Research Article

Received: 21 July 2024 / Accepted: 27 October 2024 / Published: 05 November 2024

CDIO Methodology for the Development of a Heart Rate Meter as an Electronic Product of the Internet of Things

Johan J. Molina Mosquera[1,2]

Rodrigo Cadena Martínez[3]

Eduardo E. Loza Pacheco[4]

*[1]Electronic Engineering Department,
Faculty of Engineering,
Surcolombiana University,
Neiva, Huila, Colombia
[2]PhD student in Computer Science,
American University of Europe,
Cancun, Quintana Roo, Mexico
[3]PhD Department in Computer Science,
American University of Europe,
Cancun, Quintana Roo, Mexico
[4]Department of Applied Mathematics and Computing,
Faculty of Higher Studies "Acatlán",
Autonomous University of Mexico,
State of Mexico, Mexico*

DOI: https://doi.org/10.36941/ajis-2024-0176

Abstract

The academic article shows the use of the CDIO (Conceive, Design, Implement and Operate) methodology to develop a heart rate meter as an electronic product of the Internet of Things. Due to the outdated curriculum of the Electronic Engineering program of the Surcolombiana University, the lack of knowledge in electronic design issues with prototyping of circuits applied to IoT solutions and the lack of a clear reference methodology for the development of this type of products, it is considered necessary to know, propose and apply the stages of the CDIO initiative. This will allow sixth-semester students, who enroll in the course "Cloud Applications with Embedded Systems" belonging to the new curricular plan, and who will be the future Electronic Engineers of the Huila region, to acquire the technical and soft skills and competencies necessary to develop IoT products and solutions that are robust, scalable, secure, and tailor-made. With the design and use of a web application as an intuitive technological tool in the conception stage, it is easier to diagnose the context of the application, select devices and IoT platforms, determine technical characteristics and analyze costs based on the sizing of the IoT solution.

Keywords: *CDIO, IoT, Web application, Electronic product, Heart rate meter*

1. Introduction

The Electronic Engineering program of the Surcolombiana University of Neiva (Huila) is in the process of structuring its curriculum and improving methodologies, didactics, teaching-learning strategies and formative assessment, after applying a diagnostic instrument to a population of 327 enrolled students and in which a sample of 104 students (31.8%) responded. with a confidence level of 95% and a margin of error of 8%. The purpose of the survey was to collect information related to the "Development of Internet of Things IoT Projects". The analysis of the responses was as follows:

- 65.4% of the students in one of the courses of the academic program have learned about the IoT (Internet of Things) topic.
- 69.2% of students have not developed a course project on the topic of IoT
- 83.2% of students have not used a methodology to develop their IoT project as a reference
- 19.6% of students who completed a course project have had websites as a reference.
- 95.3% of students have not managed to turn their IoT project into a robust, scalable, secure and customized electronic product.

For these reasons, the research question is generated:

How do students of the Electronic Engineering program at Surcolombiana University acquire the skills to develop electronic products applied to the Internet of Things?

1.1 Internet of Things IoT

It is the latest technology to make life easier and more comfortable for humanity. It adds sensor-based technology to the strong network through which the world is connected to other networks and is one of the biggest booms in the IT industry. Many applications use sensor-based technologies that produce a large amount of data, such as date and time, reading values, and observations, increasing the veracity and volume of data leading to the large amount of data creation, but the cloud provider integrates with the IoT to sort the data as per the requirement. IoT refers to the interconnected network of everyday objects, which are often equipped with ubiquitous intelligence. IoT will increase the ubiquity of the internet by integrating each object through embedded systems, software, sensors, and actuators for interaction (Singh et al., 2019).

Relentless technological development, as well as the need for advanced functionalities, have led to the integration of complex systems (often conceived, developed and deployed as monoliths) into so-called SoS. Examples where complex systems have been converted into subsystems of a broader SoS to achieve the desired additional capabilities can be retrieved in many domains, such as transportation (integrated subway, road, and rail management systems), healthcare (integrated regional emergency systems and personal health devices), or energy (grid, housing and production/consumption systems). In general terms, SoS has five characteristics (Maier, 1998) of 1) operational independence: the subsystems have autonomous behavior, a purpose, and a useful existence, being able to operate even if they are separate from the SoS; 2) management independence: the subsystems are self-controlled and managed according to different policies; 3) evolutionary development: SoS evolves as subsystems evolve continuously and independently; 4) geographical distribution: SoS components can be distributed in several places but remain interconnected; and 5) emergent behavior: as dynamic and heterogeneous subsystems interact, new behaviors are introduced at the SoS level. While the first two features are mandatory, the other three are optional: in fact, the independence of constituent systems is a driving feature, regardless of their complex operation or geographic distribution. Likewise, IoT systems and their components are remarkably heterogeneous in terms of functionalities and technologies and, therefore, the demand for different degrees of autonomy and intelligence to be interoperable with human users or other IoT components.

1.2 Theoretical-methodological approach to study

The CDIO (Conceive-Design-Implement-Operate) methodology (Crawley et al., 2014; Liu et al., 2017) is an educational approach to engineering education that focuses on developing practical competencies through project creation. This methodology has been successfully applied in various fields of engineering, including the mechanical, electrical and software areas. In the field of the Internet of Things (IoT), the CDIO methodology allows the design and development of integrated electronic products, where solutions that respond to the needs of users are conceived, functional prototypes are designed, hardware and software systems are implemented, and scalable and secure solutions are operated. The CDIO approach is a key tool for the development of both technical and soft competencies in modern engineering, especially for IoT-related projects, facilitating active and practice-based learning.

Conceive: The conception phase of an IoT project involves identifying the needs and goals of users, as well as defining technical and business requirements. Techniques such as use case analysis and market research can be used to determine the features and functionalities that an IoT electronics product should have.

