REPORT ON PCB DESIGN AND SOFTWARE DEVELOPMENT

Prepared By : MD Shahnauze Ahsan

FEBRUARY 18, 2025

EHSAN.PUC16@GMAIL.COM
https://sites.google.com/view/ahsanslab

PCB design and Development Project :

This report details the design, development, and challenges of a **two-PCB integrated system**, comprising:

- **PCC (Peripheral Control Card)** – A high-precision 4-layer PCB designed for current control, ADC/DAC integration, and real-time data acquisition.
- **MPCB (Motherboard)** – The central control unit managing communication, power distribution, and wireless connectivity using ESP32 for Wi-Fi/Bluetooth and STM32F412 for embedded control.

The project was designed in **EasyEDA** and aims to provide stable signal integrity, efficient power distribution, and modular connectivity**.**

1. My Role in the Project

As the **Lead PCB & Embedded Systems Engineer**, my tasks included:

- **Schematic Design & Circuit Architecture** – Component selection and layout planning.
- **PCB Layout & Routing** – Ensuring signal integrity, EMI control, and optimized component placement.
- **Power Management & Regulation** – Implementing efficient voltage regulators for multiple power levels.
- **Wireless & High-Speed Communication** – Integrating I2C, SPI, UART, USB, Wi-Fi, and Bluetooth.
- **Debugging & Testing** – Performing power validation, signal integrity checks, and firmware flashing.

2. System Overview & Design Choices

2.1 Motherboard-PCB System Architecture

The **MPCB (Motherboard)** acts as the central hub, while the **PCC (Peripheral Board)** handles analog operations and current control of our Photo-Reactors.

❖ MPCB (Motherboard) - Central Control Unit

- **STM32F412 MCU**: Manages system operations and data exchange.
- **ESP32 Module**: Provides Wi-Fi connectivity for remote communication.
- **Power Regulation**: Converts 12V to 9V & 3.3V for peripherals and microcontrollers.
- **Communication Interfaces**: I2C, UART, SPI, and USB ensure reliable inter-board connectivity.
- **Rotary Encoder & LEDs**: For user interaction and status monitoring.

© *JLF Lab, ICIQ*

❖ **PCC (Peripheral Control Card) -** Analog Processing Unit

- **STM32F401Cx MCU**: Dedicated to DAC, ADC, and MOSFET-based current control.
- **High-Precision Analog Circuits**:
  - **8 OP-AMP channels** for MOSFET drivers.
  - **MCP4728 DAC** for programmable voltage control.
  - **ADS1015 ADC** for accurate current detection.
- **USB Tracer**: Enables USB data monitoring.
- **Multi-Layer Routing**: Ensures low-noise analog signal processing.

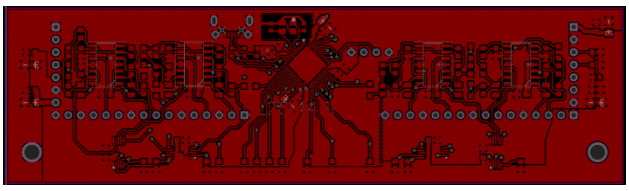


Fig.1 : Top Layer (PCC Board)

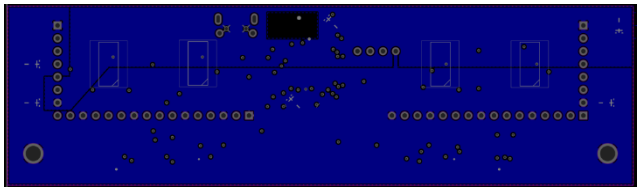


Fig.2 : Bottom Layer (PCC Board)

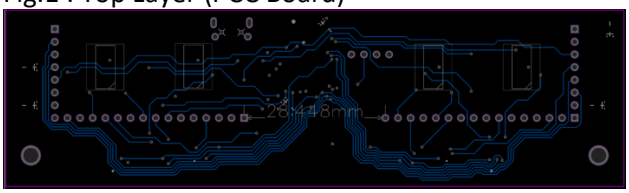


Fig.3 : Inner Layer 1 (PCC Board)

