Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Especialidad en Sistemas Embebidos

Design of an automotive embedded system for stratospheric environments

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presenta:

JULIÁN CAMILO ARISTIZÁBAL ARISTIZÁBAL
MAXIMILIANO MANCILLA TORRES
MARCO ANTONIO MEDRANO ACOSTA
NOE GUADALUPE ORTIZ
CARLOS ALBERTO ZEPEDA LÓPEZ

Asesor **LUIS RIZO DOMÍNGUEZ**

Tlaquepaque, Jalisco. Agosto de 2024.

Design of an Automotive Embedded System for Stratospheric Environments

Julian Camilo Aristizabal Aristizabal
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
julian.aristizabal@iteso.mx

Maximiliano Mancilla Torres
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
mmancilla.torres@iteso.mx

Marco Antonio Medrano Acosta
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
marco.medrano@iteso.mx

Noé Guadalupe Ortiz
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
noe.ortiz@iteso.mx

Carlos Alberto Zepeda Lopez
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
carlosa.zepeda@iteso.mx

Luis Rizo Dominguez
Departamento de electrónica, sistemas e informática
Instituto tecnológico y de estudios superiores de Occidente
Tlaquepaque, México
lrizo@iteso.mx

***Abstract*—The Experimental Module for the Iterative Design for Satellite Subsystems (EMIDSS) is a nanosatellite project to obtain meteorological data from the low Earth orbit. The aim of this fifth version (EMIDSS-V) is to optimize the code and architecture from the previous versions. The AUTOSAR-like architecture model from EMIDSS-IV was preserved, while the application components were redesigned based on the NXP S32K144W microcontroller. For the memory module, the driver was recoded, and a new reading/writing algorithm was added. Also, the sensor driver interfaces were rewritten to standardize data collection. Additionally, a battery module was incorporated to make EMIDSS-V energy-independent. Finally, a user interface module was added to facilitate the configuration and data retrieval from the module. This work will serve as the basis for EMIDSS-VI.**

Keywords—nanosatellite, microcontroller, AUTOSAR, sensor, memory, user interface, UDS, software architecture

I. Introduction

Nanosatellites have revolutionized the space exploration field in recent years by allowing small companies and universities to test their new developments. Nanosatellites are characterized by their low weight, which reduces the costs of launching into the earth's orbit compared to traditional satellites, and by their shorter development cycle, allowing postgraduate students to be involved in the development process [1]. Another factor that can explain the popularity of nanosatellites among universities is the inner challenges of their architecture, such as their small size and reduced energy storage capacity [2].

The Experimental Module for the Iterative Design for Satellite Subsystems (EMIDSS) is a project developed by the *Instituto Tecnológico y de Estudios Superiores de Occidente* (*ITESO*) in partnership with the *Universidad Nacional Autónoma de México* (*UNAM*) and the *Instituto Politécnico Nacional* (*IPN*). Its purpose is to collect and save environmental data from the stratosphere. ITESO joined the project in the third version, in which the software architecture was built around a Field Programmable Gate Array (FPGA) [3]. Its main accomplishments included the development of both hardware and software to properly record environmental data such as temperature, humidity, pressure, and air quality. However, the addition of new features was restricted due to a lack of documentation and a highly hardware-dependent software architecture. To address this, the fourth iteration was intended to facilitate the addition of new software modules by creating a scalable software architecture [4]. As a result, the main microcontroller (MCU) was changed to a S32K144W NXP MCU. The new architecture was based on AUTOSAR, a layered architecture in which the application is separated from the firmware, facilitating the incorporation of new software modules. Nevertheless, this iteration had issues with the memory-saving algorithm and the large amount of software libraries that were implemented, increasing the complexity of the software development process. Therefore, the main objective of this work is to optimize the software used in EMIDSS and solve the memory issues from EMIDSS-4.

The embedded system described in this article is tested in NASA's Columbia Scientific Balloon Facility (CSBF). The proposed system will be launched during the FY 24 schedule in New Mexico, USA.

This paper is organized as follows: Section II explains the proposed software modules. Section III describes the results of the tests applied to the new features. Section IV presents the conclusions and future work.

