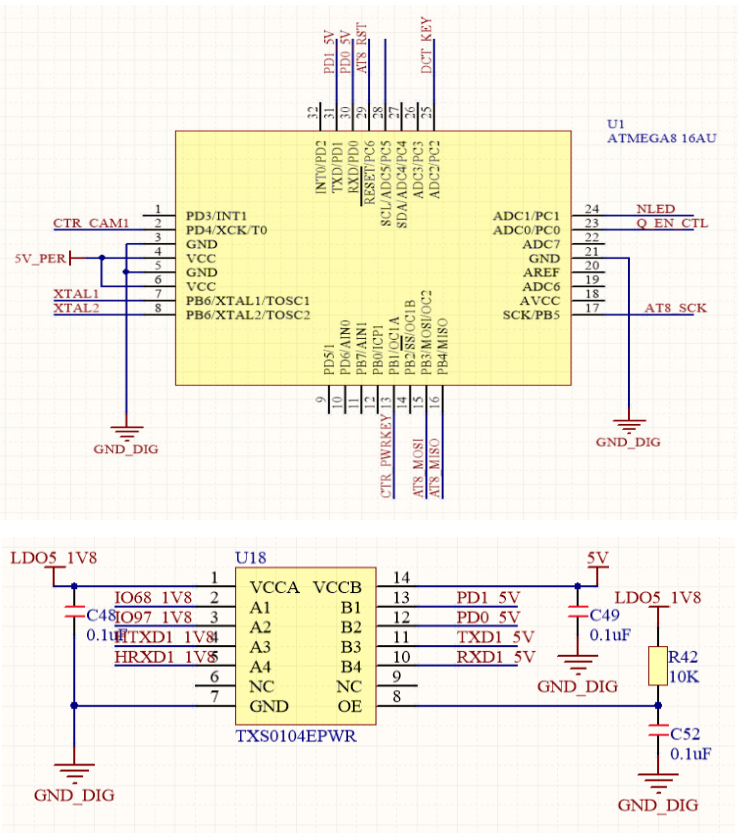

Fig. 5. ATMega8 Microprocessor schematic

The heart of the whole design is a Qualcomm-based System on Chip (SoC). We utilize Quectel SC series of SoC; SC series include several pin-compatible versions which are fulfil different performance requirement from the users (see Table 4).

Table 4. Quectel SC series of SoC

Name	Basic specification
SC20	32 bit quad-core ARM Cotex-A7 1.1 GHz 8 GB eMMC + 1 GB LPDDR3 (2 GB optional) Adreno 304 GPU Android/Linux OS Supply voltage: 3.5 - 4.2 VDC, 3[a]
SC200R	64 bit quad-core ARM Cotex-A53 1.3 GHz 8 GB eMMC + 1 GB LPDDR3 (2 GB optional) Adreno 308 GPU Android/Linux OS Supply voltage: 3.5 - 4.2 VDC, 3[a]
SC200L	64-bit octa-core ARM Cotext-A53 1.4 GHz 16 GB eMMC + 2 GB LPDDR3 (4 GB optional) Mali T820 GPU Android/Linux OS Supply voltage : 3.5 - 4.2 VDC, 3A
SC200K	64-bit octa-core ARM Cotext-A55 1.6 GHz 16 GB eMMC + 2 GB LPDDR3 (4 GB optional) GE8322 GPU Android/Linux OS Supply voltage : 3.5 - 4.2 VDC, 3A

Besides the computing power of the Qualcomm-based CPU and GPU, SC series also provides a rich set of other features including positioning engine, I/O interface, communication, etc.. We detail some key features in Table 6 below.

Fig. 6. illustrate the schematic design of the SoC module. It can be seen from the schematic, we use an opto-coupler TLP2904 with four channels to isolate the binary input to the CPU. This isolation helps ensuring that the system can work receive external trigger signals at a wide range of voltage. Since SC series provide UART interfaces only, we keep one UART port and interface the other to a RS232 transceiver (MAX232E) to provide better compatibility serial ports when connecting to external peripherals. USB port by default works as a slave device when debugging and developing apps; however, in many applications we need the so-called OTG (on-the-go) feature with an USB host. Therefore, we put two switches to config the USB to be slave or host as our desire. These two switches mainly do two things: power up the USB peripheral and pull down the USB ID pin to the ground in host mode. The peripheral power is generated by the LM2576 HVS ADJ that we described previously. Since the maximum current of the power IC is up to 3 A, it is surely enough for most of the USB peripherals. The host USB can also be extended by using external hub or converter to allow more than one peripheral; in our application described below we have successfully connected two HD camera with 30 fps in Jpeg format.

