IoT Carrier

Modular Sensing Platform

HUBEX

Revision History

Version	Date	Description
v0.1	July 2026	Initial release

Abbreviations

Abbreviations	Explanation
ADC	Analog-to-Digital Converter
ADC1	Analog-to-Digital Converter 1
CO_2	Carbon Dioxide
CMOS	Complementary Metal-Oxide-Semiconductor
CP2102	USB-to-UART Bridge Controller
ESP32	Espressif 32-bit Microcontroller
GPIO	General-Purpose Input/Output
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
I/O	Input/Output
LED	Light Emitting Diode
LiPo	Lithium Polymer
NDIR	Non-Dispersive Infrared
PCB	Printed Circuit Board
Qwiic	Quick Wire Inter-Integrated Circuit
SCL	Serial Clock Line
SDA	Serial Data Line
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VIN	Voltage Input
V	Volt
mm	Millimeter
ppm	Parts Per Million
%RH	Percent Relative Humidity
dB	Decibel
dB SPL	Decibel Sound Pressure Level

Naming Conventions

Full name	Shorthand Reference
ESP32 NodeMCU-32S (ESP-32S)	NodeMCU-32S
ESP32-C3-DevKitM-1	C3-DevKitM-1
DEVKIT V1	DEVKIT V1

Table of Contents

Revision History....2
Abbreviations....3
Naming Conventions....3
1. Introduction....7
1.1 Functional Block Diagram....7
1.2 Board Information....8
1.3 IoT Carrier v0.1 Features....9
1.4 Fully Assembled Prototype....9
1.5 Programming Power Supply Note....10
2. Supported ESP32 Board Variants....11
2.1 NodeMCU-32S....11
2.1.1 Board Overview and Hardware Features....11
2.1.2 Operating Voltage and I/O Logic Level....12
2.1.3 Pin-Out Diagram....12
2.1.4 USB Programming Method....12
2.1.5 GPIO Naming Convention....13
2.1.6 Usable ADC pin....13
2.1.7 I²C Pins Used....13
2.1.8 Restricted and Special-Purpose Pins....13
2.1.9 Arduino IDE Board Configuration....13
2.2 C3-DevKitM-1....13
2.2.1 Board Overview and Hardware Features....13
2.2.2 Operating Voltage and I/O Logic Level....14
2.2.3 Pin-Out Diagram....14
2.2.4 USB Programming Method....15
2.2.5 GPIO Naming Convention....15
2.2.6 Usable ADC pin....15
2.2.7 I²C Pins Used....15
2.2.8 Restricted and Special-Purpose Pins....15
2.2.9 Arduino IDE Board Configuration....15
2.3 DEVKIT V1....15
2.3.1 Board Overview and Hardware Features....15
2.3.2 Operating Voltage and I/O Logic Level....16
2.3.3 Pin-Out Diagram....16
2.3.4 USB Programming Method....17
2.3.5 GPIO Naming Convention....17
2.3.6 Usable ADC pin....17
2.3.7 I²C Pins Used....17
2.3.8 Restricted and Special-Purpose Pins....17
2.3.9 Arduino IDE Board Configuration....17
2.4 Board Feature Comparison....17
3. Supported Sensor Variants....19
3.1 GP2Y1010AU0F Dust Sensor (analog / ADC)....19
3.1.1 Pin-Out Diagram....19
3.1.2 Communication Interface....19
3.1.3 ADC Connection....20
3.1.4 LED Control Pin....20
3.1.5 Measurement Timing....20

3.1.6 Analog Reading and Dust Concentration Calculation....20
3.1.7 Calibration....20
3.1.8 Troubleshooting and Safe Usage Notes....20
3.2 LMV324-Based Sound Detector (analog / ADC)....21
3.2.1 Pin-Out Diagram....21
3.2.2 Communication Interface....21
3.2.3 ADC Connection....22
3.2.4 Expected Input Voltage Range....22
3.2.5 Analog Reading....22
3.2.6 Calibration....22
3.2.7 Troubleshooting and Safe Usage Notes....23
3.3 Sensirion SCD30 Sensor (I²C)....23
3.3.1 Pin-Out Diagram....24
3.3.2 Communication Interface....24
3.3.3 Arduino Library....24
3.3.4 I²C and Sensor Initialization....25
3.3.5 Sensor Detection....25
3.3.6 Reading Sensor Measurements....25
3.3.7 Troubleshooting and Safe Usage Notes....26
4.Mechanical Information....27
5. Arduino Library and Example Code Research....28
5.1 GP2Y1010AU0F Dust Sensor....28
5.1.1 Required Libraries and Object Definitions....28
5.1.2 Important Pin Definitions....28
5.1.3 Important setup() Flow....28
5.1.4 Important loop() Flow....28
5.1.5 Data Acquisition....28
5.1.6 Data Processing....29
5.2 LMV324-Based Sound Detector....29
5.2.1 Required Libraries and Object Definitions....29
5.2.2 Important Pin Definitions....29
5.2.3 Important setup() Flow....29
5.2.4 Important loop() Flow....29
5.2.5 Data Acquisition....29
5.2.6 Data Processing....29
5.3 Sensirion SCD30 Sensor....29
5.3.1 Required Libraries and Object Definitions....29
5.3.2 Important Pin Definitions....30
5.3.3 Important setup() Flow....30
5.3.4 Important loop() Flow....30
5.3.5 Data Acquisition....30
5.3.6 Data Processing....30

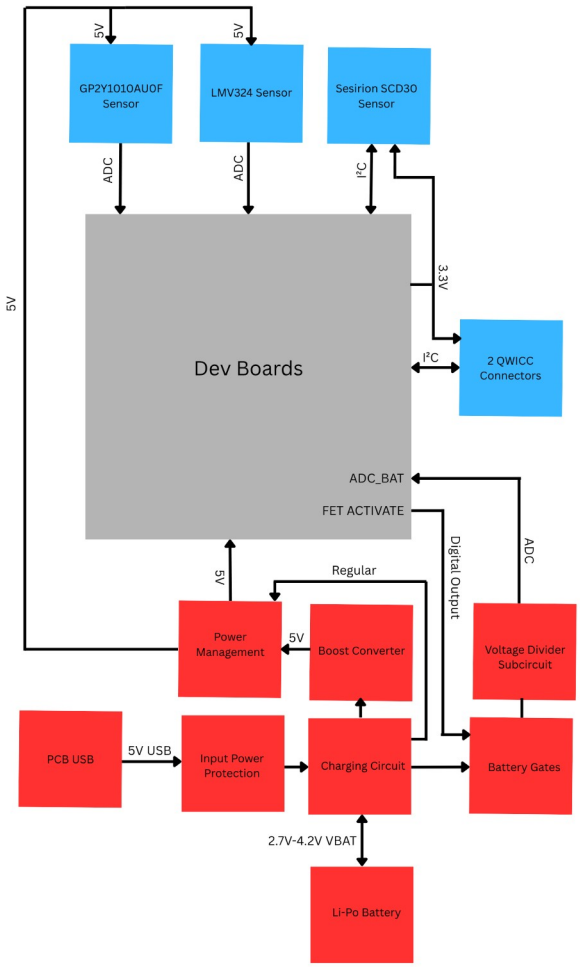
1. Introduction

IoT Carrier V0.1 is a carrier board designed to interface the NodeMCU-32S,C3-DevKitM-1, and DEVKIT V1 development boards with the peripherals implemented on the PCB. The board integrates dedicated interfaces for the GP2Y1010AU0F optical dust sensor, LMV324-based Sound Detector, and Sensirion SCD30 environmental sensor, together with two Qwiic-compatible I²C expansion connectors for additional peripherals.