II. PROPOSED DESIGN

A. Hardware Architecture

The hardware architecture of EMIDDS-V (seen in "Fig. 1") was built to accomplish its main functionality, which is the storage of meteorological data in flash memory. The architecture contains the following components:

- Microcontroller – S32K144W, an MCU manufactured by NXP was used. This MCU features an ARM Cortex-M4 architecture that provides the necessary resources to accomplish the requirements of the nanosatellite [5].
- Humidity and temperature sensor – required to obtain meteorological data via I2C. This sensor was selected due to its data precision and communication capability.
- Memory flash – stores the sensor data. An SST26VF032B memory flash was selected for its efficiency. This module provides high-speed data transfer and the capability to read/write data, including the incorporation of useful services such as reset and mass erase via SPI.
- Dual battery charger – provides a battery management system via SPI. This module manages the battery selector and battery protection against short circuits.
- Buttons and LEDs – indicate power and SW run status.
- Connector PC/40 – supports external connections.

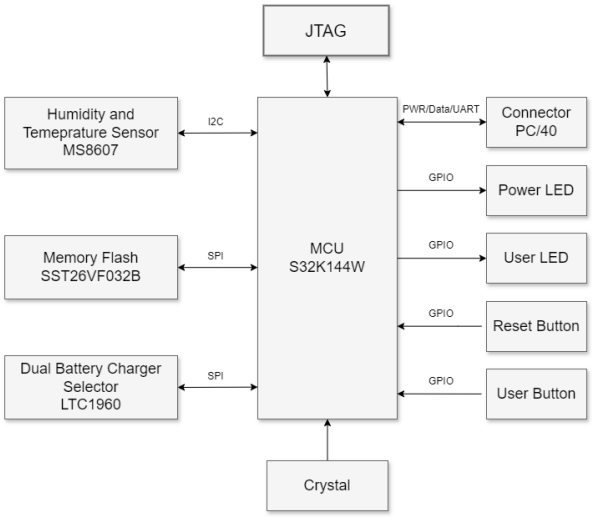

Fig. 1. EMIDSS-5 Hardware architecture.

B. Software Architecture.

The software architecture is based on AUTOSAR 4.4, which is a standard used in the automotive industry [6]. The purpose of this architecture is to obtain the highest abstraction level in the software layers: application layer (ASW), run time environment (RTE), and basic software (BSW) ["Fig. 2"].

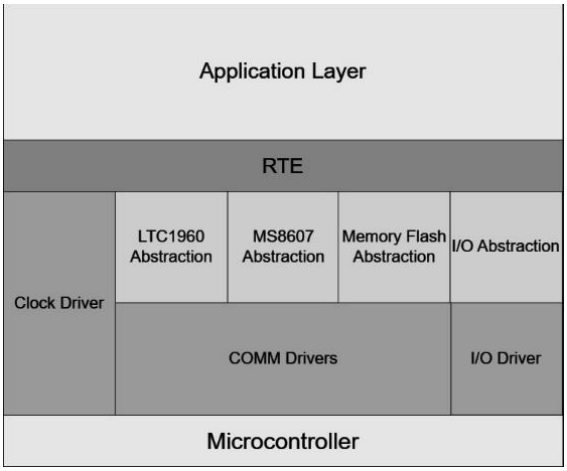

Fig. 2. EMIDSS-5 Software Architecture

- *Application Software (ASW)*

The module's application stores one set of meteorological data per minute in a structured format, where each set includes temperature, humidity, and pressure readings from the onboard sensor, the battery's charge level, and the time in which the sample was taken ("Fig. 3").

Bytes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Data	Hour	Min	Temperature				Pressure				Humidity				Batery's charge level			

Fig. 3. Data array structure.

The ASW algorithm flow diagram is shown in "Fig 4". The algorithm works as an operative system (OS) that runs the application which obtains the atmospheric data from the on-board sensor; then, the data is stored in the memory flash using the format shown in "Fig. 3". Lastly, the state of the user LED is toggled every second as an indicator that the ASW is working.

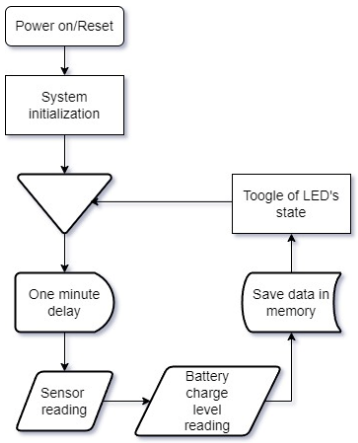

Fig. 4. Software flow diagram.