Design: In the design phase, models and prototypes are developed that allow the IoT solution to be visualized. In this phase, design and simulation tools are used to validate concepts and test different design options. Aspects such as system architecture, component selection, security, and data privacy must be considered.

Implement: The implementation phase focuses on building and testing the IoT solution. Software and hardware development tools are used to implement the solution, and tests are performed to verify the correct functioning of the system. Aspects such as interoperability between different components and devices, the ability to scale the solution, and power consumption must be considered.

Operate: In the operation phase, the deployment and management of the IoT solution is carried out. Aspects such as system configuration, maintenance, and upgrade, as well as monitoring and analysis of the data generated by the solution, must be considered.

In the update of the CDIO V2.0 syllabus in which the update of the objectives for engineering education is declared (Gustavsson & Stolterman, 2015) it is mentioned that Conception, Design, Implementation and operation of systems in the business, social and environmental context presents a vision of how the development of products or systems moves through these four metaphases. The terms chosen are descriptive of the hardware, software, and process industries.

According to the specific objectives, the table is built with the respective activities, which will serve as reference phases and methodological route for the development of an electronic IoT product:

Table 1: Objectives and activities for the CDIO methodological development

General Objective: Design the CDIO (Conceive, Design, Implement and Operate) methodology to develop robust, scalable, secure and customized IoT (Internet of Things) electronic products.				
ACTIVITIES IN STAGES OF CDIO METHODOLOGY				
Specific Objectives	Conceiving Stage	Design Stage	Implementation Stage	Operation Stage
Design the computer technology tool that allows diagnosing the context of the application, selecting the use of electronic devices and open source IoT platform, defining the development stages according to CDIO methodology and estimating the costs of the IoT electronic product.	Consult the state of the art of open source Internet of Things platforms. Select robust embedded systems that allow connection by Wi-Fi protocol. Select sensors and actuators that can be adapted to embedded systems.	Design a computer tool in accordance with the consultation of the platform and selection of embedded systems, sensors and actuators, carried out in the conception stage.	Implement the software tool as a web application for online consultation to show the recommendations of the IoT solution and cost estimation of the finished product, including the cost of using the IoT platform.	Verify connectivity to the software application and perform functionality tests and possible improvements in the technological tool.
Develop an electronic IoT product that complies with the stages of the CDIO methodology determined by the computer	Identify the problem that the IoT solution requires. Add the theoretical framework of the IoT platform	Design the schematic circuit that will connect with IoT platform Design the algorithm as	Implement the circuit that will be connected to the IoT platform on the breadboard. Implement the coding of the	Write electronic product operation manual

General Objective: Design the CDIO (Conceive, Design, Implement and Operate) methodology to develop robust, scalable, secure and customized IoT (Internet of Things) electronic products.				
ACTIVITIES IN STAGES OF CDIO METHODOLOGY				
Specific Objectives	Conceiving Stage	Design Stage	Implementation Stage	Operation Stage
tool.	and the hardware devices such as embedded system, sensor and actuator, used for the IoT solution	embedded system pseudocode that fits the IoT platform Design the PCB printed circuit board of the IoT electronic product	algorithm in the language required by the embedded system Assemble PCB components to obtain a robust and scalable product.	
Validate the use of the IoT electronic product.		Design the graphical user interface for machine-to-machine M2M interaction between mobile application and electronic product	Implement the graphical user interface depending on the functionalities of the selected IoT platform.	Verify connectivity of the electronic product with IoT platform Evaluate product usage and determine potential design improvements.
Development of Heart Rate Meter	Consultation of the computer technology tool "Conception for low-scale IoT solution" that allows you to select the MAX30102 sensor , ESP32 embedded system, LCD Display and Arduino IoT platform, with the respective data sheets, platform plans and estimated costs in Colombian pesos.	Design the electronic circuit diagram and connection and processing algorithm between ESP32 and Arduino IoT platform.	Perform functionality tests of the electronic circuit on the breadboard. Implement the coding in Arduino language of the ESP32 and interaction with IoT Platform through the graphical user interface. Assembly of components on a board with copper traces according to PCB generated from the Gerber file. Connectivity tests between assembled board and Arduino IoT platform.	Write the heart rate monitor's user manual. Evaluate the functionality and robustness of the heart rate meter's graphical user interface and printed circuit board. Verify the connectivity of the heart rate meter with Arduino's IoT platform.

1.3 *Diagram designs in UML*

The computer system of conception for a small-scale IoT solution is represented in UML design models (PlantUML, 2024, https://plantuml.com/) as a class diagram, sequence diagram, and use case diagram.

1.3.1 *Class Diagram Design*

To design the class diagram of the design system for a small-scale IoT solution, the main classes and their relationships are identified. Below is the diagram of classes and relationships.

Definition of Classes:

PlataformaIoT Class	**Sensor Class**	**SistemaEmbebido Class**
- plan: String	- name: String	- name: String
- cost: Float	- cost: Float	- cost: Float
- linkWeb: String	- linkDatasheet: String	- linkDatasheet: String

BaseDeDatos Class	**Actuador Class**
- Platforms: List<PlataformIoT>	- name: String
- sensores: List<Sensor>	- Cost: Float
- EmbeddedSystems: List<SistemaEmbebido>	- linkDatasheet: String
- Actuators: List<Actuador>	
- consultarPlataforma(plan: String): PlataformaIoT	
- consultarSensor(nombre: String): Sensor	
- consultarSistemaEmbebido(nombre: String): SistemaEmbebido	
- consultarActuador(nombre: String): Actuador	

InterfazUsuario Class

- seleccionarPlataformaIoT(): String
- seleccionarSensor(): String
- seleccionarSistemaEmbebido(): String
- seleccionarActuador(): String
- consultarBaseDeDatos(): void
- cargarAPIRest(): void
- mostrarCostoYEnlacePlataforma(platform: PlataformaIoT): void
- mostrarCostoYEnlaceSensor(sensor: Sensor): void
- mostrarCostoYEnlaceSistemaEmbebido(EmbeddedSystem: SistemaEmbebido): void
- mostrarCostoYEnlaceActuador(actuator: Actuador): void

Relations:

- The BaseDeDatos class has a composition relationship with the PlataformaIoT, Sensor, SistemaEmbebido and Actuador classes.
- The InterfazUsuario class interacts with the BaseDeDatos Class to query information and update the interface.