The carrier board incorporates battery charging and power management circuitry, battery voltage monitoring, regulated power distribution, dedicated analog input channels, and shared I²C communication lines required by the supported sensors. Analog signals are routed to board-specific ADC-capable GPIOs, while the digital communication interface is mapped to the corresponding I²C pins of each supported ESP32 development board.

The carrier board architecture is developed to support the integration of the selected ESP32 development boards on the same platform while maintaining identical peripheral connectivity and electrical interfaces. This approach reduces hardware reconfiguration requirements, improves system flexibility, and provides a standardized environment for software development and testing of sensor-based IoT applications.

1.1 Functional Block Diagram



1.2 Board Information

The figures below identify the main components and connectors on the IoT Carrier V0.1 PCB. These figures provide a hardware overview and assist in locating the functional interfaces during prototype assembly and operation.

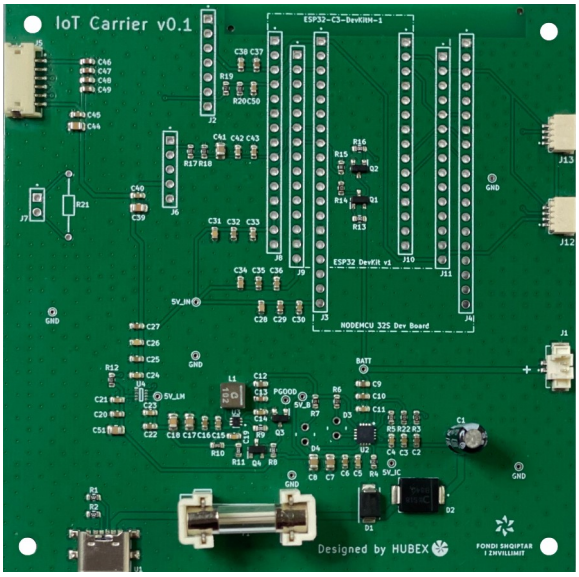


Figure 1.1: Board components - Front

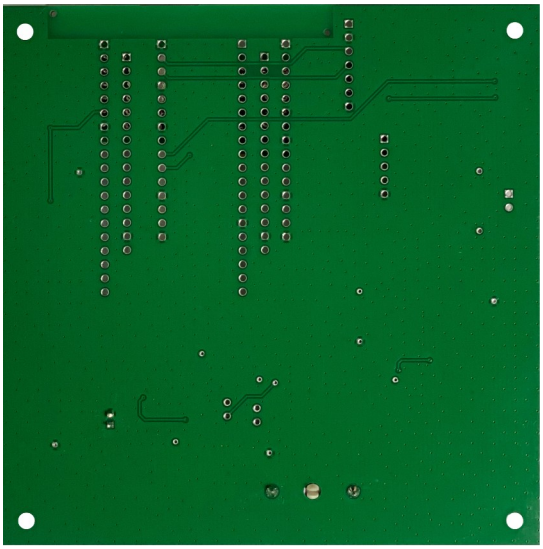


Figure 1.2: Board components - Back

1.3 IoT Carrier v0.1 Features

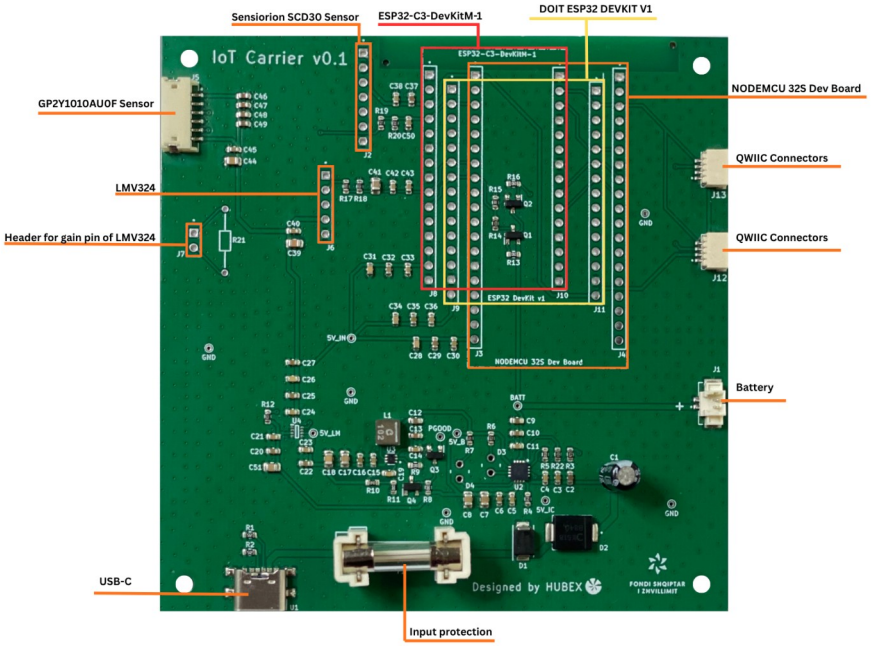


Figure 1.3: IoT Carrier v0.1 Features

1.4 Fully Assembled Prototype

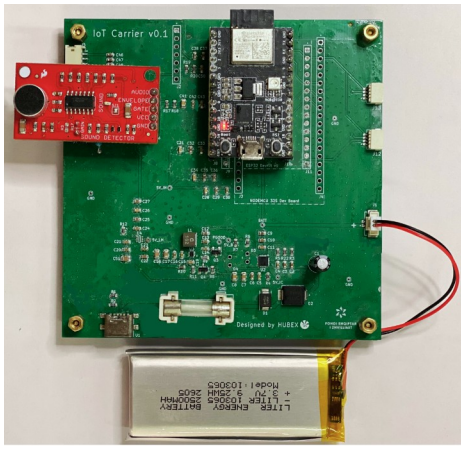


Figure 1.4: Fully assembled IoT Carrier V0.1 with the LMV324-based Sound Detector.

