FOLLOW - UP

Sage Marks and Matt Krueger

Home Décor

Embedded Systems 2025

Issues 1: Address Space of Uno

- **Problem:** **memory size** of ATmega328p
- ATmega328P has **2KB SRAM** whilst there are **4096 addresses** on the LED matrix (64^2). Ultimately, the ATmega328p is **NOT COMPATIBLE** with any 64x64 led matrix due to its limited address space.

- **Solution:** use adequate microcontroller
- An easy alternative (and a microcontroller that we have on hand already) is the **ESP32.** The ESP-Wroom-32 has **512MB** of RAM, which can access the 4096 led addresses in the matrix… and more.

ESP-32 Specs:

- Datasheet: ESP-WROOM-32
- Blog featuring esp32 to 64x64 lcd : Waveshare

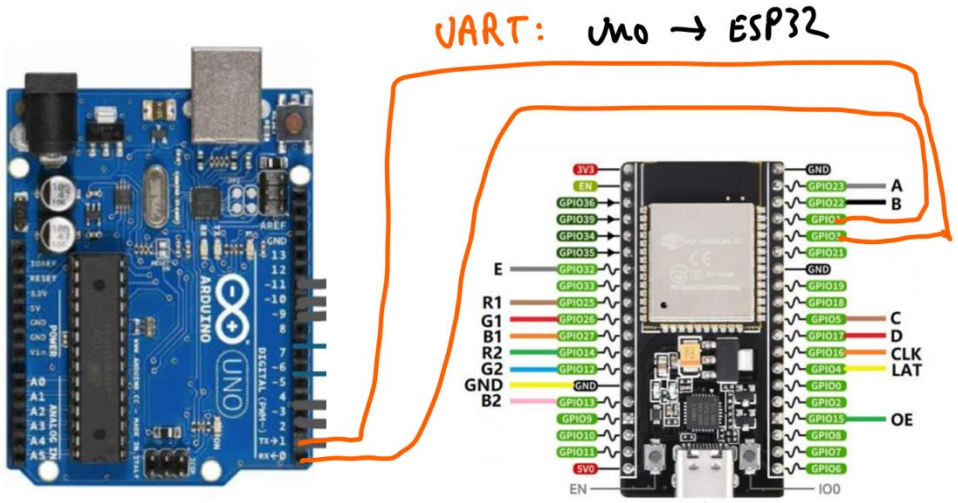
3.1 CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

Compromise

UART pins would be open after connecting ESP32 to the LED matrix, which we could **connect to our Arduino** with other sensors.



For most ESP32 boards the UART pin assignment is as follows:

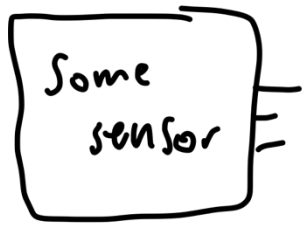
UART Port	TX	RX	Remarks
UART0	GPIO 1	GPIO 3	Used for Serial Monitor and uploading code; Can be assigned to other GPIOs;
UART1	GPIO 10	GPIO 9	Must be assigned to other GPIOs
UART2	GPIO 17	GPIO 16	Can be assigned to other GPIOs

Separate programs would be written:

1. Arduino + peripheral device data capture
2. ESP32 + LED interface

As a compromise, we could use **ESP32 solely for interfacing the LED** and not utilize any of its Bluetooth/Wifi capabilities. It would serve as a middleman for the Arduino and the LED

High Level Workflow



+

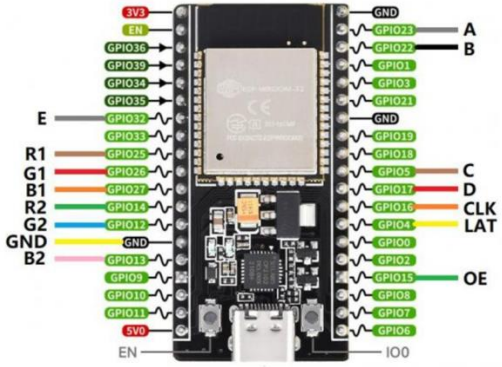
Peripheral Sensors and Arduino analysis

- Capture data to send to LED driver (esp)

1) Data Capture

2) Communication

UART seems feasible



ESP32 analysis and LED Interface

Interpret signal to drive LED

- Compute LED row/col and RGB
- Send to LED Matrix

3) Display

Required Components

Component	Amount Required	Usage	Do we have?
ESP32	1	LED Matrix Interface	**YES**
64x64 LED Matrix	1	Display sensor data	**YES**
Arduino UNO	1	Capture sensor data	**YES**
Sensor	?	?	**no**