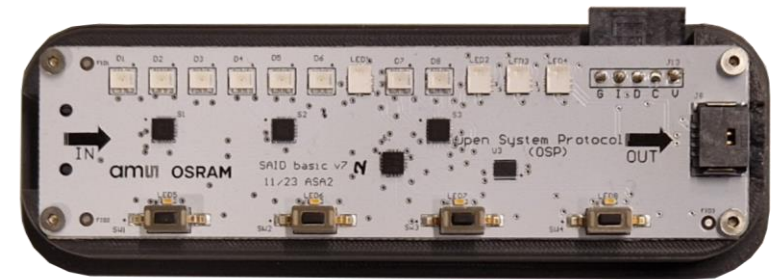


# SAIDbasic manual

[https://github.com/ams-OSRAM/OSP\\_aotop/tree/main/examples/saidbasic](https://github.com/ams-OSRAM/OSP_aotop/tree/main/examples/saidbasic)

Last update 2025 Aug 6



# SAID, RGBi and OSP ecosystem

## Open System Protocol (OSP)

- A protocol, introduced by ams OSRAM, for (automotive indoor) decorative lighting (initially for “OSIRE”)
- OSP is an open “standard” and full documentation is available for market players and competitors to implement own devices
- ams OSRAM is actively working with partners to build an ecosystem of components around OSP
- An OSP system consists of a root MCU with OSP firmware and a daisy chain of up to 1000 OSP compliant nodes
- Two node types exist from ams OSRAM: RGBi and SAID

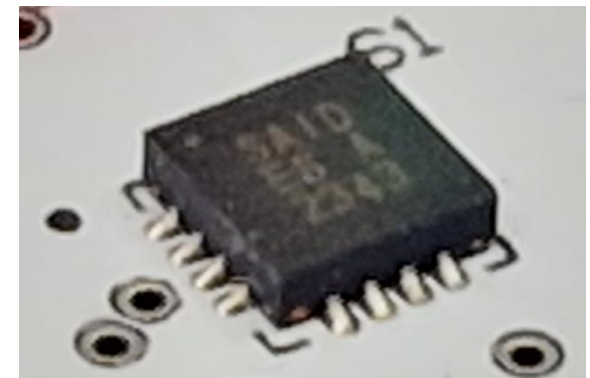
## RGBi (or “RGB intelligent ”)

- First type of node (available 2023), the E3731i nicknamed RGBi
- Contains three (PWM) drivers and three integrated LEDs (red, green, blue)
- “intelligent” (=OSP compliant, can send and receive OSP telegrams)
- color/temperature calibrated