In addition, there is a user interface (UI) application feature that is activated when the MCU receives an UART command (see "TABLE I") at a 9600 baud rate in the LPUART1 pins (PTC6 and PTC7). The purpose of the UI is to configure and read the memory data without a Joint Test Action Group (JTAG). JTAG allows the user to have access to the MCU memory, so the UI can protect the intellectual property of EMIDSS-5 to avoid module hacking. The UI will permit communication with other MCUs through services, such as read data (service 22), write data (service 23), reset modules (service 11), and run routines (service 31). The UI logic is based on the Unified Diagnostic Services (UDS) standard, which facilitates communication with other MCUs to exchange data and request services. The commands are displayed in "Table 1".

TABLE I. USER INTERFACE COMMANDS

Reset service (11)			
Reset sensor S1101	Reset memory S1102	Empty S1103	Empty S1104
Read service (22)			
Read time S2201	Read all memory S2202	Read SW v. S2203	Empty S2204
Write service (23)			
Write time S2301	Empty S2302	Empty S2303	Empty S2304
Routine service (31)			
Re-init drivers S3201	Empty S3202	Empty S3103	Empty S3104

The last ASW feature is the battery management system (BMS) using the LTC1960 module. This module manages the battery charger and battery selector through SPI, allowing the MCU to control and monitor the status of the batteries. This feature improves energy consumption, allowing EMIDSS-V to prolong battery life after the last charge cycle.

Considering that batteries are affected by low temperatures, a heat resistor is placed near the batteries to control the temperature. The actual voltage data of each battery is provided by the LTC1960 module; thus, if the value is significantly low, the module will change the battery that is providing energy and start to charge the battery that has no voltage, executing this in a loop. The LTC1960 commands ("TABLE II") are sent via SPI, where D0 is Data0 and A0 is Address0.

TABLE II. LTC1960 SPI COMMANDS

2-Byte SPI Write Forma							
Byte1	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**
MOSI	D0	D1	D2	D3	D4	D5	D6
MISO	X	D0	D1	D2	D3	D4	D5
Byte2	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**
MOSI	D7	D8	D9	D10	A0	A1	A2
MISO	1	D7	D8	D9	D10	A0	A1
1-Byte SPI Write Forma							
Byte1	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**
MOSI	D0	D1	D2	X	A0	A1	A2
MISO	X	D0	D1	D2	X	A0	A1

- *Run Time Environment (RTE)*

The implementation of the RTE on EMIDSS-V ("Fig. 5") consists of SW APIs which enable the communication between the software modules (BSW and ASW), thus preventing data corruption and improving data robustness in SW modules. The RTE serves as a bridge between ASW and BSW. This is a key factor for the portability and modularization of the project.

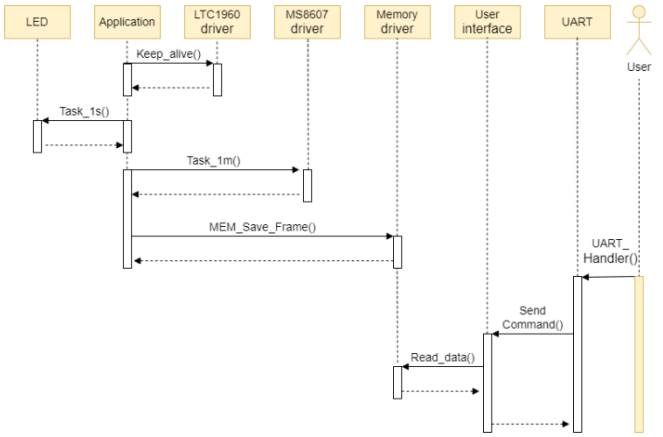

Fig. 5. EMIDSS-V Sequence Diagram.

- *Basic Software (BSW)*

BSW is known as the base/platform of the software. MCU configurations and integrated circuits (IC) are developed in BSW. The main purpose of the BSW in EMIDSS-V is to handle the required MCU configurations and provide the data to RTE, so ASW can consume this data. The components developed for the BSW are described below:

1. Clock driver – supports a timer configured to trigger a callback every millisecond. The clock driver is the base for the system pulse, required to store the data every minute and have a real-time counter (RTC).
2. COM – includes the required configuration of communication protocols such as I2C, UART, and SPI used to communicate with the external ICs.
3. I/O driver – manages the digital input/output of the MCU.
4. LTC1960 driver – contains the required configurations and interfaces to use this IC via SPI.
5. Memory flash driver – contains all the configurations and interfaces required to read/write and configure the memory flash.
6. MS8607 driver – contains all the configurations and interfaces to configure the MS8607 [7] and read the atmospheric data from the sensor.

