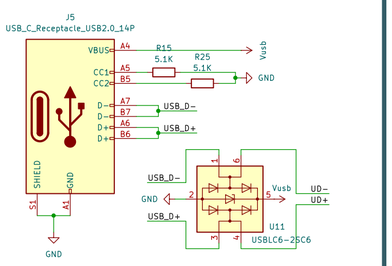
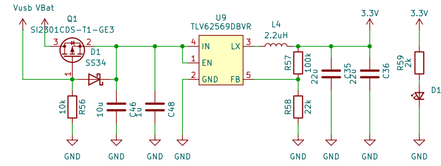
USB_type_c connector & esd protection



3.3 V Power Supply and Power-Path Management



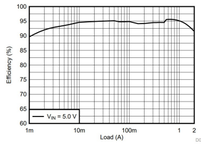
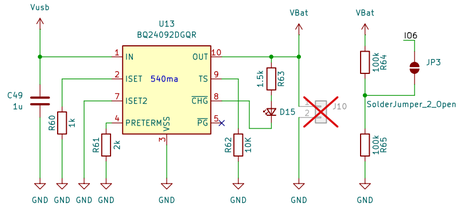


Figure 9. 3.3-V Output Efficiency

$$V_{OUT} = V_{FB} \times \left(1 + \frac{R1}{R2}\right) = 0.6V \times \left(1 + \frac{R1}{R2}\right)$$

Elyssa uses a TLV62569 synchronous buck converter to efficiently generate the main 3.3 V rail. A PMOS + Schottky diode power-ORing stage provides input protection and prevents reverse current between power sources. The 2.2 µH inductor and ceramic capacitors ensure stable, low-ripple operation, while the feedback network sets the output precisely to 3.3 V.

1S Li-Po Battery Charger



Elyssa uses the BQ24092 single-cell Li-ion/Li-Po charger, powered from 5v USB. The charging current is programmed to approximately 540 mA using the 1 kΩ ISET resistor. The charger regulates the battery to 4.2 V, making it suitable for a 1S 3.7 V nominal Li-ion/Li-Po battery.

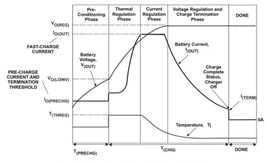
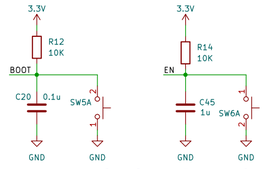


Figure 9-1. Charging Profile with Thermal Regulation

A 1S 1500–2000 mAh Li-Po is recommended for a good balance between charging time and battery stress. The IC may become noticeably warm during charging and can reach around 125 °C,where thermal regulation limits the charging current. A solder bridge can be enabled to monitor the battery voltage through a 1/2 voltage divider connected to IO6.

boot & reset buttons



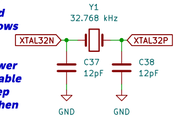
Difference & How to Use Them for BootingNormal Boot: When you power on the board or press the EN button while BOOT is unpressed (held HIGH by the pull-up resistor) ,the MCU boots normally from its internal flash memory.

Firmware Download Mode:To upload new code manually, press and hold the BOOT button, press and release the EN button, and then release the BOOT button. This pulls GPIO0 LOW during reset,instructing the ESP32-S3 bootloader to enter serial flashing mode.

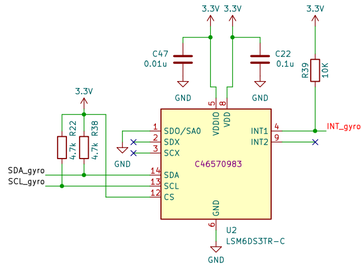
Note on GPIO Usage: While the BOOT .button is primarily used for entering download mode,the pin can also be configured and used as a normal digital input(GPIO)in your firmware after the boot process is complete.

Deep_Sleep_Clock

The external 32.768 kHz crystal connected to XTAL32N and XTAL32P provides an accurate,low-frequency clock for RTC timekeeping and timed wake-up during deep sleep. This allows Elyssa, powered by its 3.7 V LiPo battery, to maintain time and wake upperiodically while minimizing power consumption. The ESP32-S3 can disable most of its circuitry during deep sleep and use the RTC timer to wake up when needed, extending battery life. For maximum efficiency, unnecessary peripherals and power-consuming circuits should also be disabled before entering deep sleep.