Table 5. SC series' features

Name	Value
IO Ports and GPIO	2xUART, 1xUSB, 1xMIPI (display), 1xCSI (camera), 1xI2C, 20+ GPIO Pins
Positioning	Multi-constellation GPS, GLONASS, BEIDOU, GALILEO
Communication	Wifi, BLE 4.2, GSM/UMTS/LTE connection,
Codecs	Audio: AMR-NB, AMR-WB, EVS-NB, EVS-WB, EVS-SWB, MP3, AAC, AAC+, AMR, PCM Video: Encoding/Decoding 1080P @ 30 fps + 1080P @ 30 fps
ADC	One general purpose ADC interface, supports up to 12-bit resolution
Realtime Clock	Supported

To conclude this section, we have briefly presented our schematic design of the proposed SBC using Quectel SC series. We put into the solution some additional components like the microprocessor, RS232 transceiver, USB configuration switch, voltage level buffer to ensure a smooth connection of different internal components and external peripherals. Fig. 7 illustrates the final PCB and photos of the manufactured product. The advantages of this approach are clearly the robustness of the power management block, and the flexibility of I/O block, the convenience of application development in high-level programming languages and operating systems. Since it is difficult to present all the details in the scope of this paper, we put all the design material on a public link for the reader to access (https://shorturl.at/sFIK7).