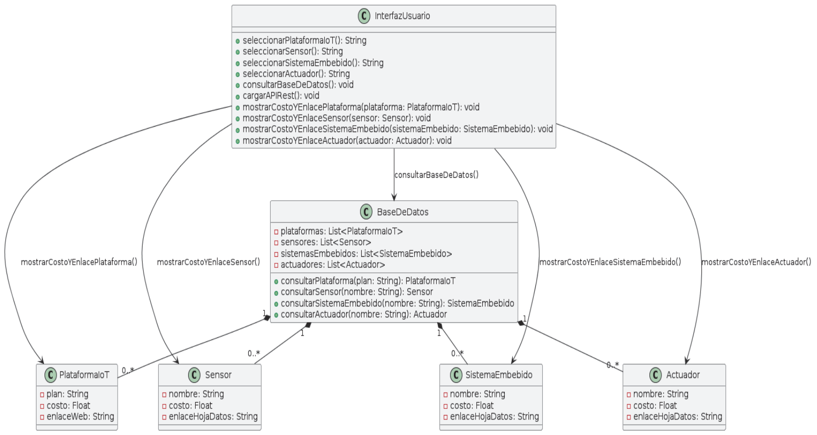
InterfazUsuario
seleccionarPlataformaIoT(): String
seleccionarSensor(): String
seleccionarSistemaEmbebido(): String
seleccionarActuador(): String
consultarBaseDeDatos(): void
cargarAPIRest(): void
mostrarCostoYEnlacePlataforma(plataforma: PlataformaIoT): void
mostrarCostoYEnlaceSensor(sensor: Sensor): void
mostrarCostoYEnlaceSistemaEmbebido(sistemaEmbebido: SistemaEmbebido): void
mostrarCostoYEnlaceActuador(actuador: Actuador): void
consultarBaseDeDatos()
BaseDeDatos
plataformas: List<PlataformaIoT>
sensores: List<Sensor>
sistemasEmbebidos: List<SistemaEmbebido>
actuadores: List<Actuador>
consultarPlataforma(plan: String): PlataformaIoT
consultarSensor(nombre: String): Sensor
consultarSistemaEmbebido(nombre: String): SistemaEmbebido
consultarActuador(nombre: String): Actuador
mostrarCostoYEnlacePlataforma()
mostrarCostoYEnlaceSensor()
mostrarCostoYEnlaceSistemaEmbebido()
mostrarCostoYEnlaceActuador()
1
0..*
PlataformaIoT
plan: String
costo: Float
enlaceWeb: String
Sensor
nombre: String
costo: Float
enlaceHojaDatos: String
SistemaEmbebido
nombre: String
costo: Float
enlaceHojaDatos: String
Actuador
nombre: String
costo: Float
enlaceHojaDatos: String

Figure 1: Class Diagram in UML

The figure of the Class Diagram shows that:

- Each class has its attributes defined as '-' (private) or '+' (public).
- Classes also have methods, such as ' consultarPlataforma ', ' consultarSensor ', etc.
- The relationship ‘BaseDeDatos’ "1" *-- "0..*" ‘PlataformaIoT ' indicates that ' BaseDeDatos ' has a composition relationship with ' PlataformaIoT ', which means that ' BaseDeDatos' contains multiple instances of ' PlataformaIoT '.
- Similar relationships apply to 'Sensor', ' SistemaEmbebido ', and 'Actuador'.
-'UserInterface' interacts with ' BaseDeDatos ' to consult information and with ' PlataformaIoT ', 'Sensor', ' SistemaEmbebido ', and 'Actuator' to display the information.

1.3.2 Sequence Diagram Design

In the design of the sequence diagram in UML, the following entities are used: Usuario, InterfazUsuario, BaseDeDatos, PlataformaIoT, Sensor, SistemaEmbebido and Actuador.

Process Description:

- The user selects an IoT platform, sensor, embedded system, and actuator in the user interface.

- The user clicks on the query button.
- The user interface queries the local database.
- The database returns the information about the selected IoT platform, sensor, embedded system, and actuator.
- The UI updates the columns with the cost information and links to the data.

The Sequence Diagram figure shows that:

- The **Usuario** selects the options of IoT platform, sensor, embedded system and actuator in the **InterfazUsuario**.
- The **Usuario** clicks on the query button.
- The **InterfazUsuario** queries the **BaseDeDatos**.
- The **BaseDeDatos** returns the corresponding information to the **InterfazUsuario**.
- The **InterfazUsuario** updates the interface with the information obtained.