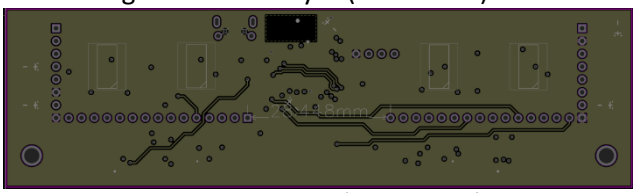


Fig.4 : Inner Layer 2 (PCC Board)

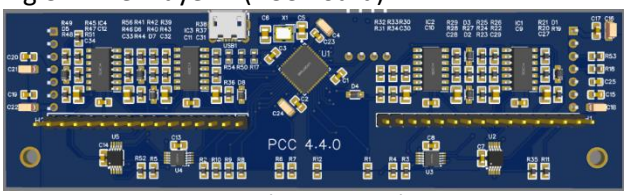


Fig.5 : PCBA Top view (PCC Board)

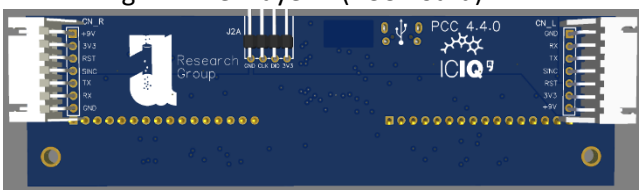


Fig.6 : PCBA Bottom View (PCC Board)

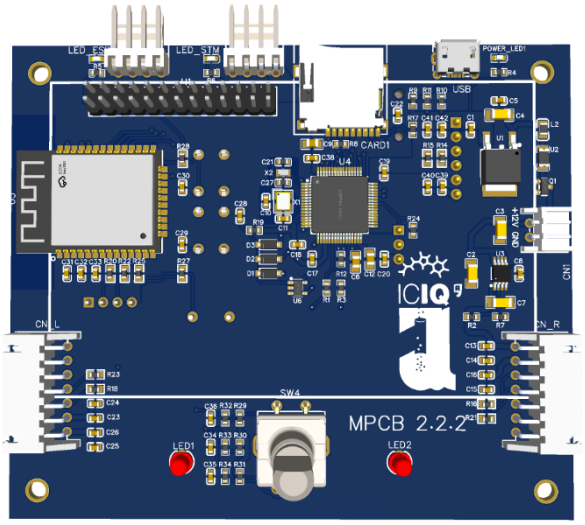


Fig.7 : MPCB Board PCBA Top View

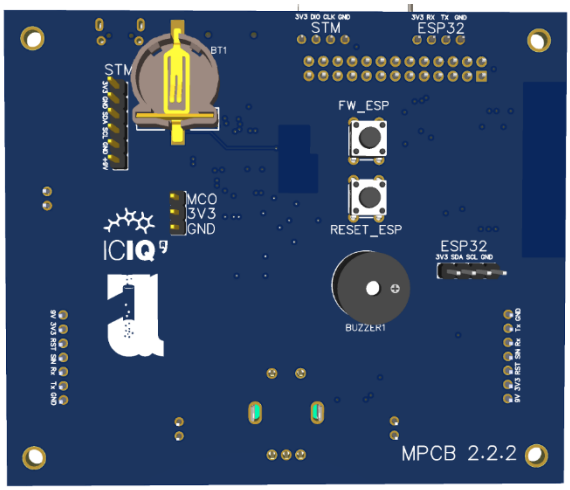


Fig.8 : MPCB Board PCBA Bottom View

© *JLF Lab, ICIQ*

2.2 PCB Layer Stack-Up & Routing Strategy

Both MPCB & PCC are **4-layer PCBs**, providing improved signal integrity and efficient power distribution:

Layer	Function
Top Layer	High-speed signals, power, and main components.
Inner Layer 1	Power plane (3.3V, 9V) for noise-free distribution.
Inner Layer 2	Ground plane for EMI shielding.
Bottom Layer	Secondary routing, low-speed signals, and connectors.

❖ **Benefits of a 4-layer design:**

- Reduced EMI & noise.
- Better power integrity and heat dissipation.
- More efficient routing of analog and digital signals.

2.3 Power Distribution Strategy

Power supply is critical for maintaining stable system performance.

Power Rail	Purpose	Voltage Regulation
12V Input	Main power source	Step-down regulator
12V → 9V (150mA)	Used for specific peripherals	TPS54294 Buck Converter
12V → 3.3V (1A)	Supplies microcontrollers & communication interfaces	UAB71336 Regulator
3.3V	Used for logic circuits & I2C communication	Direct power rail

Optimized with decoupling capacitors to prevent noise & voltage fluctuations.