LSM6DS3TR-C - 6-Axis IMU



The onboard **LSM6DS3TR-C** combines a 3-axis accelerometer and a 3-axis gyroscope, enabling motion sensing, orientation estimation, and gesture recognition. It communicates with the ESP32-S3 over I²C using **GPIO8 (SDA)** and **GPIO9 (SCL)**, while **GPIO21** is connected to its interrupt output for event-driven sensing.

***Functions available to users:**

- **Motion measurement:** Read acceleration along X, Y, and Z axes, angular velocity, and sensor temperature.*
- **Orientation:** Estimate pitch and roll, and track rotational movement.*
- **Activity recognition:** Detect steps, single and double taps, free fall, tilt, and significant motion.*
- **Low-power operation:** Configure motion detection to wake the ESP32-S3 from deep sleep, enabling battery-powered applications.*
- **Advanced processing:** The `ElyssaIMU` library provides FIFO-based data recording, digital filtering, calibration, self-test, stillness detection, timestamps, wrist-tilt recognition, and interrupt callbacks.*

***Example applications:** Robot motion tracking, activity monitoring, gesture-controlled interfaces, wearable devices, and motion-triggered systems.*

***Getting started:** Basic IMU functions are integrated into the Elyssa board package. For advanced features, install or include the `ElyssaIMU` library using `#include <ElyssaIMU.h>` and explore the available examples.*

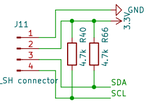
Documentation and source code: ***for ElyssaIMU library and examples***

IMU axes are labeled on the dev board

I2C Bus & Qwiic Connector

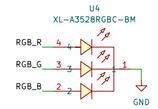
Elyssa dev board features a Qwiic connector wired in parallel with the exposed pin headers on GPIO 8 and 9 (SDA/SCL), complete with onboard 4.7kOhm pull-up resistors.

Connection Options: You can plug Qwiic-compatible sensors directly into the onboard JST connector or wire your I2C devices manually to the corresponding SDA and SCL pin headers.

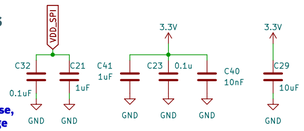


On Board RGB Led

The onboard RGB LED provides visual status feedback and is controlled by the ESP32-S3 through GPIO39 (Red), GPIO38 (Green), and GPIO33 (Blue).



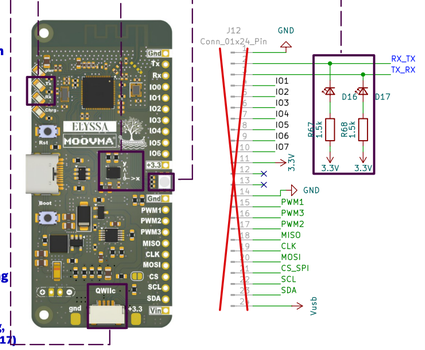
Decoupling caps



Capacitors placed across the power rails reduce noise, stabilize the supply voltage, and provide local charge during rapid current changes.

boot & reset buttons

Elyssa development board is built around the ESP32-S3FN8 MCU and features a compact 22-pin header along with two dedicated power pads (GND and VCC) to solder for upcoming project expansions. The board is designed for seamless prototyping, allowing easy, direct mounting and use with standard breadboards.



1. Pin Header & Breadboard Integration

Breadboard Friendly: The layout of the 22-pin header allows the board to plug directly into standard breadboards for fast prototyping.

Solder Pads: Includes dedicated GND and VCC pads on the edge for permanent project wiring and extensions coming in upcoming releases.

2. Communication Protocols & Specific GPIO Mapping

UART (Serial):
Pins Used: RX and TX.
Description: Used for serial communication, programming, and debugging. They are tied to indicator LEDs (D16 and D17) through 1.5kohm resistors (R67, R68) to visually display data transmission.
I2C Bus:
Pins Used: SCL (Clock) and SDA (Data).
Description: Features an onboard Qwiic connector wired in parallel with the exposed pin header lines.
SPI Bus:
Pins Used: MISO, CLK (Clock), MOSI, and CS_SPI (Chip Select).
Description: Dedicated hardware SPI lines designed for high-speed peripherals like displays, sensors, or storage modules.
PWM Channels:
Pins Used: PWM1, PWM2, and PWM3.
Description: Dedicated pins configured for pulse-width modulation control, ideal for actuators, dimming lights, or driving motors.

3. General-Purpose I/O & Power Management

Flexible IOs:
The digital and analog pins support standard digital input/output, analog reading (ADC), and capacitive touch sensing features.