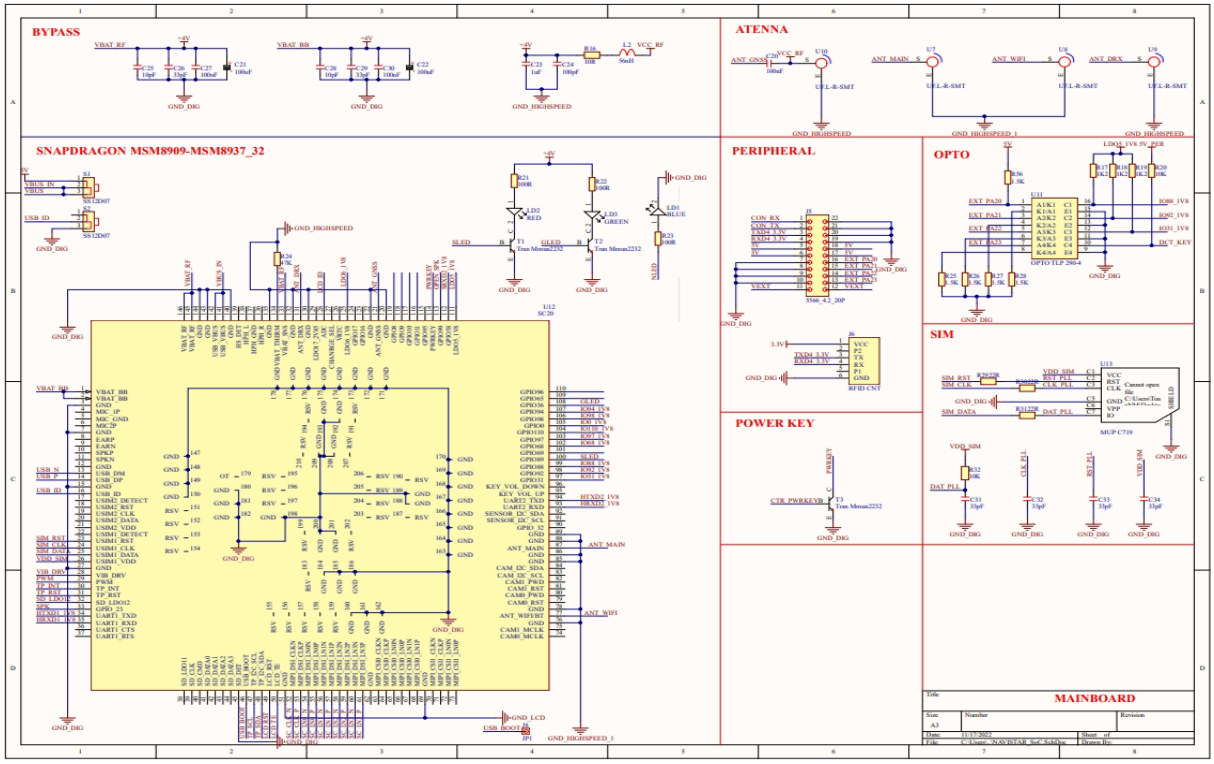

Fig. 6. Quectel SoC schematic

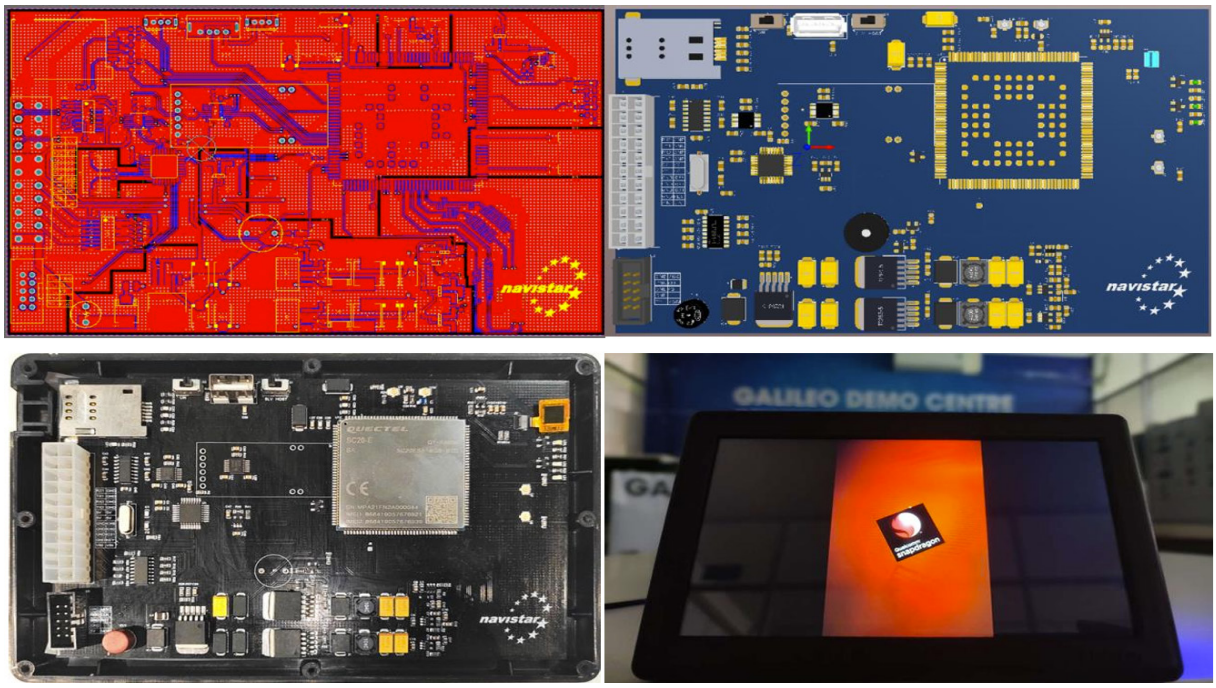

Fig. 7. PCB design (Top-Left), 3D rendering (Top-Right),
Assembled device back side (Bottom-Left), Assembled device front side (Bottom-Right)

4. Applications

In this section, we demonstrate the use of our designed SBC with a computer vision application, in which we implement an Android client application to recognize face and liveness detection based on the Google's MLKit. In this demonstration, we even use the lowest CPU of SC20E SoC and our experiments show that with such a low-performance CPU, we still can achieve at frame rate of 5 frames per second, which is reasonable.

ML Kit is a mobile software development kit (SDK) that brings Google's on-device machine

learning expertise to Android and iOS apps. All are powered by Google's best-in-class machine-learning (ML) models and offered to developers at no cost. ML Kit has been used and evaluated in a number of existing research to be one of the best AI SDK on mobile devices. ML Kit vision APIs has several modules (for example: face detection, face mesh detection, text recognition, image labelling, object detection and tracking, etc.), in which we utilize the face mesh detection module to implement our demonstration.