2.4 Communication & Data Exchange

MPCB ↔ PCC Communication:

- **I2C Bus**: Used for data exchange between STM32 & peripherals.
- **UART Interface**: Handles debugging and serial data transmission.
- **USB Connector**: Enables direct programming & debugging.
- **SPI Bus**: Facilitates high-speed data transfer with ADC/DAC.

Wireless Communication:

- ESP32 handles Wi-Fi/Bluetooth for remote monitoring.
- UART/I2C integration enables real-time updates via network.

© *JLF Lab, ICIQ*

3. Challenges and Solutions

3.1 Power Management & Heat Dissipation

Problem: Voltage regulators generated excessive heat.
Solution: Implemented thermal vias, ground plane heat spreading, and copper pouring.

3.2 Signal Integrity & EMI Control

Problem: High-speed USB, I2C, and SPI lines experienced noise interference.
Solution: Used shielded differential pairs for USB. Shortened trace lengths for high-speed signals. Ground plane isolation reduced EMI issues.

3.3 Wireless Interference (ESP32)

Problem: Wi-Fi signals interfered with digital circuits.
Solution: Dedicated RF clearance zone for ESP32. Ground separation around the antenna to improve signal strength.

3.4 XTAL & Load Capacitance Tuning

Problem: Proper Load capacitance selection for XTAL and stability issues.
Solution: Fixed load capacitance by using the eqn. CL = (C1 * C2) / (C1 + C2) + Cstray. Ground separation around the antenna to improve signal strength. To prevent interference, noise, and unwanted coupling that can affect the oscillator's stability and accuracy kept some area out below the XTAL and MCU.

3.5 Debugging & Firmware Updates

Problem: Both STM32 & ESP32 required accessible programming interfaces.
Solution: SWD programming headers for STM32 firmware flashing. Reset & Firmware Update buttons for ESP32.

4. Testing & Validation

- **Power-On Tests:** Verified voltage levels & current draw.
- **Continuity & Signal Integrity:** Checked PCB traces for stability.
- **Microcontroller Firmware Flashing:** Ensured STM32 & ESP32 firmware loaded correctly.
- **Wireless Communication Testing:** Tested ESP32's Wi-Fi/Bluetooth capabilities.
- **Thermal Monitoring:** Ensured efficient heat dissipation.
- **DAC/ADC Calibration:** Verified precision in voltage/current control.

© *JLF Lab, ICIQ*

Embedded Software Development Projects

1. Raspberry Pi-based Photo-Reactor Control System

Project Overview

This project involves developing an embedded application on a Raspberry Pi with a touchscreen interface to control and switching of photo-reactors. The system also incorporates temperature sensors that utilize the one-wire communication protocol to provide real-time temperature readings.

Hardware Used

- **Raspberry Pi (Model 4B)**: Acts as the main controller, running the embedded software.
- **Touchscreen Display**: Provides a user interface for controlling the photo-reactors.
- **LED Dimmer**: Adjusts the intensity of the LEDs in the photo-reactors.
- **RF 2.4GHz Module**: Facilitates wireless communication for remote control.
- **Power Supply**: Supplies power to the Raspberry Pi and other components.
- **Temperature Sensors (DS18B20)**: One-wire sensors used for temperature monitoring in the reactors.