III. RESULTS

The PCB shown in "Fig 6" was subsequently integrated into EMIDSS-5 "Fig 7".

Fig. 6. Final PCB.

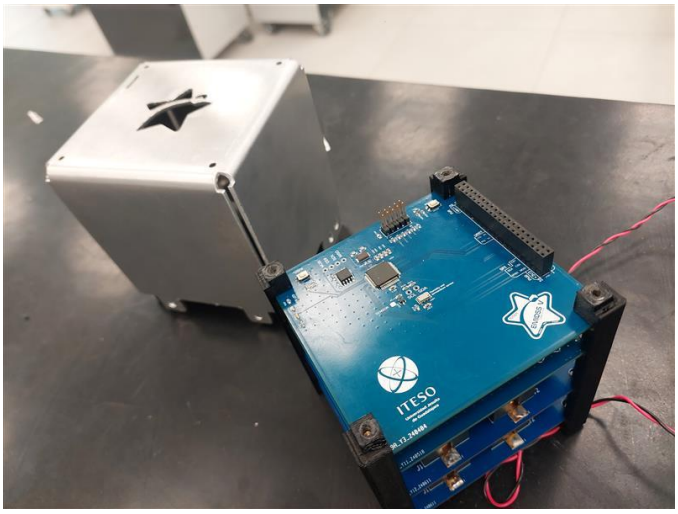

Fig. 7. EMIDSS-5.

Communication was established between the microcontroller and sensor with I2C as evidenced by the ACK bit reception shown in “Fig 8”.

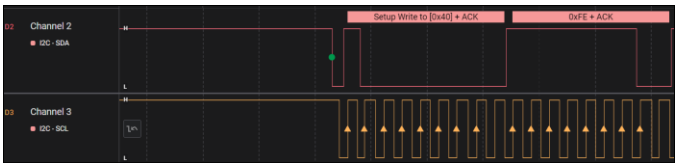

Fig. 8. MS8607 Sensor communication.

The values obtained from the sensor were compared with commercial products to evaluate the accuracy of the module MS8607 (TABLE III).

TABLE III. SENSORS MS8607 VS BME680 VS TH01

Device	Temperature	Humidity	Pressure
MS8607	25.77	51.68	847.52
BME680	27.24	48.63	847.99
TH01	27.2	51	-

Temperature, humidity and pressure sensor BME680 [8]. Temperature and humidity sensor TH01 [9].

The system that controls batteries was designed in a separate PCB from the microcontroller (Fig 9).

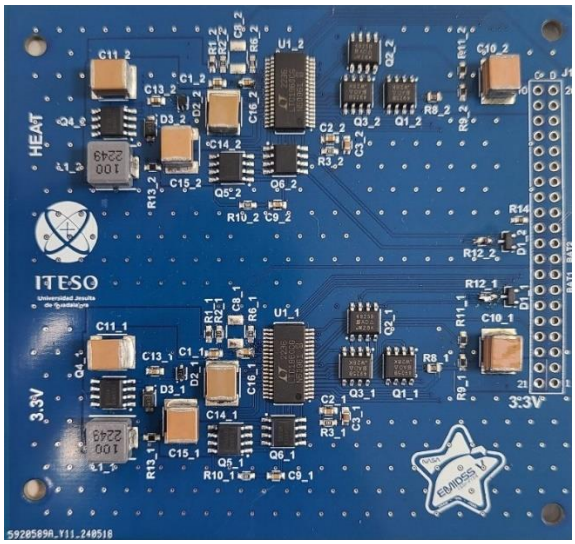

Fig. 9. LTC1960 PCB.

Machine-human communication was established by the user interface to request data from the EMIDSS-V as shown in “Fig. 10”.

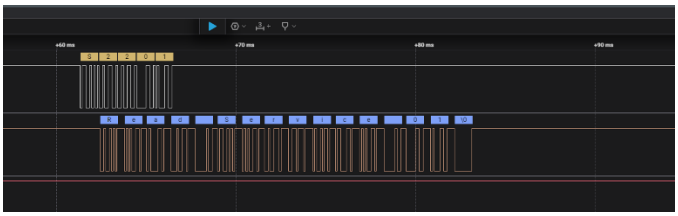

Fig. 10. User interface.