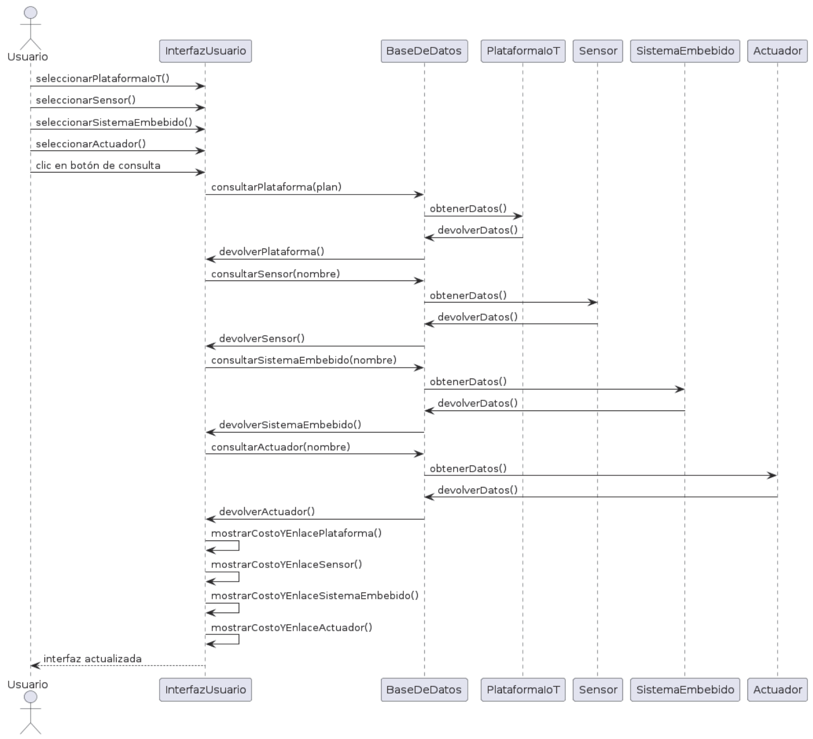

Figure 2: Sequence Diagram in UML

1.3.3 Design of the Use Case Diagram

For the mentioned system use case diagram, the actor and the main use cases are identified.

Actor:

- User: Interacts with the system to select and consult information on IoT platforms, sensors, embedded systems and actuators.

Use Cases:

Select IoT Platform, Select Sensor, Select Embedded System, Select Actuator, Query Database, Load REST API, Show IoT Platform Information, Show Sensor Information, Show Embedded System

Information, Show Actuator Information.

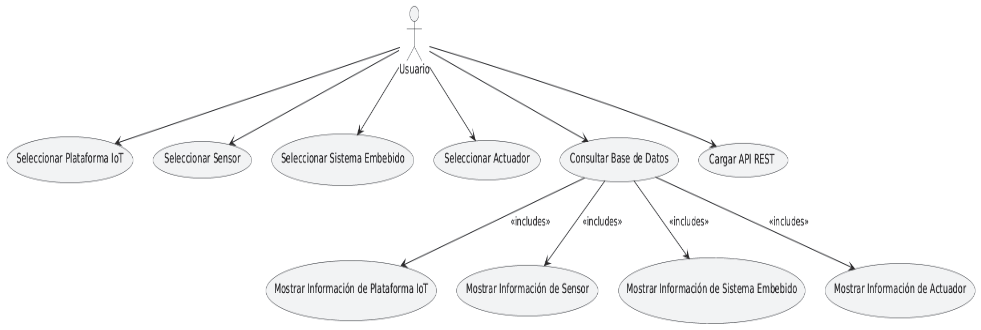
Usuario
Seleccionar Plataforma IoT
Seleccionar Sensor
Seleccionar Sistema Embebido
Seleccionar Actuador
Consultar Base de Datos
Cargar API REST
«includes»
«includes»
«includes»
«includes»
Mostrar Información de Plataforma IoT
Mostrar Información de Sensor
Mostrar Información de Sistema Embebido
Mostrar Información de Actuador

Figure 3 : UML Use Case Diagram

The figure in the Use Case Diagram shows that:

- 'User Actor' defines the user who interacts with the system.
- 'usecase' defines the use cases.
- 'Usuario → UC1' indicates that the user can select an IoT platform.
- 'UC5 → UC7 : <<includes>>' indicates that the "Query Database" use case includes "Mostrar Información de Plataforma IoT".
- The " Mostrar Información de Sensor ", " Mostrar Información de Sistema Embebido ", and " Mostrar Información de Actuador " use cases are similarly included.

This use case diagram shows how the user interacts with the system to perform various actions related to selecting and querying information about IoT components.

1.4 Web application connection model

The collaborative development platform "GitHub" offers hosting services and REST API. A responsive web application called " Concepción para solución de IoT a baja escala " is hosted on the hosting. Through the REST API, a database is accessed in JSON format, structured in key-value pairs and including objects such as platform, sensor, embedded system and actuator. This enables flexible data consumption and facilitates interface design.

The user, either from a mobile device or a computer connected to the internet, makes an HTTP request to GitHub hosting. In response, GitHub delivers the graphical user interface in HTML, enabling interaction between the user and the web application.

In the app, information on links and costs is displayed to access the datasheets of sensors, embedded systems and actuators, as well as the different plans of IoT platforms available. This information is obtained by querying the database in JSON format using the corresponding keys.

The connection to the REST API is made through the HTTP request, which responds by deploying the database in JSON format. This allows the exchange of data with another programming language such as Python.

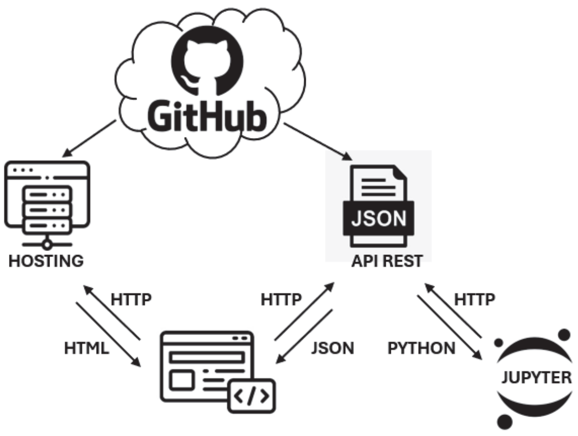

Figure 4 : Web Tool Connection Model

1.5 Description of the system designed in HTML5

The “Concepción para solución de IoT a baja escala” system shows a graphical user interface designed in HTML5 with the following components:

- Drop-down list to select the Internet of Things platform.
- Drop-down list to select the type of sensor.
- Drop-down list to select the embedded system.
- Drop-down list to select the type of actuator.
- Button to query the JSON database.
- Button to load the REST API.
- Data column to show cost information and access link to the selected IoT platform plans.
- Data column to display cost information and access link to the data sheet of the selected sensor.
- Data column to display cost information and access link to the data sheet of the selected embedded system.
- Data column to display cost information and access link to the data sheet of the selected actuator.