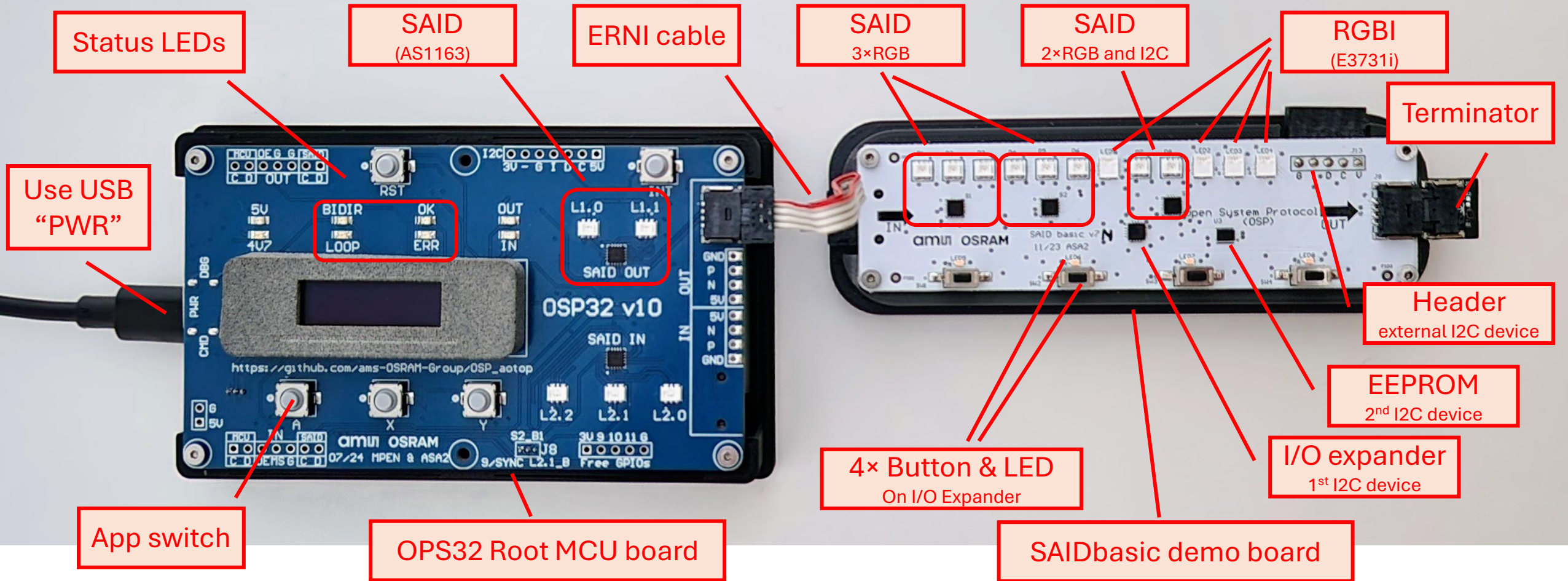
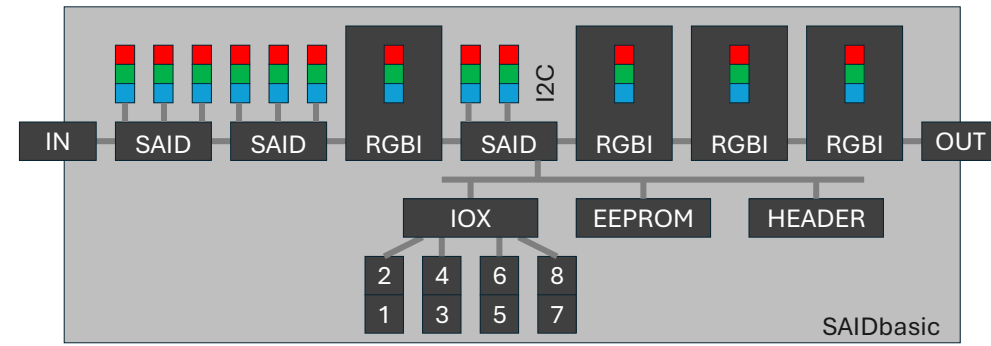


## SAID (or “stand-alone intelligent driver”)

- A second type of node (available 2024), the AS1163 nicknamed SAID
- Has 9 PWM drivers: 3 channels to drive 3 external RGB modules (or 9 stand-alone LEDs)
- One channel can be configured to act as I2C master allowing I2C devices in the OSP chain for example, I2C sensors, or I2C EEPROMs with calibration parameters
- It has 2-wire SPI (towards MCU) and group cast



# Hardware setup



# Software (generic)

The **SAIDbasic** firmware contains these apps:

- Animation script
- Running LEDs
- Switch flag
- Dithering

- The firmware for a demo typically contains multiple **apps** (and a commands over serial)
- Switch between apps by pressing “A” button (app starts fresh with a scan of the chain)
- The OLED shows the app name and what the function of X and Y button is (in that app)
- Autodetect of BiDir/Loop mode (on each app start) – see status LEDs
- Apps auto-detect RGB triplets and I2C bridges – they adapt their animation to that
- While an app runs, the green “OK” led blinks (“heartbeat”)
- When there is an error, green stops, red “ERR” switches on, OLED shows message
- LEDs are driven in “night” mode (OSIRE 10mA, SAID 12mA).
- If there are too many LEDs the power over USB is not enough (LEDs will switch off)
- RST resets the firmware