The memory communication was tested by reading its manufacturer and device identifier. As described in the memory’s datasheet [10], when the microcontroller sends the JEDEC-ID read command (0x9F), the memory should respond with the mentioned IDs, as shown in “TABLE IV”.

TABLE IV. MEMORY’S MANUFACTURER AND DEVICE IDS

Product	Manufacturer ID (byte 1)	Device ID	
		Device type (byte 2)	*Device ID (Byte 3)*
SST26VF032B/032BA	BFH	26H	42H

The test result is shown in “Fig. 11”. The memory’s response matches exactly the expected values.

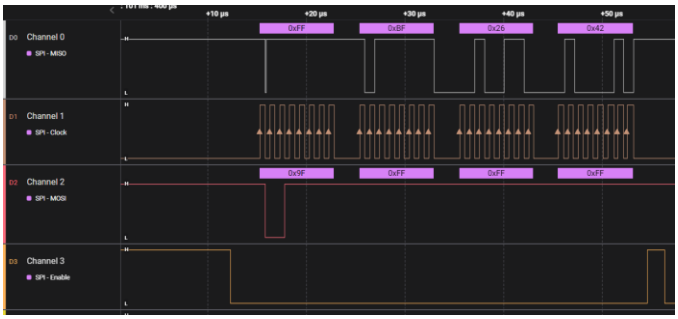

Fig. 11. Flash memory response to JEDEC-ID command.

IV. Conclusion and Future Work

After this fifth version, EMIDSS is now able to sense and save data more efficiently. Its user interface allows students and researchers to retrieve data, update the configuration, and debug EMIDSS more easily and safely. Lastly, its battery module makes it energy-independent, increasing the operational time.

The results from this work will be used for improvements in the next stage of the device. In future work, a low-energy state could be incorporated to save energy while the nanosatellite is not processing data. Also, an automated hardware testing framework can be created to increase the reliability of the code when new features are added. Finally, the incorporation of an RTOS can improve the application abstraction and resource management.

Acknowledgments

This work was supported by the Jalisco government through the SICyT. We would like to thank Dr. Luis Rizo for all the support given during the development of this project.

References

[1] G. Lastovicka-Medin, «Nano/PicoIFemto-Satellites: Review of Challenges in Space Education and Science Integration towards Disruptive Technology,» de *2016 5th Mediterranean Conference on Embedded Computing (MECO)*, Bar, 2016.

[2] M. Yaqoob, A. Lashab, J. C. Vasquez, J. M. Guerrero, M. E. Orchard y A. D. Bintoudi, «A Comprehensive Review on Small Satellite Microgrids,» *IEEE Transactions on Power Electronics*, vol. 37, nº 10, pp. 12741-12762, 2022.

[3] B. Peraza, E. Guzmán, J. Osuna y L. Rizo, «Adaptable On-Board Computer for Nanosatellites,» de *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, 2022.

[4] M. Amador, D. Platón, L. Flores, L. Hernández, J. Rivas, B. Romero y L. Rizo, «Layered software architecture for Nanosatellites,» de *DESI - Trabajos de fin de Especialidad en Sistemas Embebidos*, Tlaquepaque, 2023.

[5] NXP Semiconductors, «S32K1xx datasheet,» 2020. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/S32K1xx.pdf. [Last access: 15 June 2024].

[6] AUTOSAR, «AUTOSAR 4.4,» 2017. [Online]. Available: https://www.autosar.org/. [Last access: 15 June 2024].

[7] TE Connectivity, «MS8607-02BA01 datasheet,» 2017. [Online]. Available: https://www.te.com/en/product-CAT-BLPS0018.html?q=ms8607&source=header. [Last access: 23 June 2024].

[8] BOSCH, «BME6801 datasheet,» 2017. [Online]. Available: https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf. [Last access: 23 June 2024].

[9] Eujgoov, «TH01 Temperature and Humidity sensor,» 2024. [Online]. Available: https://www.zoominformatica.com/sensor-de-temperatura-wifi-tuya-smart-th01.html. [Last access: 23 June 2024].

[10] Microchip, «SST26VF032B/SST26VF032BA 2.5V/3.0V 32-Mbit Serial Quad I/O™ (SQI™) Flash Memory,» 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/20005218e.pdf. [Last access: 30 June 2024].