Buttons in the graphical user interface perform processes such as:

- Query button allows you to manage a local JSON database that has the following objects: platform, sensor, embedded system and actuator. The platform object defines the keys for the plan, cost and web-to-plan link for IoT. For sensor, embedded system and actuator objects; The keys for the name, cost and link to the data sheet of the devices such as sensor, embedded system and actuator are defined.
- The REST API load button manages the web query to the GitHub web server that stores the JSON database, so that the complete format is displayed with the platform, sensor, embedded system, and actuator objects, with their respective keys and values.

When selecting the drop-down lists of the type of IoT platform, type of sensor, type of embedded system and type of actuator. proceed to press the query button, it will query the local database type JSON and then update in the columns the information of the values obtained from the keys that are part of the platform, sensor, embedded system and actuator objects.

1.6 Responsive web application implementation

The graphical user interface called "Concepción para solución de IoT a baja escala" (Figure 5), shows how the overall design of an IoT electronic product is structured in which things are considered as sensors and/or actuators and edge devices as embedded system, means of connection, communication protocols, and cloud platform. Allowing the "Consulta" button to manage the query of JSON data locally and REST API service to the device datasheets, IoT platform plans and the respective cost estimate.

Here's how to arrange the interaction views with the graphical UI that is implemented in HTML5:

- Initial view when executing the URL request https://julmolina.github.io/iotherramienta/
- IoT Platform Selection View
- Sensor Selection View
- Embedded System Selection View
- Actuator Selection View
- Updated initial view of selected devices
- REST API JSON file view when executing the url request https://my-json-server.typicode.com/julmolina/IoT/db

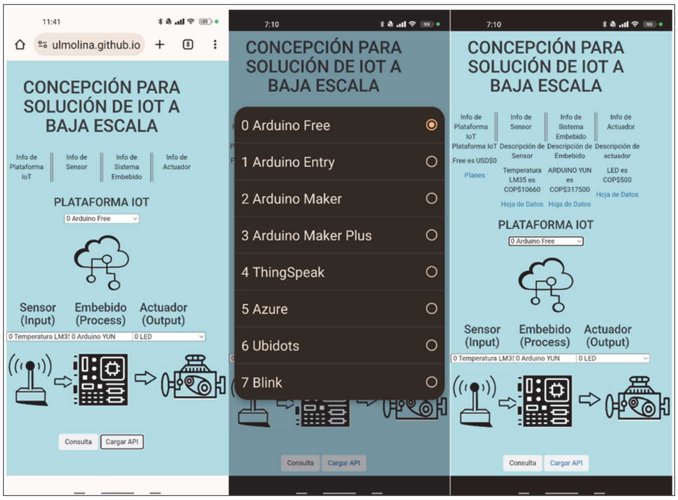

Figure 5: Home Web App; View Drop-down list; View Updated "Consulta"

The percentages of languages used for the development of the web application were: 29.4% HTML, 68.9% JavaScript and 1.7% CSS.

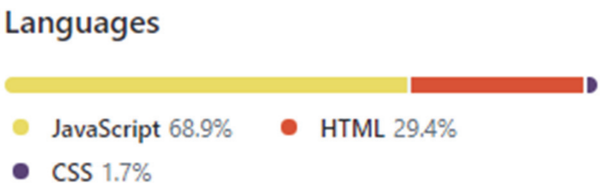

Figure 6: Web Application Languages (GitHub, n.d)

2. IoT Electronic Product

The IoT electronic product developed with the CDIO methodology is the "Heart Rate Monitor", and each methodological stage is described as follows:

2.1 *Conceiving Stage*

The web application "Concepción para solución de IoT a baja escala" is used to select and learn about technical characteristics, plans and costs of electronic devices (sensor, embedded system) and cloud platform that are part of the IoT solution.

The thing used as the main device for heart rate measurement is the MAX30102 sensor , it is an integrated module of heart rate monitor and pulse oximetry. It includes internal Leds (Red Led and Infrared Led), photodetectors, optical elements and low-noise electronics with ambient light rejection (Figure 7). The MAX30102 provides a complete system solution to facilitate the design process of portable and mobile devices. The MAX30102 is powered by a single 1.8V power supply and a separate 5V power supply. Communication is done through a standard I2C-compatible interface (ALLDATASHEET.COM,2023, https://pdf1.alldatasheet.com/datasheet-pdf/view/859400/MAXIM/MAX30102.html).

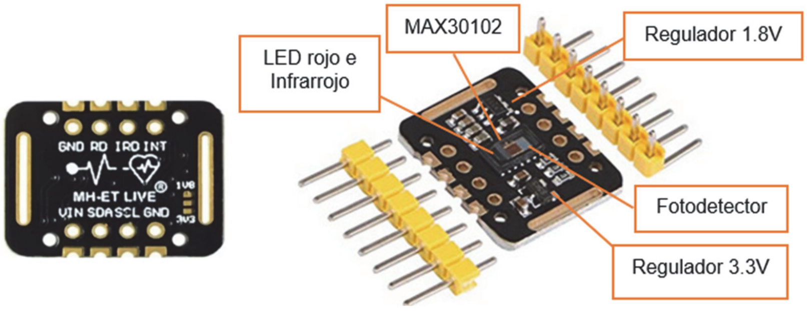

Figure 7: Pins and Parts of the MAX30102 sensor module (Molina et al., 2023)

Table 2 shows the parameters of the sensor module MAX30102:

Table 2: Sensor module parameters MAX30102

Parameters	**Value**
Supply Voltage	3.3V – 5V
Supply Current	0.6 mA – 1.2 Ma
Sample Rate (samples per sec)	50 sps – 3200 sps
Temperature Range	-40°C to +85°C
Temperature Accuracy	+- 1°C
ADC Resolution	18 bits (2^18 = 262144)
ADC Decimal Count of Red and IR LEDs	55536 – 75536
Maximum Wavelength Led IR	880 nm
Maximum Wavelength Red Led	660 nm

The MAX30102 sensor module also has the ability to operate in the following subsystems: SpO2 Oxygen Saturation Subsystem, Temperature Sensor Subsystem, Led Controller Subsystem, and Proximity Subsystem.