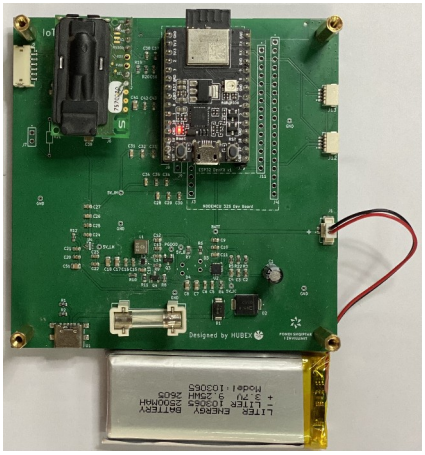


Figure 1.5: Fully assembled IoT Carrier V0.1 with the Sensirion SCD30 sensor.

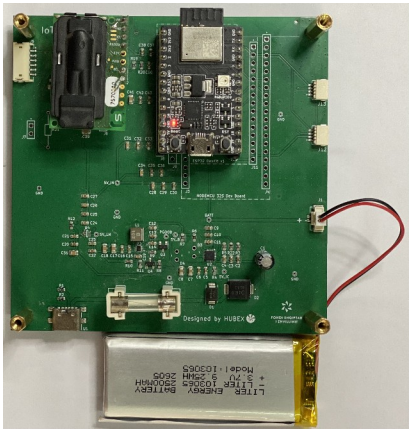


Figure 1.6: Fully assembled IoT Carrier V0.1 with the GP2Y1010AU0F dust sensor.

1.5 Programming Power Supply Note

During firmware programming, the ESP32 development board installed on the IoT Carrier V0.1 shall be powered only through its USB programming connector. The USB interface provides the required supply voltage for the development board and establishes the communication link necessary for firmware upload through the Arduino IDE.

The USB-C connector on the IoT Carrier PCB is reserved exclusively for power input. It does not support data communication and therefore cannot be used for firmware programming. All programming operations must be performed through the dedicated USB interface of the ESP32 development board.

Before initiating the programming process, the external power input of the IoT Carrier V0.1 must be disconnected. This includes removing the Li-Po battery, if present, and avoiding simultaneous power supply from both the carrier board and the USB connector. Supplying the board from two different sources at the same time may create unwanted current paths between the USB supply, battery circuit, and onboard voltage regulators. Such conditions can compromise programming reliability and may result in unstable operation.

The USB cable used for programming must support data communication. Charging-only USB cables cannot establish communication with the ESP32 and will prevent firmware upload. The correct ESP32 board profile and communication port must also be selected in the Arduino IDE before initiating the upload process.

2. Supported ESP32 Board Variants

IoT Carrier V0.1 is designed to support multiple ESP32 development boards through a common electrical interface and standardized pin headers. This standardized connector interface allows different ESP32 variants to be installed on the same carrier PCB without hardware redesign, ensuring compatibility across prototypes.

The platform supports three widely used boards:

- NodeMCU-32S (ESP-32S)
- C3-DevKitM-1
- DEVKIT V1

The NodeMCU-32S and DEVKIT V1 were selected because they are among the most commonly available ESP32 boards in the Albanian market. Their availability makes it easier to procure, replace, and maintain hardware during prototyping, ensuring that anyone can build and test prototypes without delays or sourcing difficulties.

The C3-DevKitM-1 was included to extend compatibility to the ESP32-C3 architecture, providing validation of the carrier board across different ESP32 families and enabling developers to test both Xtensa LX6 and RISC-V based implementations.

All supported boards are fully compatible with the analog (ADC) and digital (I^2C) peripherals implemented on the carrier PCB. The design enforces 3.3 V logic-level operation, meaning that both the ESP32 boards and any connected peripherals must operate at 3.3 V logic.

By standardizing power distribution, GPIO routing, and peripheral interfaces, the IoT Carrier V0.1 ensures that each supported board can connect to the same sensor circuitry and expansion headers while maintaining a consistent hardware architecture. This approach simplifies. It ensures that developers can swap boards easily, evaluate hardware, and validate firmware without modifying the carrier PCB.

2.1 NodeMCU-32S

2.1.1 Board Overview and Hardware Features

Feature	Description
Module	ESP32-WROOM-32
Processor	Dual-core Xtensa® LX6, up to 240 MHz
Memory	520 KB SRAM, 4 MB SPI Flash
Wireless	Wi-Fi 802.11 b/g/n (2.4 GHz), Bluetooth v4.2 + BLE
Interfaces	UART, SPI, I^2C, I^2S, PWM, timers, GPIO
ADC/DAC	12-bit SAR ADC (18 channels), 2 × 8-bit DAC
USB Programming	CP2102 USB-to-UART bridge

Logic Level	3.3 V CMOS
GPIO Count	34 (with restrictions)

2.1.2 Operating Voltage and I/O Logic Level

The development board is powered through a 5 V Micro-USB interface or the VIN pin. An onboard voltage regulator generates the 3.3 V supply required by the ESP32 module. All GPIO pins operate at 3.3 V CMOS logic levels and are not 5 V tolerant. The recommended operating voltage of the ESP32 module is 3.0–3.6V.

2.1.3 Pin-Out Diagram

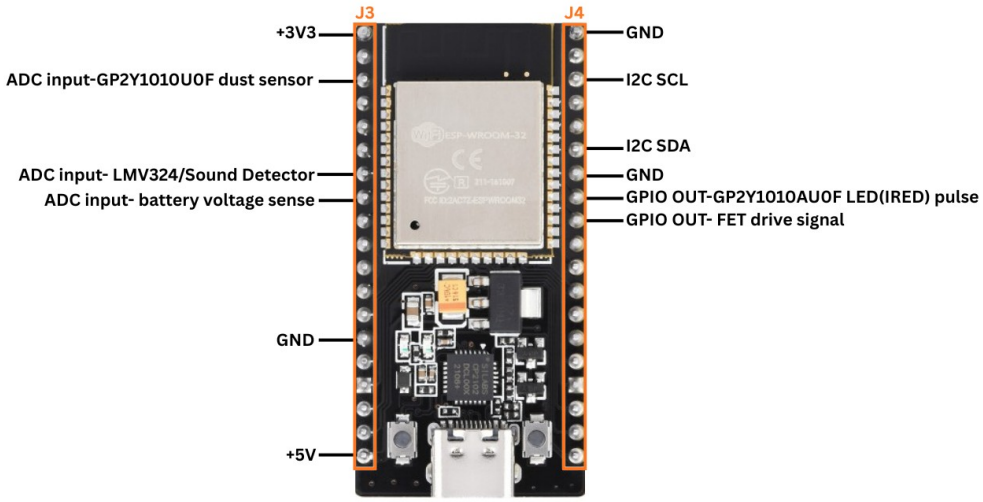


Figure 2.1:NodeMCU-32S pinout

Pin Name	GPIO / Pin	Function in Prototype	PCB Header
SDA	GPIO21	I²C data line	J4-06
SCL	GPIO22	I²C clock line	J4-03
ADC – dust	GPIO36 (ADC1_CH0)	GP2Y1010AU0F analog output Vo	J3-03
ADC – sound	GPIO32 (ADC1_CH4)	LMV324	J3-07
ADC – battery	GPIO33 (ADC1_CH5)	Battery voltage level detection	J3-08
GPIO out	GPIO19	GP2Y1010AU0F LED (IRED)	J4-08
GPIO out	GPIO18	FET signal	J4-09
3V3	3.3 V	Logic / sensor supply rail	J3-01
5V	5.0 V	Dust sensor / Sound Detector supply	J3-19
GND	GND	Common ground reference	J3-14;J4-01;J4-07

2.1.4 USB Programming Method

Firmware programming is performed through the onboard CP2102 USB-to-UART bridge, which communicates with the ESP32 using the UART0 interface.