It helps to plug the USB cable in the center USB connector labeled “PWR” (but the “CMD” one shows diagnostic info on a PC terminal)

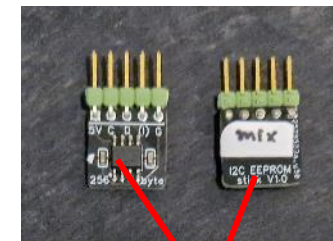
# Animation script

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|with| start of | end of | red | green | blue |
|prev| region | region | brightness| brightness| brightness|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Script instructions only have 3 bits for each color, and 3 for a region, so script animations are coarse.



External EEPROM  
with animation scripts

## Description

- Plays a light show as defined by an animation script
- Tries to find a SAID with an I2C bridge with an EEPROM
- If there is an external EEPROM stick (I2C address 0x51) it favors that over an internal EEPROM (address 0x50)
- If there are multiple (of the same kind, external or internal) the first one is taken
- If no EEPROM is found, uses the heartbeat script included in the firmware
- If an EEPROM is found, loads the script from the EEPROM and plays that
- The internal EEPROM (on the SAIDbasic board) contains the rainbow script
- External EEPROMs are flashed with bouncing-block and color-mix

## Notes

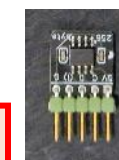
- Ensure the I2C EEPROM stick faces "chip up" otherwise there is a short circuit (see PCB labels)
- Safest is to only swap an EEPROM when USB power is removed
- The script loading takes place on app start: (1) power cycle, (2) reset, (3) "A" button.

## Buttons

- The X and Y buttons control the FPS level (frames-per-second animation speed).

## Goal

- Show that the root MCU can access I2C devices (EEPROM) e.g. for calibration values



Internal EEPROM  
with animation scripts



Connector  
for external EEPROMs

To flash an animation script see

[https://github.com/ams-OSRAM/OSP\\_aotop/tree/main/examples/eepromflasher](https://github.com/ams-OSRAM/OSP_aotop/tree/main/examples/eepromflasher)

# Running LEDs

## Description

- There is a "virtual cursor" that runs from the begin of the chain to the end and then back
- Chain length and node types are auto detected
- Every 25ms the cursor advances one LED and paints that in the current color
- Every time the cursor hits the begin or end of the chain, it steps color
- Color palette: red, yellow, green, cyan, magenta

## Buttons

- The X and Y buttons control the dim level (RGB brightness)

## Goal

- To show that various OSP nodes can be mixed and have color/brightness matched

In this demo, there is no algorithm running using LED color calibration data to stabilize colors over LEDs and over temperature



The command "apps config swflag" (via USB-to-COM terminal) allows configuring which flags to show. By putting the command in boot.cmd the configuration is persistent.

# Switch flag

## Description

- Shows one (static) flag at a time, e.g. the Dutch national flag red/white/blue spread over the OSP chain
- Tries to find a SAID with an I2C bridge with a "selector", an I/O-expander with four buttons and four indicator LEDs
- If there is no I/O-expander, shows four static flags switching on a time basis
- If there are multiple selectors the first SAIDbasic one is taken, or the first SAIDsense when no SAIDbasic
- When a selector is found the four buttons select which flag to show
- The indicator LEDs indicate which button/flag was selected