The I2C (Inter-Integrated Circuit) interface serves to facilitate communication between different electronic devices in a system, allowing them to exchange data and control operations efficiently (Figure 8). This interface is especially useful when it comes to connecting multiple components in an electronic system, as it uses only two signal lines (SCL and SDA) to transmit information between devices.

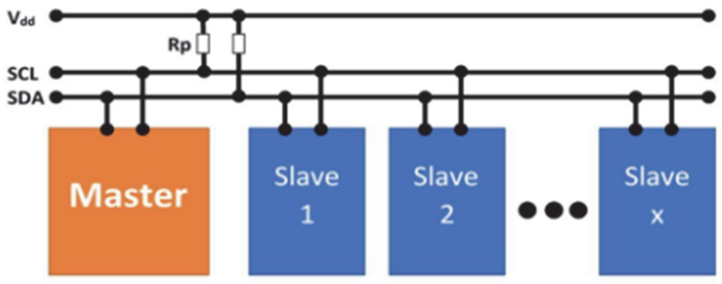

Figure 8: I2C Communication Interface

As an edge device, which performs the processing of input data from the MAX30102 sensor and transmits that data to the IoT platform, is the ESP32 card. This is a system on SoC (System on Chip) Chip that offers connectivity by WiFi and Bluetooth signals, with an operating frequency of 2.4 GHz, with computational performance due to CPU, memory, data input and output peripherals, RTC real-time clock, support for communications interfaces such as SPI, I2C, I2S; operating at low power and with hardware blocks used for security on a single chip (Bertoleti, 2019). Figure 9 shows the functional blocks of the ESP32 embedded system.

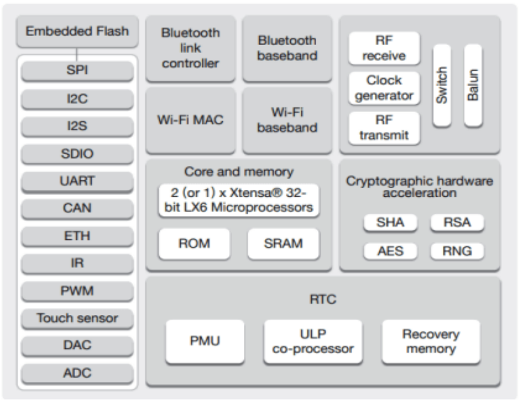

Figura 9: Bloques funcionales del Sistema embebido ESP32 (ESP32 Series Datasheet, 2023, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)

The cloud platform uses "Arduino IoT Cloud", which facilitates the creation, implementation, and monitoring of Internet of Things projects (Arduino Cloud, 2023, https://docs.arduino.cc/arduino-cloud/).

When entering the website (IoT Cloud Documentation, 2023, https://docs.arduino.cc/arduino-cloud/getting-started/IoT-cloud-getting-started) the necessary documentation is shown to start developing projects with different development environments and hardware compatible with the platform.

Subscription plans depend on the number of things connected, sketch storage, builds, interaction panels, time for data retention, LoRaWan connectivity, among others (Arduino Plans, 2023, https://digital-store.arduino.cc/subscriptions/plans).

2.2 *Design Stage*

In this stage, the block diagram of the electronic product "heart rate meter", the schematic circuit of the connections between data input-output peripherals and the ESP32 module, with the respective pseudocode, is designed.

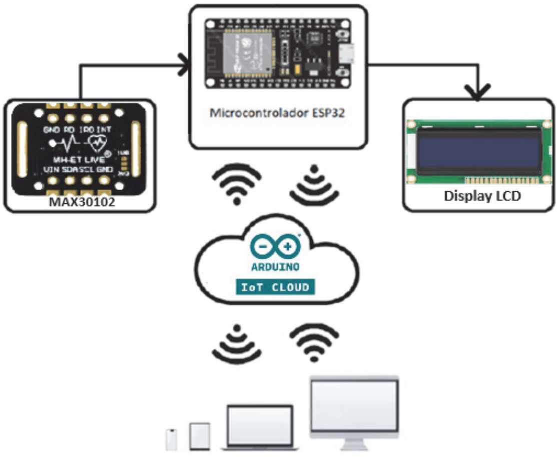

Figure 10: Heart rate meter block diagram

Figure 10 shows the connection of the sensor MAX30102 as an input peripheral to the ESP32 module, and the connection of the embedded system with the output peripheral, which would be an LCD display, which will indicate the BPM (Beats Per Minute) measurement according to the detection of the finger located on the surface of the photodetector.

The wifi chip that the module has will be connected to the IoT platform in the cloud known as "Arduino IoT CLOUD", so that from the graphical user interface built with its components, the heart rate measurement is displayed.

The circuit diagram of the connection between the ESP32 embedded system and the input peripheral (MAX30102 sensor) and output peripheral (LCD display), both peripherals connected by I2C interface, is shown in Figure 11.