2.1.5 GPIO Naming Convention

The board follows the standard ESP32 GPIO naming convention. Each GPIO is identified by its corresponding pin number, and the labels printed on the PCB match the identifiers used in software.

2.1.6 Usable ADC pin

ADC pins used by the prototype. The analog rails are routed to three ADC1 channels: GPIO36 (dust sensor output), GPIO32 (Sound Detector envelope), and GPIO33 (battery sense). ADC2 is not used, because it is shared with Wi-Fi and becomes unreliable while the radio is active.

2.1.7 I²C Pins Used

The prototype uses the default ESP32 I²C interface:

I²C Signal	GPIO
SDA	GPIO21
SCL	GPIO22

2.1.8 Restricted and Special-Purpose Pins

GPIO	Description
GPIO0	Bootstrapping pin
GPIO2	Bootstrapping pin
GPIO5	Bootstrapping pin
GPIO12	Bootstrapping pin
GPIO15	Bootstrapping pin
GPIO6-GPIO11	Reserved for onboard SPI Flash memory
GPIO34-GPIO39	Input-only GPIOs

2.1.9 Arduino IDE Board Configuration

Programming from the Arduino IDE requires installation of the Espressif ESP32 Boards package through the Boards Manager. The recommended board profile is NodeMCU-32S. If this profile is unavailable, ESP32 Dev Module provides equivalent functionality and pin compatibility.

2.2 C3-DevKitM-1

2.2.1 Board Overview and Hardware Features

Feature	Description
Module	ESP32-C3-MINI-1
Processor	Single-core RISC-V, up to 160 MHz
Memory	400 KB SRAM, 384 KB ROM, integrated SPI Flash
Wireless	Wi-Fi 802.11 b/g/n (2.4 GHz), BLE 5.0

Interfaces	UART, SPI, I²C, PWM, timers, GPIO
ADC/DAC	12-bit SAR ADC (up to 6 channels), no DAC
USB Programming	Integrated USB Serial/JTAG controller
GPIO Count	22
Logic Level	3.3 V CMOS

2.2.2 Operating Voltage and I/O Logic Level

The board can be powered through the Micro-USB connector, or the 3.3 V header pins. The ESP32-C3 operates at 3.3 V, and all GPIOs use 3.3 V CMOS logic levels. The GPIO pins are not 5 V tolerant, therefore all external peripherals connected directly to the board must also operate at 3.3 V logic levels.

2.2.3 Pin-Out Diagram

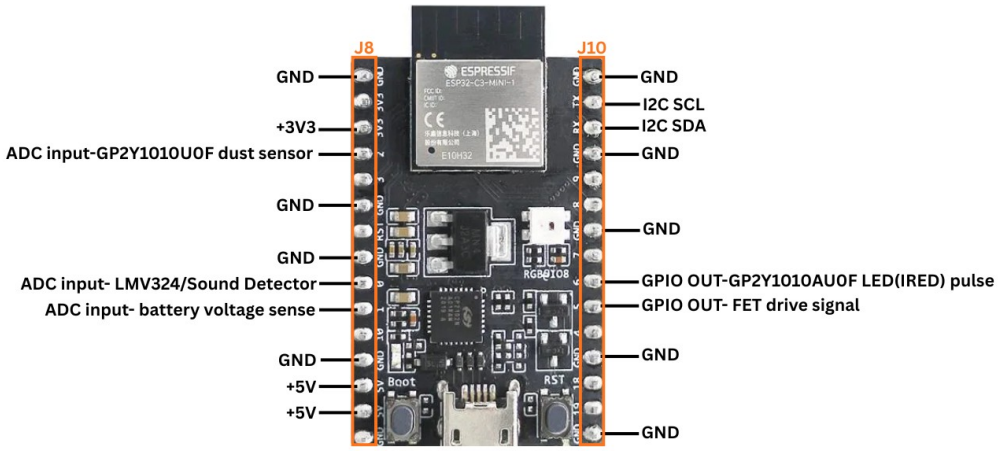


Figure 2.2: C3-DevKitM-1 pinout

Pin Name	GPIO / Pin	Function in Prototype	PCB Header
SDA	GPIO20	I²C data line	J10-03
SCL	GPIO21	I²C clock line	J10-02
ADC – dust	GPIO2 (ADC1_CH2)	GP2Y1010AU0F analog output Vo	J8-04
ADC – sound	GPIO0 (ADC0_CH0)	Sound Detector envelope output	J8-09
ADC – battery	GPIO1 (ADC1_CH1)	Battery voltage sense	J8-10
GPIO out	GPIO6	GP2Y1010AU0F LED (IRED) pulse	J10-09
GPIO out	GPIO5	FET drive signal	J10-10
3V3	3.3 V	Logic / sensor supply rail	J8-03
5V	5.0 V	Dust sensor / Sound Detector supply	J8-13, J8-14

GND	GND	Common ground reference	J8-01, J8-06, J8-08, J8-12, J8-15, J10-01, J10-04, J10-07, J10-12, J10-15

2.2.4 USB Programming Method

Firmware is uploaded through the onboard USB-to-UART interface using the Micro-USB connector. The USB interface provides both power and serial communication with the ESP32-C3 bootloader, allowing firmware to be programmed directly from the development environment.

2.2.5 GPIO Naming Convention

The board follows the standard ESP32-C3 GPIO naming convention. The GPIO labels printed on the PCB correspond directly to the GPIO identifiers used by the ESP32-C3 software libraries and development tools.

2.2.6 Usable ADC pin

ADC pins used by the prototype. The analog rails are routed to three ADC1 channels: GPIO2 (dust sensor output), GPIO1 (Sound Detector envelope), and GPIO0 (battery sense). ADC2 is not used.

2.2.7 I²C Pins Used

The ESP32-C3 supports flexible I²C pin assignment through the GPIO matrix.

I²C Signal	GPIO
SDA	GPIO21
SCL	GPIO20

2.2.8 Restricted and Special-Purpose Pins

The following GPIOs should be used carefully because they are sampled during device reset:

GPIO	Description
GPIO2	Bootstrapping pin
GPIO8	Bootstrapping pin
GPIO9	Bootstrapping pin

Incorrect logic levels on these pins during startup may prevent normal boot operation.

2.2.9 Arduino IDE Board Configuration

Programming with the Arduino IDE requires installation of the Espressif ESP32 Boards package through the Boards Manager. The recommended board profile for this development board is ESP32C3 Dev Module.