Face mesh detection API takes an input face image and generates 468 3D points of the face (should be within about 2 meters of the camera). Fig. 8 illustrates an example of the mesh detection API's output.

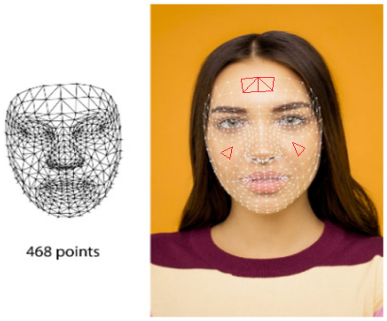

Fig. 8. Output of the face mesh detection

Though, face mesh detection API generates a mesh of points, it does not provide the classification feature. Therefore, we proposed our method for face classification based on the generated points. In our experiments, using the full mesh does not bring a good classification accuracy because many of the points (or triangles) do not contain any specific facial features (such as those on the forehead area, we highlight some using triangles on the girl's face in Fig. 8). That is the reason why we propose to use a subset of points and triangles which contain facial features (for example: distance between two eyebrows, length of the nose bridge, length of the philtrum, size of the mouth, etc.). Those points (together with their index in the total 468 points) and triangles are presented in Fig. 9 below.

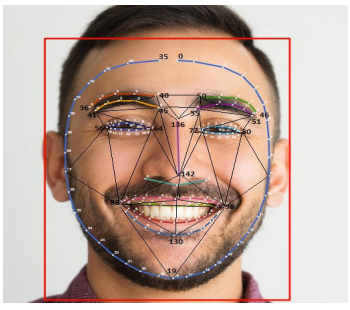

Fig. 9. The proposed triangles used in face classification experiments.

Let two faces be represented by two sets of triangles:

Face A: $\{T_1^A, T_2^A, \dots, T_N^A\}$ and

Face B: $\{T_1^B, T_2^B, \dots, T_N^B\}$.

A triangle T is represented by a set of three points $\{\begin{pmatrix}x_1\\y_1\\z_1\end{pmatrix}, \begin{pmatrix}x_2\\y_2\\z_2\end{pmatrix}, \begin{pmatrix}x_3\\y_3\\z_3\end{pmatrix}\}$. We perform a face matching between face A (assumed that A is an image from the camera in realtime) and face B (another facial image in the database associated with personal information) by calculating a similarity score.

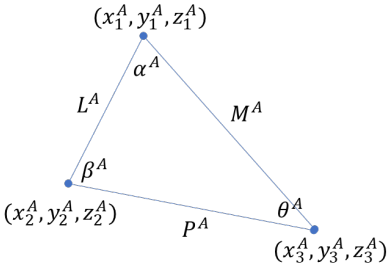

Fig. 10. A triangle's parameters

To calculate the similarity score, we first compute the angle of each triangle given the coordinates of all three points (as shown in Fig. 10). We denote L^A, M^A, P^A are the length of three edges, those quantities are calculated from the point coordinates. The angles are computed as follows:

$$\alpha = \text{acos}\left(\frac{M^2 + L^2 - P^2}{2LM}\right) \quad (1)$$

$$\beta = \text{acos}\left(\frac{L^2 + P^2 - M^2}{2LP}\right) \quad (2)$$

$$\theta = \text{acos}\left(\frac{M^2 + P^2 - L^2}{2MP}\right) \quad (3)$$

Given two triangles A and B with corresponding angle values $(\alpha^A, \beta^A, \theta^A)$ and $(\alpha^B, \beta^B, \theta^B)$, the similarity score between A and B is defined as:

$$S^{AB} = \frac{\left|\frac{\alpha^A}{\alpha^B} - 1\right| + \left|\frac{\beta^A}{\beta^B} - 1\right| + \left|\frac{\theta^A}{\theta^B} - 1\right|}{3} \quad (4)$$

S^{AB} is a non-negative real value, $S^{AB} = 0$ means that the two triangles are congruent; the greater value of S^{AB} means the higher dissimilarity between A and B. The similarity between two sets of triangles (two faces) are calculated as the average similarity between all pairs of corresponding triangles. The matched face of an input face (I) is decided by:

$$Matched\ Face = argmin_{A\ in\ face\ DB} S^{AI} \quad (5)$$

Since the proposed approach is based on the average congruency of triangles, it mitigates the image size problem in matching.