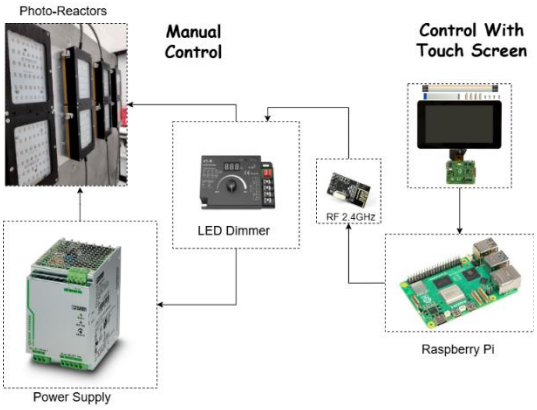


Fig.9 : Photoscale control unit block diagram

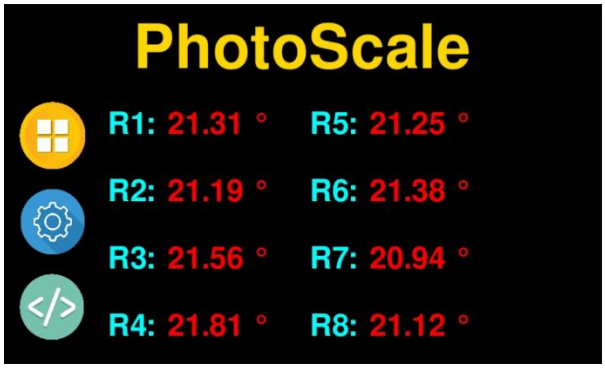


Fig.10 : Touchscreen app first page



Fig.11 : Touchscreen app Developer page

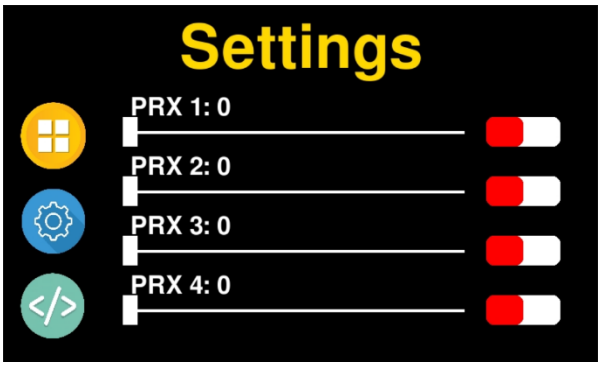


Fig.12 : Touchscreen app Settings page

© *JLF Lab, ICIQ*

Programming Languages & Tools

- **Python**: Main programming language for interfacing with hardware and implementing control logic.
- **PyGame (GUI Toolkit)**: Used to develop the touchscreen interface.
- **OneWire Protocol**: Utilized for temperature sensor communication.
- **RF Communication Libraries**: Integrated for remote operation.

Key Functionalities

- **Touchscreen Control**: The user can manually control the light intensity of the photo-reactors using a graphical user interface.
- **Remote Control via RF**: An RF module allows for wireless control of the system.
- **Real-time Temperature Monitoring**: Temperature readings from each reactor are displayed on the screen.
- **Embedded Data Logging**: Logs the temperature and light intensity for performance analysis.
- **Failsafe Mechanism**: In case of a system failure, the dimmer reverts to a default state.

Code Structure

```
PhotoScale/
├── main.py                # Application entry point
├── photo_scale_app.py     # Core class managing UI and event handling
├── gpio_control.py         # Manages GPIO pin operations (PWM, toggles)
├── sensors.py             # Reads temperature data from DS18B20 sensors
├── ui_components.py       # Handles UI elements (buttons, sliders, switches)
├── assets/                # Stores images/icons for UI
└── docs/                  # Documentation files
```


© *JLF Lab, ICIQ*

2. ADS131M08-based Electrical Data Acquisition and Supervision System

Project Overview

At Schneider Electric, the project involved developing a high-accuracy electrical data acquisition system for monitoring voltage, current, temperature, and noise parameters. The system also integrates LTE-M connectivity for cloud-based data analysis and anomaly detection.

Hardware Used

- **Arduino MKR NB 1500**: Serves as the primary microcontroller, supporting LTE-M communication.
- **ADS131M08**: An 8-channel, 24-bit delta-sigma ADC for high-precision data acquisition.
- **Current & Voltage Sensors**: Measure electrical parameters in real-time.
- **LTE-M Module**: Provides IoT-based cloud connectivity.
- **PCB Design in KiCad**: Developed and fabricated for component integration.