2.3 DEVKIT V1

2.3.1 Board Overview and Hardware Features

Feature	Description
Module	ESP32-WROOM-32
Processor	Dual-core Xtensa® LX6, up to 240 MHz
Memory	520 KB SRAM, 4 MB SPI Flash
Wireless	Wi-Fi 802.11 b/g/n (2.4 GHz), Bluetooth v4.2 + BLE
Interfaces	UART, SPI, I²C, I²S, PWM, timers, GPIO
ADC/DAC	12-bit SAR ADC (18 channels), 2 × 8-bit DAC
USB Programming	CP2102 or CH340 USB-to-UART bridge
GPIO Count	34 (with restrictions)
Logic Level	3.3 V CMOS

2.3.2 Operating Voltage and I/O Logic Level

The development board is powered through the Micro-USB connector or the VIN pin. An onboard voltage regulator provides the 3.3 V supply required by the ESP32 module. All GPIO pins operate at 3.3 V CMOS logic levels and are not 5 V tolerant. The recommended operating voltage of the ESP32 module is 3.0–3.6 V.

2.3.3 Pin-Out Diagram

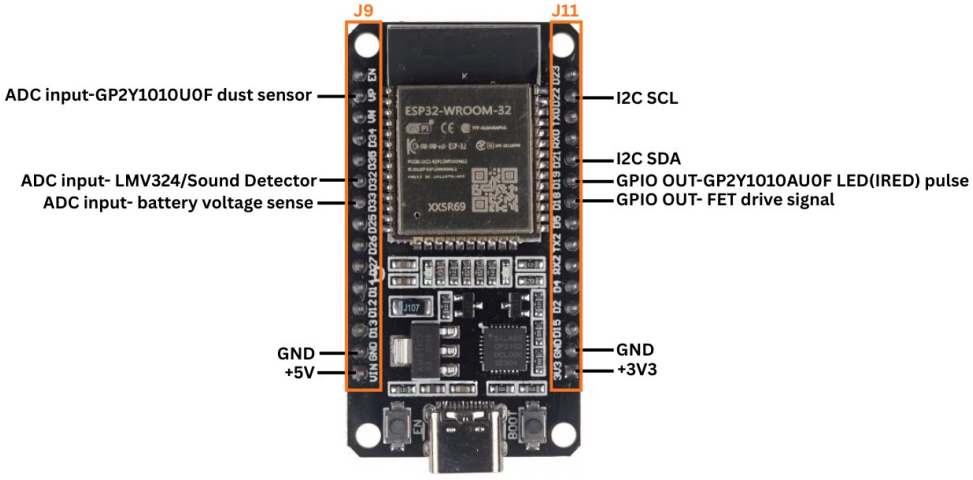


Figure 2.3: DEVKIT V1 pinout

Pin Name	GPIO / Pin	Function in Prototype	PCB Header
SDA	GPIO21	I²C data line	J11-05
SCL	GPIO22	I²C clock line	J11-02
ADC – dust	GPIO36 (ADC1_CH0)	GP2Y1010AU0F analog output Vo	J9-02
ADC – sound	GPIO32 (ADC1_CH4)	Sound Detector envelope output	J9-06
ADC – battery	GPIO33 (ADC1_CH5)	Battery voltage sense	J9-07
GPIO out	GPIO19	GP2Y1010AU0F LED (IRED) pulse	J11-06
GPIO out	GPIO18	FET drive signal	J11-07
3V3	3.3 V	Logic / sensor supply rail	J11-15
5V	5.0 V	Dust sensor / Sound Detector supply	J9-15
GND	GND	Common ground reference	J9-14, J11-14

2.3.4 USB Programming Method

Firmware is uploaded through the onboard USB-to-UART interface using the Micro-USB connector. The USB interface provides both power and serial communication with the ESP32-C3 bootloader, allowing firmware to be programmed directly from the development environment.

2.3.5 GPIO Naming Convention

The board follows the standard ESP32-C3 GPIO naming convention. The GPIO labels printed on the PCB correspond directly to the GPIO identifiers used by the ESP32-C3 software libraries and development tools.

2.3.6 Usable ADC pin

The analog rails are routed to three ADC1 channels: GPIO36 (dust sensor output), GPIO32 (Sound Detector envelope), and GPIO33 (battery sense). ADC2 is not used.

2.3.7 I²C Pins Used

The prototype implements the hardware I²C interface using the default ESP32 pin assignment:

I²C Signal	GPIO
SDA	GPIO21
SCL	GPIO22

2.3.8 Restricted and Special-Purpose Pins

GPIO	Description
GPIO0	Bootstrapping pin
GPIO2	Bootstrapping pin
GPIO12	Bootstrapping pin
GPIO15	Bootstrapping pin

GPIO6-GPIO11	Reserved for onboard SPI Flash memory
GPIO34-GPIO39	Input-only GPIOs

2.3.9 Arduino IDE Board Configuration

Programming with the Arduino IDE requires installation of the Espressif ESP32 Boards package through the Boards Manager. The recommended board profile for this development board is ESP32C3 Dev Module.

2.4 Board Feature Comparison

Feature	**NodeMCU-32S**	**C3-DevKitM-1**	**DEVKIT V1**
CPU	Dual-core Xtensa LX6, 240 MHz	Single-core RISC-V, 160 MHz	Dual-core Xtensa LX6, 240 MHz
Memory	520 KB SRAM, 4 MB Flash	400 KB SRAM, 384 KB ROM, SPI Flash	520 KB SRAM, 4 MB Flash
Wireless	Wi-Fi + BT v4.2 + BLE	Wi-Fi + BLE 5.0	Wi-Fi + BT v4.2 + BLE
USB Interface	CP2102 UART bridge	USB Serial/JTAG	CP2102 or CH340 UART bridge
GPIO Count	34	22	34
ADC/DAC	12-bit ADC (18 ch), 2 DAC	12-bit ADC (6 ch), no DAC	12-bit ADC (18 ch), 2 DAC

The comparison confirms that the IoT Carrier V0.1 is not dependent on a specific ESP32 development board. Although the supported boards offer different processing architectures and peripheral resources, each board is capable of supporting the implemented sensor interfaces and carrier board functions. This allows developers to select a board based on application requirements, availability, or cost, while maintaining the same hardware platform and peripheral connectivity.

3. Supported Sensor Variants

The platform reads two analog sensors through the ADC and one digital sensor over I²C.

3.1 GP2Y1010AU0F Dust Sensor (analog / ADC)

Sharp's GP2Y1010AU0F is an optical air quality sensor, designed to sense dust particles. An infrared emitting diode and a phototransistor are diagonally arranged into this device, to allow it to detect the reflected light of dust in air. It is especially effective in detecting very fine particles like cigarette smoke, and is commonly used in air purifier systems.