We evaluate our approach with a database of ten people (randomly selected from our staff page: https://soict.hust.edu.vn/can-bo), with ten tests for each in different light conditions and achieve an accuracy of 95%. Since the matching process is based on simple geometry calculations, most computation resource consumed by the mesh detection phase. We measure and see an average of about two hundred milliseconds per frame; this performance allows the device to process up to five frames per seconds, which is practical enough for daily use (for example: access control, attendance check, etc.). We also make a performance comparison of our board, Raspberry Pi 3 and Raspberry Pi 4 running the mesh detection algorithm; the result is presented in Table 6 below.

Table 6. Performance comparison

	Our board	Raspberry Pi 3	Raspberry Pi 4
CPU	Quectel SC20 1.2 GHz Quad-core	Broadcom Cortex A53 1.4 GHz Quad-core	Broadcom Cortex A72 1.8 GHz Quad-core
RAM	1GB	1GB	1GB
OS	Android (AOSP)	Android (Lineage)	Android (Lineage)
FPS	4.9	5.2	7.6

5. Conclusion

In this work, we have presented an overview of the use of single board computers (SBC) in research, and we also point out that existing SBCs have limitations such as: connectivity availability, robustness against various power input, flexibility, and expandability. From our observations, we propose to open our design as an open-source project and describe all the detail in this paper to make it more accessible for users. We also present preliminary result from our tests with the proposed SBC to see that even with the lowest CPU configuration, it is still practical to deploy complicated algorithms with real-time performance.

For our conclusion, in this work we have two main contributions: (1) propose, implement, and publish a practical design of a single board computer, (2) propose a simple face matching algorithm based on Google ML Kit for evaluating the design in real-time.

Acknowledgments

This work was supported by the project under the Grant T2023-PC-005.

References

[1] Kanagachidambaresan, G. R., Role of Single Board Computers (SBCs) in rapid IoT Prototyping. Springer, 2021. https://doi.org/10.1007/978-3-030-72957-8

[2] Johnston, Steven J., *et al*. Applicability of commodity, low cost, single board computers for Internet of Things devices. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). IEEE, 2016. https://doi.org/10.1109/WF-IoT.2016.7845414

[3] P. Amruthavarshini, C. V. Raghu, and G. Jagadanand. Development of an IoT enabled smart projection system for classroom needs. 2022 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT). IEEE, 2022. https://doi.org/10.1109/IAICT55358.2022.9887528

[4] Godinho, António, *et al*. IoT single board computer to replace a home server. 2023 18th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2023. https://doi.org/10.23919/CISTI58278.2023.10212031

[5] Ali, Omer, *et al*. A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface. Sensors 22.3 (2022): 995. https://doi.org/10.3390/s22030995

[6] Guha, S., Chakrabarti, A., Biswas, S., & Banerjee, S. (2020, December). Implementation of Face Recognition Algorithm on a Mobile Single Board Computer for IoT Applications. In 2020 IEEE 17th India Council International Conference (INDICON) (pp. 1-5). IEEE.

[7] Mikhaylevskiy, S., *et al.* Fast emotion recognition neural network for IoT devices. 2021 International Seminar on Electron Devices Design and Production (SED). IEEE, 2021. https://doi.org/10.1109/SED51197.2021.9444517

[8] Sruthy, S., and Sudhish N. George. Wifi enabled home security surveillance system using Raspberry Pi and IoT module. 2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES). IEEE, 2017. https://doi.org/10.1109/SPICES.2017.8091320

[9] Galkin, Pavlo, Lydmila Golovkina, and Igor Klyuchnyk. Analysis of single-board computers for IoT and IIoT solutions in embedded control systems. 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). IEEE, 2018. https://doi.org/10.1109/INFOCOMMST.2018.8632069

[10] Twinn, Gary. Combining low-cost single board computers with open-source software to control noble gas extraction lines. MethodsX 10 (2023): 101974. https://doi.org/10.1016/j.mex.2022.101974