## Buttons

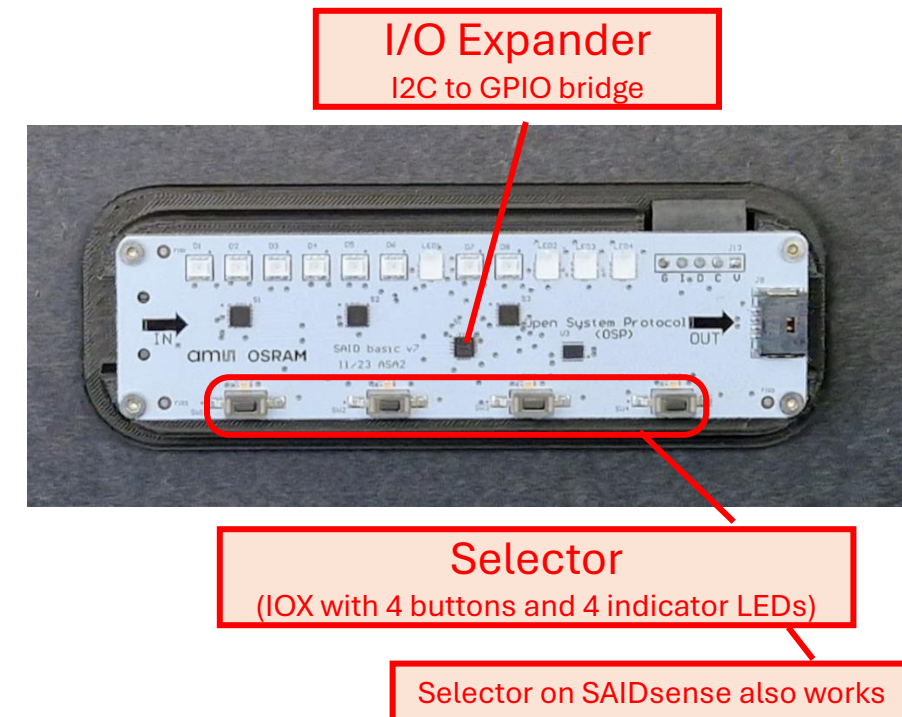
- The X and Y buttons control the dim level (RGB brightness)

## Notes

- When the app quits, the indicator LED switches off
- This app adds a command to configure which four flags will be shown

## Goal

- To show a "sensor" (button) being accessible from the root MCU (the ESP)



# Dithering

## Description

- The LEDs are in a dimming cycle (dim up, then dim down, then up again, etc).
- All LEDs dim synchronously and at the same level (so RGBs look white).
- Dithering can be enabled/disabled.