3.1.1 Pin-Out Diagram

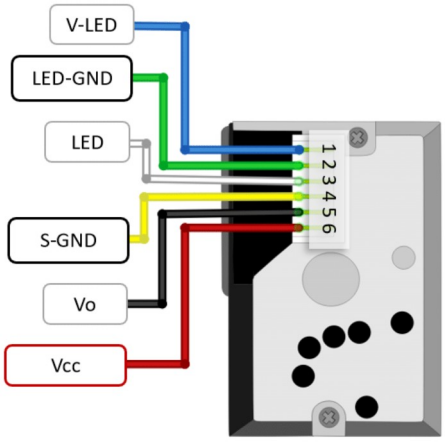


Figure 3.1: GP2Y1010AU0F Dust Sensor

Pin	Comments
V-LED	LED Power Supply.
LED-GND	LED Ground. Connect to the main circuit ground
LED	LED Control Pin. Connect to a digital output pin to toggle the IR LED
S-GND	Sensor Ground. Connect to the main circuit ground.
Vo	Sensor Analog Output. Connect to an analog input(ADC) to read dust levels
VCC	Supply Voltage

3.1.2 Communication Interface

The GP2Y1010AU0F does not use a digital communication protocol such as I²C, SPI, or UART. The sensor output is an analog voltage that is read by the ESP32 using an ADC-capable GPIO.

3.1.3 ADC Connection

The GP2Y1010AU0F dust sensor provides an analog output that is connected to the shared analog input rail of the IoT Carrier V0.1. The carrier board routes this signal to different ADC-capable GPIOs depending on the installed ESP32 development board.

Supported Development Board	GPIO	ADC
NodeMCU-32S	GPIO36	ADC1_CH0
C3-DevKitM-1	GPIO0	ADC1_CH0
DEVKIT V1	GPIO36	ADC1_CH0

3.1.4 LED Control Pin

The GP2Y1010AU0F requires an external GPIO to control its internal infrared LED. The LED is driven by the ESP32 before each measurement to ensure that the sensor output corresponds to the current dust concentration.

3.1.5 Measurement Timing

Before each measurement, the ESP32 drives the sensor LED control pin LOW to turn on the internal infrared LED. After a delay of 280 µs, the sensor output is sampled using the *analogRead()* function. The LED control pin is then driven HIGH to turn off the LED, and the measurement cycle is completed after an additional 9.68 ms delay, resulting in a total sampling period of approximately 10 ms.

3.1.6 Analog Reading and Dust Concentration Calculation

The value returned by *analogRead()* is converted to the corresponding sensor output voltage. After compensating for the ADC characteristics, the prototype calculates the dust concentration using the following equation:

$$Dust\ Density = \left(K_{SENSITIVITY} \times V_{sensor} \right) - BASELINE_{OFFSET}$$

- V_{sensor} is the corrected sensor output voltage;
- $K_{SENSITIVITY}$ is the calibration coefficient that converts the sensor voltage into dust concentration;
- *BASELINE_OFFSET* compensates for the sensor output measured in clean air.

3.1.7 Calibration

Calibration is recommended to improve measurement accuracy. Sensor output may vary due to component tolerances and environmental conditions; therefore, calibration using reference measurements is required for reliable dust concentration estimation.

3.1.8 Troubleshooting and Safe Usage Notes

If the dust sensor does not operate as expected, verify the following:

- Confirm that the sensor is supplied with the recommended operating voltage and that the power supply is stable. Variations or excessive noise on the supply may affect the analog output.
- Verify that the internal IRED is driven using the timing specified in the sensor datasheet. The analog output should be sampled during the LED-on measurement window.

- Confirm that the analog output is connected to an ADC-capable GPIO and that the correct ADC channel is selected in the firmware.
- If the measured dust concentration remains close to zero or does not change, inspect the sensor air inlet and outlet for dust accumulation or blockage.
- If the measured values are unstable, verify the grounding and analog signal routing. Keep the analog signal away from high-current or high-frequency switching circuits.
- Do not disassemble or modify the sensor, as this may affect the optical alignment and measurement accuracy.
- Do not expose the sensor to water, condensation, or corrosive gases.
- Avoid touching the optical chamber or allowing foreign particles to enter the air passage.
- Operate the sensor only within the electrical and environmental limits specified by Sharp.

3.2 LMV324-Based Sound Detector (analog / ADC)

The SparkFun Sound Detector is an analog sound sensing module based on an electret microphone and an LMV324 quad operational amplifier. The module detects variations in sound pressure and provides three independent outputs: Audio, Envelope, and Gate. For the PCB prototype, the Envelope output is used because it provides an analog voltage proportional to the detected sound amplitude.

3.2.1 Pin-Out Diagram

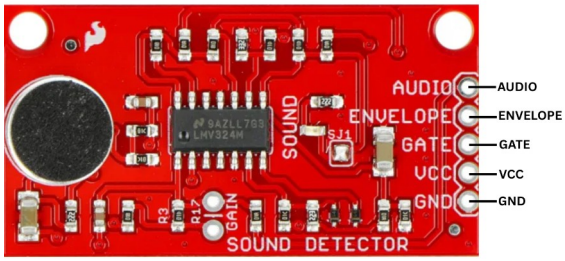


Figure 3.2: LMV324-Based Sound Detector

Pin	Comments
AUDIO	Raw analog output.
ENVELOPE	Analog output
GATE	Digital output.
VCC	Supply Voltage
GND	Ground

3.2.2 Communication Interface

The Sound Detector does not communicate through a digital protocol such as I²C, SPI, or UART. The sound level is represented as an analog voltage on the Envelope output, which is measured by the ESP32 using an ADC-capable GPIO.

3.2.3 ADC Connection

The analog output of the LMV324-based Sound Detector is routed through the shared analog input rail of the carrier board. Depending on the installed ESP32 development board, the signal is connected to the GPIOs listed below.

Supported Development Board	GPIO	ADC
NodeMCU-32S	GPIO32	ADC1_CH4
C3-DevKitM-1	GPIO1	ADC1_CH1
DEVKIT V1	GPIO32	ADC1_CH4

3.2.4 Expected Input Voltage Range

The Envelope output varies between approximately 0 V and the module supply voltage, depending on the detected sound level. When powered from 3.3 V, the output voltage remains within the 0–3.3 V ADC input range supported by the ESP32.

3.2.5 Analog Reading

The sound amplitude is sampled using the Arduino *analogRead()* function. Each ADC reading represents the instantaneous envelope voltage generated by the detector, where higher ADC values indicate higher sound intensity.

3.2.6 Calibration

The Sound Detector provides an analog output proportional to the detected sound amplitude and does not measure calibrated sound pressure levels (dB SPL). The output voltage represents relative sound intensity only. Applications requiring sound pressure measurements in decibels shall calibrate the detector against a reference sound level meter under controlled acoustic conditions to establish the relationship between the analog output voltage and the corresponding sound pressure level.

The sensitivity of the LMV324-based Sound Detector is determined by the preamplifier gain, which is configured by the feedback resistor network. The module is factory-configured for moderate sensitivity; however, the gain can be adjusted to suit different acoustic environments. Lower gain values reduce sensitivity and help prevent output saturation in high-noise conditions, while higher gain values improve the detection of low-level sound signals.