Power Regulation:
The 3.3V rail is driven by a high-efficiency TLV62569DBVR step-down (buck) regulator capable of supplying up to 2 Amps, providing plenty of current headroom for power-hungry sensors and modules connected to your dev board.

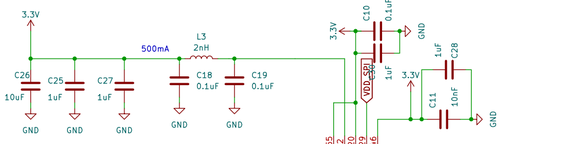
USB & VIN Power Warning:

The USB line is internally connected to the Vin pin. Do not connect the USB and an external power source to Vin at the same time to prevent backfeeding and potential damage to your power setup or computer.

esp32s3 datasheet

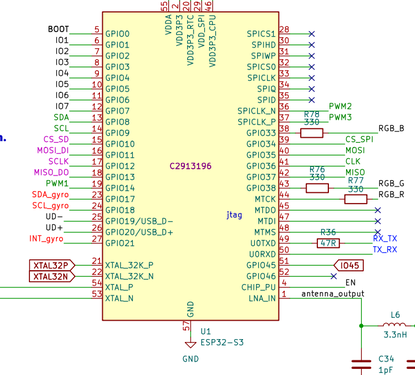
NAME	TYPE	I/O	GPIO	DIGITAL	ADC	UART	SPI	I2C	PWM	TOUCH	I2S
GND	power gnd	-									
TX	signal	output	GPIO43	yes		yes	yes	yes	yes		yes
RX	signal	input	GPIO44	yes		yes	yes	yes	yes		yes
IO0	signal	bidirectional	GPIO1	yes	ADC1	yes	yes	yes	yes	yes	yes
IO1	signal	bidirectional	GPIO2	yes	ADC1	yes	yes	yes	yes	yes	yes
IO2	signal	bidirectional	GPIO3	yes	ADC1	yes	yes	yes	yes	yes	yes
IO3	signal	bidirectional	GPIO4	yes	ADC1	yes	yes	yes	yes	yes	yes
IO4	signal	bidirectional	GPIO5	yes	ADC1	yes	yes	yes	yes	yes	yes
IO5	signal	bidirectional	GPIO6	yes	ADC1	yes	yes	yes	yes	yes	yes
IO6	signal	bidirectional	GPIO7	yes	ADC1	yes	yes	yes	yes	yes	yes
+3.3	power vcc	3.3 V									
GND	power gnd	-									
PWM1	signal	bidirectional	GPIO14	yes	ADC2	yes	yes	yes	yes	yes	yes
PWM2	signal	bidirectional	GPIO47	yes		yes	yes	yes	yes		yes
PWM3	signal	bidirectional	GPIO48	yes		yes	yes	yes	yes		yes
MISO	signal	bidirectional	GPIO37	yes		yes	yes	yes	yes		yes
CLK	signal	bidirectional	GPIO36	yes		yes	yes	yes	yes		yes
MOSI	signal	bidirectional	GPIO35	yes		yes	yes	yes	yes		yes
CS_SPI	signal	bidirectional	GPIO34	yes		yes	yes	yes	yes		yes
SCL	signal	bidirectional	GPIO9	yes	ADC1	yes	yes	yes	yes	yes	yes
SDA	signal	bidirectional	GPIO8	yes	ADC1	yes	yes	yes	yes	yes	yes
VIN	power vcc	5 V									
+	power vcc	3.7 V bat									
-	power gnd	-									
BOOT	signal	input	GPIO0	yes		yes	yes	yes	yes		yes
BAT_SENSE	signal	input	GPIO7	yes	ADC1	yes	yes	yes	yes	yes	yes

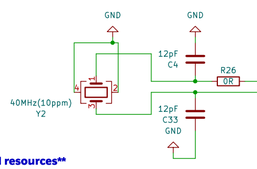
ESP32-S3FN8 - Main Microcontroller

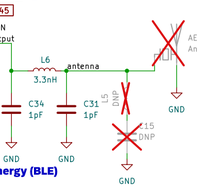


The **ESP32-S3FN8 is the main processing unit of Elyssa. It features a dual-core 32-bit Xtensa LX7 processor running at up to **240 MHz**, 8 MB of embedded flash, integrated 2.4 GHz Wi-Fi and Bluetooth Low Energy (BLE), and native USB connectivity. The chip has no PSRAM on this board.**

The MCU manages the board's peripherals, executes user applications, processes sensor data, and provides wired and wireless communication. Elyssa supports the Arduino IDE, with board-specific pin definitions and built-in functions for the onboard RGB LED and IMU.