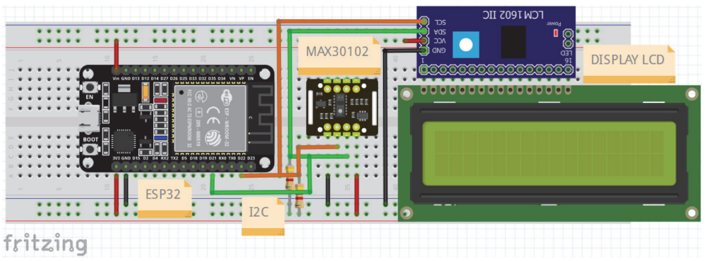

Figure 11: Heart rate meter circuit diagram

The ESP32 embedded system algorithm with the MAX30102 sensor module and LCD Display is presented as pseudocode in table 3 (Molina et al., 2023 ; IotStarters, 2020, https://iotstarters.com/esp32-based-max30100-pulse-oximeter-webserver/).

Table 3: ESP32 algorithm with MAX30102 sensor module and LCD Display

Algorithm ESP32_ritmo_cardiaco
Importing Libraries MAX30105, heartRate, spo2_algorithm, LiquidCrystal_I2C
Includes header file MAX30105.h, heartRate.h, spo2_algorithm.h, LiquidCrystal_I2C.h
Declaration of Cloud, float type variables for "heartbeats" and type CloudLocation for "location". Variables, such as byte for "rateSpot", long type for "lastBeat" and float type for "beatsPerMinute"
Definition of functions for: 1.Speed setting by serial monitor 2. Starting properties and connecting to the Arduino IoT platform 3. Cyclic execution of the algorithm 4. Function execution created for state change the variables "heartbeat" and "location".
Depending on the configuration, I2C communication between modules is Initialized MAX30102 sensor, LCD display and ESP32 embedded system (pin 22-SCL and pin 21-SDA). The particleSensor object of type MAX30105 is created.
Depending on cyclical execution, the connection to the platform is updated IoT and the process of reading the three bytes of data from the ADC converter of the sensor module MAX30102, this frame is the reading of the infrared LED pulses that are reflected off the photodetector. When checking for the presence of heartbeat, it is stored in the beatsPerMinute the arithmetic calculation 60/(delta/1000), to get the Heartbeat sample per minute. BPM value and location are displayed on the dashboard and LCD display georeferenced (latitude, longitude) of the heart rate meter. **If** (IR Value <1000) **Then**//No finger presence in MAX30102 sensor photodetector **Else** //If there is the presence of a finger on the MAX30102 sensor photodetector **End If**
The state change functions in the "heartbeat" and "location" variables they are empty.
End Algorithm

2.3 *Implementation Stage*

Based on the circuit diagram of connection between the ESP32 embedded system and the input peripheral (MAX30102 sensor) and data output peripheral (LCM1602 with LCD display), the PCB printed circuit board design (Figure 12) is carried out with the use of KiCad software (KiCad, 2024, https://www.kicad.org/). The ESP32 chip is programmed with Arduino language according to the algorithm proposed in the design stage, and finally the process of printing copper plate tracks and assembling components is carried out (Figure 13).

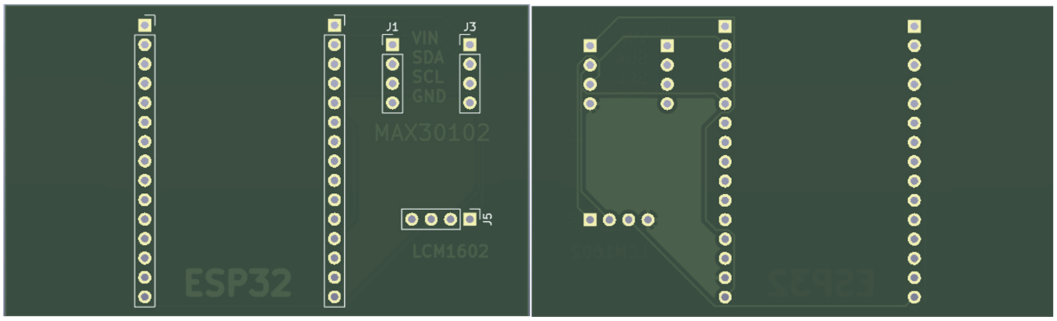

Figure 12: Front and Rear Views of PCB Printed Circuit Board

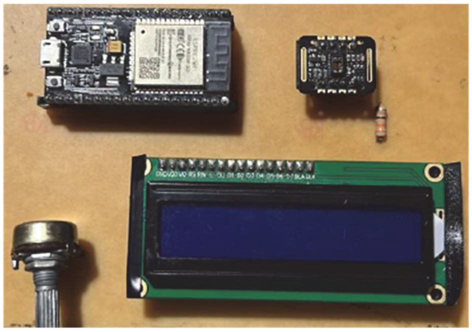

Figure 13: Printed Circuit Board on copper plate and component assembly

2.4 Operation Stage

For the validation of the operational stage of the design of the graphical user interface GUI generated from the Arduino IOT CLOUD platform, the M2M machine-to-machine interaction is carried out as shown in Figure 14 between the web application and the assembly circuit in Figure 13. In this interaction with the Value and Map component, the measurement of the heart rate measured in BPM is displayed when the finger is placed on the surface of the photodetector (sensor module MAX30102), and the georeferenced location of the device is also indicated.