The available gain configurations are summarized in the following table.

Lower Gain Configuration

R21 Value	Arithmetic Gain	Gain (dB)
-	100	40
100K	50	33
47K	32	30
22K	18	25
10K	9	19

4.7K	4	13
2.2K	2	6

Raising Gain Configuration

R21 Value	Arithmetic Gain	Gain (dB)
100K	100	40
220K	220	46
470K	470	53
1Meg	1000	60

3.2.7 Troubleshooting and Safe Usage Notes

If the Sound Detector does not operate as expected, verify the following:

- Confirm that the supply voltage is stable. Noise on the power supply may appear directly on the analog output.
- If the analog output saturates near the maximum ADC value, reduce the amplifier gain until the output remains within the ADC input range.
- If the analog output remains very low, increase the amplifier gain or verify that the microphone is functioning correctly.
- Listen to the audio output, if available, to determine whether unwanted noise, vibration, or electrical interference is present.
- If a constant tone or oscillation is observed, verify the quality of the power supply and improve power filtering if necessary.

- Operate the detector within the recommended supply voltage range.
- Avoid placing the microphone near strong vibration sources or airflow, as these may introduce false measurements.
- Do not expose the microphone to moisture or excessive dust.
- Adjust the gain carefully to prevent amplifier saturation and loss of measurement accuracy.

3.3 Sensirion SCD30 Sensor (I²C)

The Sensirion SCD30 is a digital environmental sensor that measures carbon dioxide (CO_2) concentration, temperature, and relative humidity. The sensor is based on NDIR (Non-Dispersive Infrared) technology for CO_2 measurement and includes integrated temperature and humidity sensors for environmental compensation. It provides factory-calibrated digital measurements over an I²C interface.

3.3.1 Pin-Out Diagram

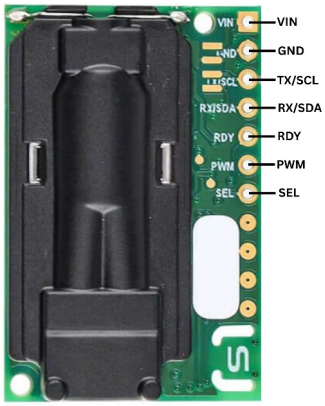


Figure 3.3: Sensirion SCD30 Sensor

Pin	Comments
VIN	Supply Voltage
GND	Ground
TX/SCL	Modbus: Transmission line (Push/Pull with 3V level) I^2C: Serial clock (internal 45kΩ pull-up resistor, pulled to 3V, for higher voltages a level shifter is needed)
RX/SDA	Modbus: Receive line (Input must not exceed 5.5V) I^2C: Serial data (internal 45kΩ pull-up resistor, pulled to 3V, for higher voltages a level shifter is needed)
RDY	Data ready pin. High when data is ready for read-out
PWM	PWM output of CO_2 concentration measurement
SEL	Interface select pin. Pull to VDD (do not exceed 4V, use voltage divider in case your VDD is >4V) for selecting Modbus, leave floating or connect to GND for selecting I^2C.

3.3.2 Communication Interface

The SCD30 communicates through the I²C interface. The IoT Carrier V0.1 routes the SDA and SCL signals to dedicated GPIOs on each supported ESP32 development board.

Supported Development Board	SDA	SCL
NodeMCU-32S	GPIO21	GPIO22
C3-DevKitM-1	GPIO20	GPIO21
DEVKIT V1	GPIO21	GPIO22

3.3.3 Arduino Library

Communication with the sensor is implemented using the following libraries :

```
#include <Wire.h>
#include <SensirionI2cScd30.h>
```

The Wire library provides the I²C communication interface, while the SensirionI2cScd30 library implements the commands required to configure the sensor and retrieve measurement data.

3.3.4 I²C and Sensor Initialization

```
Wire.begin(I2C_SDA_PIN, I2C_SCL_PIN);
```

Before starting normal operation, the firmware stops any previous measurement cycle, performs a software reset, and waits 2 seconds for the sensor to restart.

```
scd30_sensor.stopPeriodicMeasurement();
scd30_sensor.softReset();
delay(2000);
```

After initialization, the firmware reads the sensor firmware version to verify successful communication. The sampling interval is configured to 5 seconds using:

```
scd30_sensor.setMeasurementInterval(5);
```

Continuous measurements are then started with:

```
scd30_sensor.startPeriodicMeasurement(0);
```

where 0 indicates that no ambient pressure compensation is applied.

3.3.5 Sensor Detection

Successful communication with the sensor is verified during initialization by reading the firmware version:

```
scd30_sensor.readFirmwareVersion(major, minor);
```

If the function returns an error code, an error message is printed through the serial interface and the initialization process is terminated. During operation, all communication functions are also checked for errors, and the ESP32 automatically restarts if sensor initialization or configuration fails.

3.3.6 Reading Sensor Measurements

Before reading new measurements, the firmware checks whether data is available:

```
scd30_sensor.getDataReady(data_ready);
```

If *data_ready* equals 1, the latest measurements are retrieved using:

```
scd30_sensor.readMeasurementData(
  co2_concentration,
  temperature,
  humidity
);
```

The library returns the following calibrated values:

- CO_2 concentration in ppm
- Temperature in °C
- Relative humidity in %RH

These values are stored directly in floating-point variables and can be processed or transmitted without any additional conversion or ADC processing.

3.3.7 Troubleshooting and Safe Usage Notes

If the SCD30 sensor does not operate as expected, verify the following:

- Confirm that the sensor is powered correctly and that the SDA and SCL lines are connected to the appropriate I²C pins.
- If no measurements are available, allow the configured measurement interval to elapse before requesting new data.
- If communication errors occur, verify the I²C bus wiring, pull-up resistors, and sensor initialization sequence.
- If CO_2 measurements appear inaccurate, allow the sensor sufficient stabilization time and ensure adequate airflow around the sensing element.
- Operate the sensor only within the recommended supply voltage and environmental conditions.
- Do not block the air diffusion openings, as restricted airflow may affect measurement accuracy.
- Avoid exposing the sensor to liquids, condensation, or corrosive gases.
- Do not disconnect or reconnect the sensor while the I²C bus is powered.
- The SCD30 is intended for environmental monitoring applications and should not be used as a safety-critical or life-safety detection device.

4.Mechanical Information

All mechanical dimensions are expressed in millimeters (mm).