Hardware interfaces and resources

- Native USB: Programming, firmware upload, and serial debugging through the USB-C connector.**
- GPIO: Digital inputs and outputs, with support for analog input, capacitive touch, and PWM on designated pins.**
- Communication buses: I²C on GPIO8/GPIO9, UART on GPIO43/GPIO44, the expansion-header SPI bus on GPIO34-GPIO37, and a separate microSD SPI bus on GPIO10-GPIO13.**
- Onboard peripherals: RGB LED, LSM6DS3TR-C IMU, microSD socket, BOOT button, and optional battery-voltage monitoring through GPIO7 when the JP3 solder jumper is bridged.**

Wi-Fi - Wireless Networking

The integrated 2.4 GHz Wi-Fi supports connecting Elyssa to a wireless router or operating as an access point (AP). Users can build IoT devices, web-controlled hardware, sensor dashboards, local configuration pages, and network-connected data loggers.
Using the Arduino `WiFi.h` library, applications can scan nearby networks, connect to an access point, create a hotspot, and exchange data using TCP, UDP, HTTP, or other supported network protocols.
Example applications include:
- Hosting a local web server to control GPIOs or display sensor readings.
- Sending IMU and other sensor data to a computer or cloud service.
- Configuring the device through a phone connected to its own Wi-Fi access point.
- Monitoring and controlling robots or automation projects over a local network.

Bluetooth Low Energy (BLE)

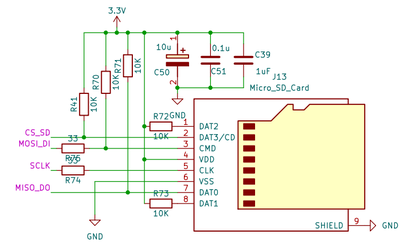
BLE enables Elyssa to communicate with smartphones, computers, and other BLE-capable devices without requiring a Wi-Fi router. Users can implement wireless sensor streaming, mobile-app control, configuration interfaces, and low-power data exchange.

Typical applications include sending sensor measurements to a phone, receiving commands from a mobile application, exposing custom GATT services and characteristics, and creating BLE peripherals or central devices using compatible ESP32 Arduino BLE APIs.

ESP-NOW - Direct ESP32-to-ESP32 Communication

ESP-NOW is a connectionless communication protocol that uses the ESP32's Wi-Fi radio to exchange data directly between compatible devices, without requiring a conventional Wi-Fi router or IP connection.

MicroSD Card Socket

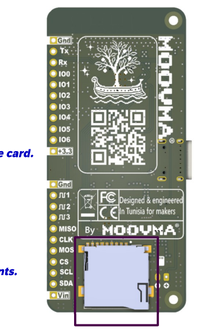


The onboard microSD socket allows Elyssa to store and retrieve data using a removable memory card. The card communicates with the ESP32-S3 through a **dedicated SPI bus**, separate from the SPI bus exposed on the board's expansion header. This allows SD storage to operate independently of external SPI peripherals.

***Hardware connections:**
- **CS - GPIO10 (`SD_CS`):** Selects the SD card.*
- **MOSI - GPIO11 (`SD_MOSI`):** Sends commands and data to the card.*
- **SCLK - GPIO12 (`SD_SCK`):** Provides the SPI clock.*
- **MISO - GPIO13 (`SD_MISO`):** Receives data from the card.*
- **Power - 3.3 V:** Supplies the socket and SD card; local capacitors help stabilize the supply.*
- **Pull-up resistors:** Keep selected card signals at defined logic levels when they are not actively driven.*

***What users can build:** Data loggers for sensors and the IMU, robot telemetry recorders, offline data storage, CSV measurement logs, and systems that read configuration files or replay recorded measurements.*

***Arduino support:** Elyssa uses the standard `SPI` and `SD` libraries. Since the SD card has its own SPI bus, initialize a separate `SPIClass` instance:*



H1 MountingHole
H2 MountingHole

Elyssa · ESP32-S3 development board
Designed & revised by **Bejaoui Hamza**
Moovma · open-source hardware from Tunisia · github.com/moovma

Sheet: /
File: elyssa_v5.kicad_sch
Title:
Size: A2 | Date: | Rev:
KiCad E.D.A. 10.0.1 | Id: 1/1