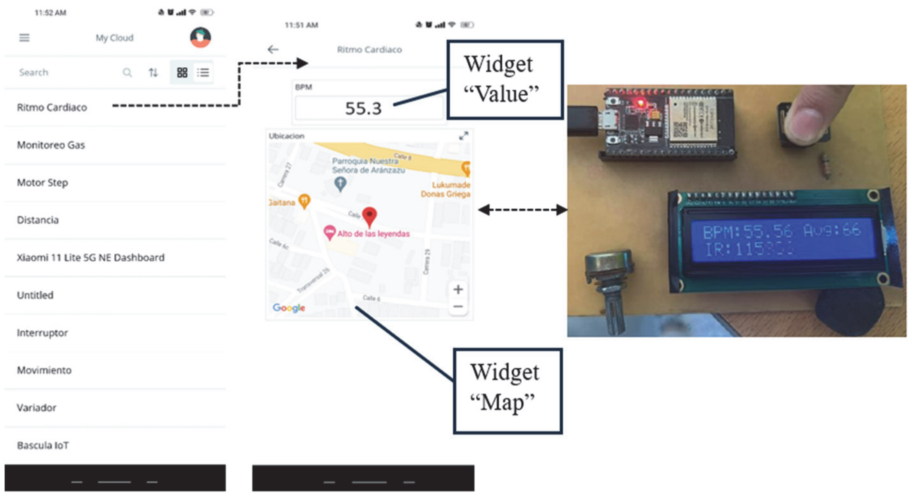

Figure 14: M2M interaction between App Dashboard and heart rate measurement circuit

3. Competence and Learning Outcomes

When developing an electronic IoT product for heart rate measurement. It will allow the integration of the CDIO methodology into the current curriculum of the Electronic Engineering program, which includes the learning outcomes and competencies acquired by the students who make up the work team. These are detailed in Table 4 (Molina et al., 2023).

Table 4: Competence and Learning Outcomes by Electronic Product Development

Competence	Learning Outcomes
Develops an IoT electronic product for heart rate measurement that connects to the Arduino IoT platform	Build and configure the circuit correctly with the MAX30102 sensor module, LCD Display and the ESP32 embedded system, connecting to the Wi-Fi network and the Internet of Things platform.
	It accurately interprets and codes the required algorithm in Arduino language, loading it into the ESP32 embedded system's program memory.
	He demonstrates prowess in the use of the Arduino IoT Cloud platform, creating things, associating variables of type floating , CloudLocation and String, and designing the M2M interaction dashboard with the widgets of type Value and Map.
	It robustly designs the printed circuit board (PCB) of the heart rate measurement system, ensuring its connection to the Arduino Internet of Things platform.

4. Conclusions

The IoT electronic product met the performance expectations, resulting in an effective and versatile solution for monitoring a person's heart rate. The ESP32 embedded system allows integration with other sensors, making the system scalable and enhancing its monitoring capabilities. For example, sensors that measure temperature, oxygen saturation or other body parameters can be added, thus obtaining a more complete view of the user's health status.

Students from the sixth semester of the Electronic Engineering program of the Universidad Surcolombiana acquire the necessary skills to develop IoT solutions following the CDIO methodology. This includes mastering various phases of development, from the use of software and hardware tools to the design of programming algorithms, PCB printed circuits, and graphical GUI user interfaces.

The development of electronic products based on the CDIO methodology for the Internet of Things (IoT) has proven to be an effective approach to the acquisition of practical engineering competencies. The use of this methodology in the design of the heart rate meter has allowed the students of the Electronic Engineering program to develop robust and scalable solutions, suitable for the real environment. This project not only meets the technical and design requirements raised, but also fosters comprehensive learning through the integration of theoretical and practical knowledge. The CDIO methodology has been key to training students in the creation of innovative electronic products, which contributes significantly to their training as engineers trained in emerging technologies.

For future research, cross-cutting areas such as the optimization of energy consumption, security and privacy in health systems and new pedagogical methodologies for IoT projects can be addressed, with the aim of further improving the acquisition of both technical and soft skills in electronic engineering students.

References

ALLDATASHEET.COM. (2023). MAX30102. Available in https://pdf1.alldatasheet.com/datasheet-pdf/view/859400/MAXIM/MAX30102.html

Arduino Cloud. (2023). Arduino Cloud. Available in https://docs.arduino.cc/arduino-cloud/

Arduino Plans. (2023). Arduino Plans. Available in https://digital-store.arduino.cc/subscriptions/plans

Bertoleti, P. (2019). Projects such as ESP32 and LoRa. Brazil: Editora NCB.

Crawley, E. F., Malmqvist, J., Östlund, S., & Brodeur, D. R. (2014). Rethinking engineering education: The CDIO approach. Springer.

ESP32 Series Datasheet. (2023). ESP32 datasheet. Retrieved from https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

GitHub. (n.d.). GitHub. https://github.com/

Gustavsson, I., & Stolterman, E. (2015). The CDIO syllabus: A statement of goals for undergraduate engineering education. International Journal of Engineering Education, 31 (4), 1074-1086.

IoT Cloud Documentation. (2023). IoT Cloud Documentation. Available in https://docs.arduino.cc/arduino-cloud/getting-started/IoT-cloud-getting-started

IotStarters. (2020, octubre 28). ESP32 based MAX30100 pulse oximeter webserver. IotStarters. https://iotstarters.com/esp32-based-max30100-pulse-oximeter-webserver/

KiCad. (n.d.). KiCad EDA: Schematic capture & PCB design software. Retrieved July 15, 2024, from https://www.kicad.org/

Liu, X., Li, G., & Wan, J. (2017). IoT-based smart rehabilitation system: A framework for ubiquitous exercise monitoring and real-time feedback. IEEE Internet of Things Journal, 4(1), 183-194.

Maier, M. W. (1998). Architecting principles for systems-of-systems. Systems Engineering, 1(4), 267-284.

Molina, J., Salgado, J., & Bravo, M. (2023). CIRCUITOS ELECTRÓNICOS CON ESP32 Y PLATAFORMA IOT DE ARDUINO. Editorial Académica Española.

PlantUML. (n.d.). PlantUML. PlantUML. Retrieved July 9, 2024, from https://plantuml.com/

Singh, S., Alam, P., Kumar, P., & Kaur, S. (2019). Internet of Things for Precision Agriculture Applications. En 2019 Fifth International Conference on Image Information Processing (ICIIP) (pp. 420-424). Shimla, India. https://doi.org/10.1109/ICIIP47207.2019.8985688