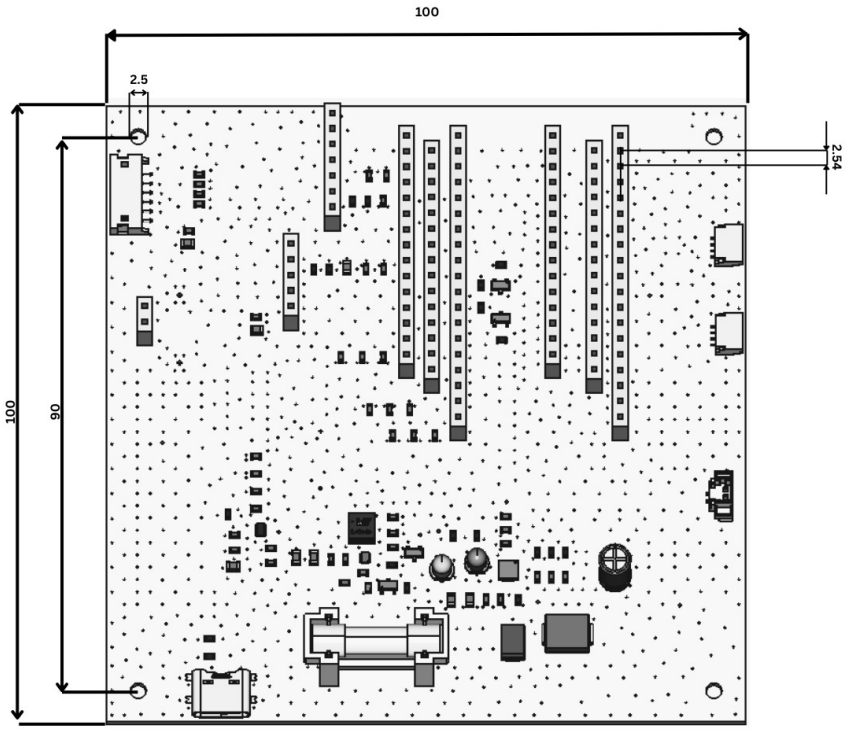


Figure 4.1: Top view of the IoT Carrier V0.1 PCB

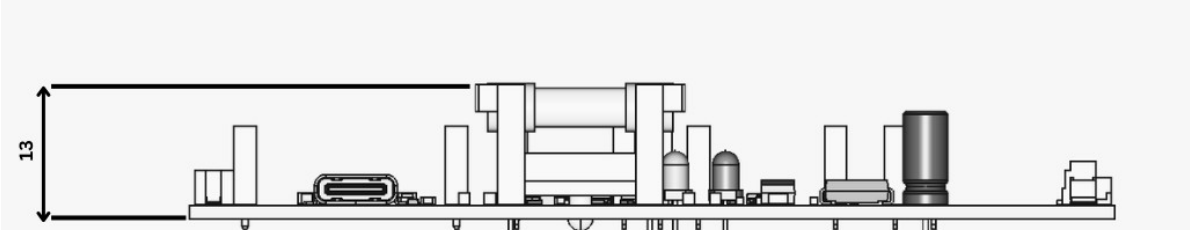


Figure 4.2: Side view of the IoT Carrier V0.1 PCB

5. Arduino Library and Example Code Research

The prototype firmware was developed using the Arduino IDE together with the ESP32 board support package and the corresponding sensor libraries. Each sensor requires a specific library to simplify hardware configuration, communication, and data acquisition. The following sections summarize the main software components and execution flow for each supported sensor.

5.1 GP2Y1010AU0F Dust Sensor

5.1.1 Required Libraries and Object Definitions

The GP2Y1010AU0F does not require a dedicated Arduino library because it provides an analog output. The firmware uses the standard Arduino library:

```
#include <Arduino.h>
```

The sensor is accessed directly through an ADC-capable GPIO and a GPIO used to control the internal infrared LED.

5.1.2 Important Pin Definitions

The firmware defines:

- One ADC pin connected to the sensor output.
- One digital output pin connected to the LED control input.

These pins are initialized during the setup phase before measurements begin.

5.1.3 Important setup() Flow

During initialization, the firmware configures:

- the LED control pin as an output;
- the ADC input pin for analog measurements.

The LED remains off until a measurement is requested.

5.1.4 Important loop() Flow

Each sampling cycle performs the following sequence:

1. Turn on the internal LED.
2. Wait for the required stabilization period.
3. Read the analog voltage using analogRead().
4. Turn off the LED.
5. Convert the ADC reading into dust concentration.
6. Print or store the calculated value.

5.1.5 Data Acquisition

The sensor output voltage is sampled using:

```
#include <Arduino.h>
```

The returned ADC value represents the optical signal generated by suspended dust particles.

5.1.6 Data Processing

The ADC reading is converted into sensor voltage and then into an estimated dust concentration using the calibration equation implemented in the firmware. The calculated value may be displayed on the Serial Monitor or transmitted to another system.

5.2 LMV324-Based Sound Detector

5.2.1 Required Libraries and Object Definitions

The Sound Detector also provides an analog output and therefore does not require a dedicated Arduino library.

```
#include <Arduino.h>
```

5.2.2 Important Pin Definitions

The firmware defines an ADC input connected to the module Envelope output.

5.2.3 Important setup() Flow

During initialization, the ADC input is configured and prepared for analog sampling.

5.2.4 Important loop() Flow

During each execution cycle the firmware:

1. Samples the analog output
2. Converts the ADC value into the corresponding voltage.
3. Determines the sound amplitude
4. Prints or stores the measured

5.2.5 Data Acquisition

Sound intensity is measured using :

```
analogRead()
```

which samples the analog voltage generated by the Sound Detector.

5.2.6 Data Processing

The measured ADC value is converted into voltage and used as an indication of the detected sound level. Since the module is not factory calibrated in dB SPL, the measured voltage represents relative sound intensity.

5.3 Sensirion SCD30 Sensor

5.3.1 Required Libraries and Object Definitions

The SCD30 requires the following Arduino libraries:

```
#include <Wire.h>
#include <SensirionI2cScd30.h>
```

The sensor object is created as:

```
SensirionI2cScd30 scd30_sensor;
```

5.3.2 Important Pin Definitions

The I²C interface is configured using:

```
#define I2C_SDA_PIN 20
#define I2C_SCL_PIN 21
```

These GPIOs provide the SDA and SCL communication lines between the ESP32 and the SCD30.

5.3.3 Important setup() Flow

The initialization procedure performs the following steps:

1. Initialize the I²C bus using Wire.begin().
2. Initialize the SCD30 library.
3. Stop any previous measurement cycle.
4. Perform a software reset.
5. Read the firmware version to verify communication.
6. Configure the measurement interval.
7. Start continuous measurement mode.

5.3.4 Important loop() Flow

During each execution cycle the firmware:

1. Checks whether a new measurement is available.
2. Reads CO_2, temperature, and humidity.
3. Prints the measured values to the Serial Monitor.
4. Repeats the process continuously.

5.3.5 Data Acquisition

The firmware first verifies that new data is available using:

```
scd30_sensor.getDataReady()
```

When data is ready, the measurements are retrieved using:

```
scd30_sensor.readMeasurementData()
```

5.3.6 Data Processing

The library returns calibrated values directly in engineering units:

- CO_2 (ppm)
- Temperature (°C)
- Relative Humidity (%RH)

No additional ADC conversion or mathematical processing is required before the values are displayed or transmitted.