## Buttons

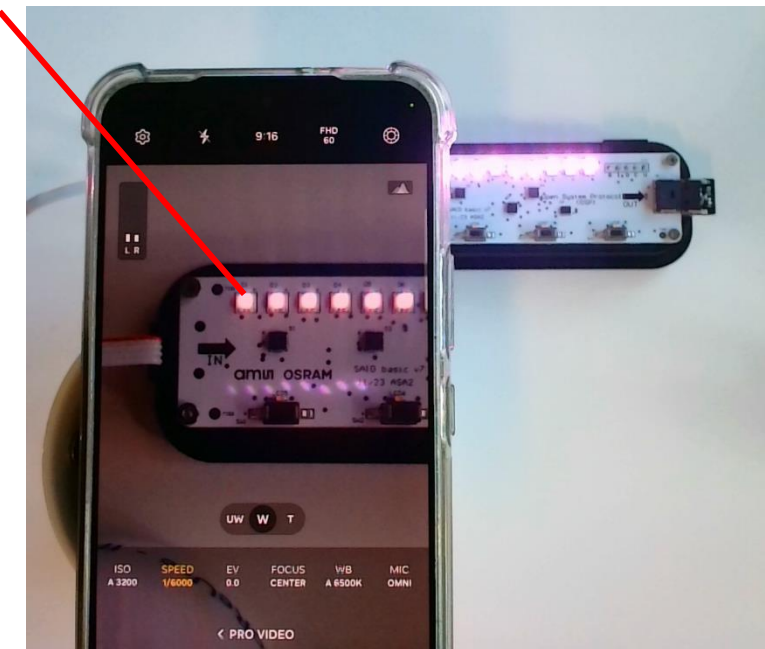
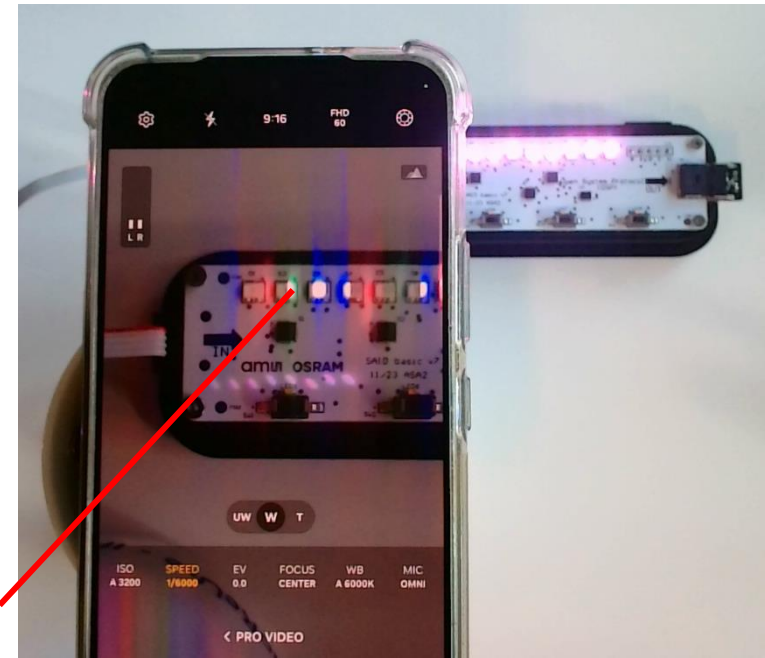
- The X button toggles dim cycling on/off.
- The Y button toggles dithering on/off.

## Goal

- To show the effect of dithering
- View the LEDs with e.g. LED Light Flicker Meter  
[https://play.google.com/store/apps/details?id=com.contechity.flicker\\_meter](https://play.google.com/store/apps/details?id=com.contechity.flicker_meter)
- Alternatively, view the LEDs with a mobile phone camera in “pro mode” video with high framerate – faster than 1/3000

Dithering disabled  
flicker

Dithering enabled  
no flicker





# Command interpreter

The saidbasic firmware also includes a command interpreter

This allows sending commands from the PC to the evalkit via a USB cable

## Prerequisites

- USB cable between PC with OSP32; use the port labeled “CMD” on OSP32 board (“UART” on ESP board)
- A serial terminal; we suggest Arduino IDE (see also next bullet) <https://www.arduino.cc/en/software/> or e.g. [putty](#)
- Driver for the USB-to-serial chip; it is part of the Arduino install, or install the CP2102N [driver](#) manually
- Start the serial terminal using 115200 8N1 as parameters. The command interpreter echos and uses only LF

## Notes

- If commands is the main goal (not demos) use the osplink firmware instead
  - [https://github.com/ams-OSRAM/OSP\\_aotop/tree/main/examples/osplink](https://github.com/ams-OSRAM/OSP_aotop/tree/main/examples/osplink)
- An introduction to the command handler can be found in aocmd
  - [https://github.com/ams-OSRAM/OSP\\_aocmd?tab=readme-ov-file#example-commands](https://github.com/ams-OSRAM/OSP_aocmd?tab=readme-ov-file#example-commands)
- To prevent apps from sending commands, by activating the “voidapp”
  - Give command **apps switch voidapp**

The command interpreter runs **boot.cmd** at startup (created with **file record**). This allows to configure the demo persistently. Suggested configurations

```
apps conf swflag set  dutch europe italy mali  
topo dim 200
```

```
apps hide aniscript  
apps switch runled
```