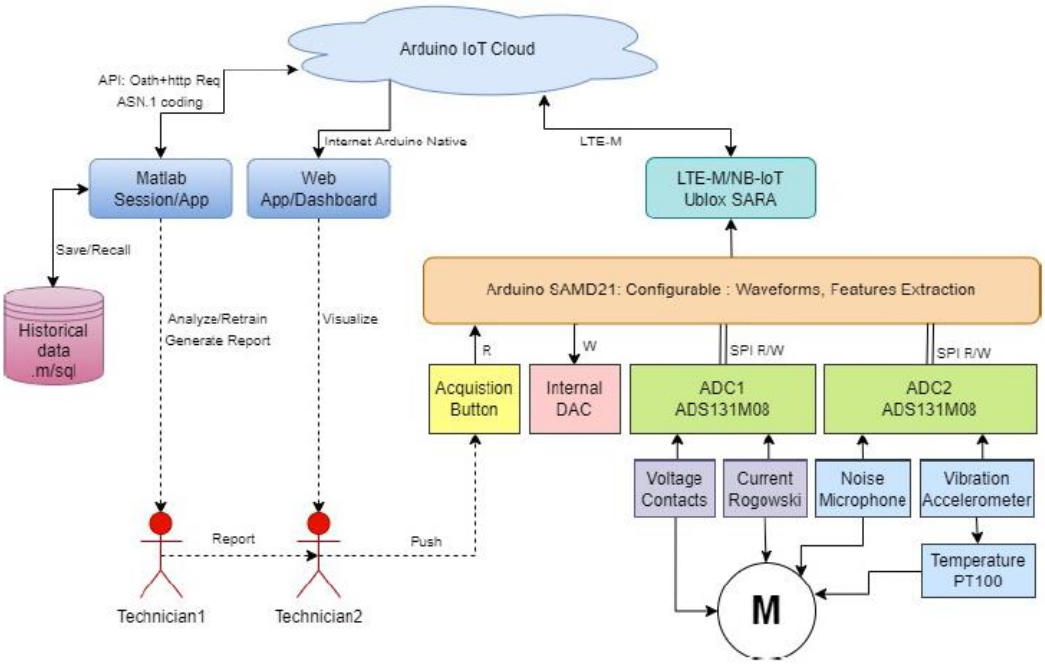


Fig.13 : Data Acquisition System Block diagram

Programming Languages & Tools

- **C++ (Arduino Framework)**: Used for firmware development and real-time processing.
- **FreeRTOS**: Implemented for task scheduling and real-time operations.
- **Matlab & Simulink**: Used for data analysis and generating embedded C-code for firmware integration.
- **Arduino IoT Cloud & API**: Enables remote monitoring and data logging.

Key Functionalities

- **High-Precision Electrical Data Acquisition**: Utilizes ADS131M08 ADC to acquire multi-channel data.

© *Schneider Electric, France*

- **Cloud Connectivity via LTE-M**: Enables real-time remote monitoring and control.
- **Matlab Integration**: Facilitates post-processing and data analysis.
- **RMS Measurement & Anomaly Detection**: Embedded Simulink-generated algorithms detect anomalies.
- **Low-Power Design**: Optimized for battery-operated applications.

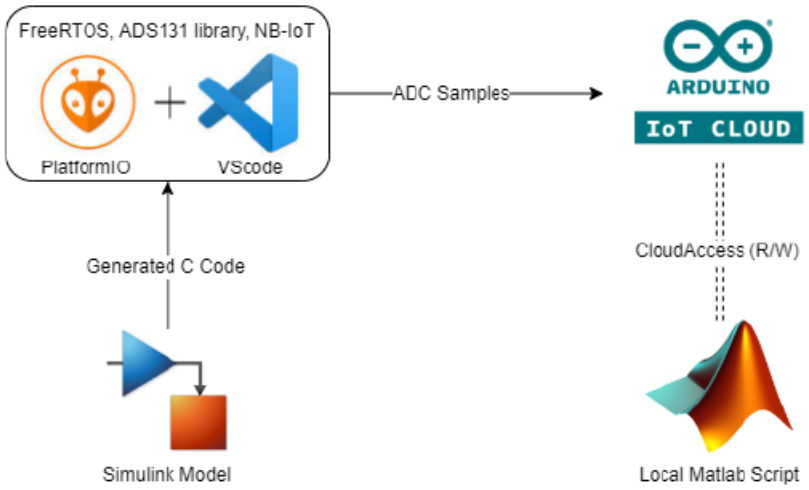


Fig.8 : Data acquisition system software architecture

Code Structure :

1. **RTOS-Based Task Management**
   - FreeRTOS is used for managing multiple tasks, including cloud updates, LED control, push button handling, and DAC updates.
   - Tasks are created using `xTaskCreate()` and scheduled with `vTaskStartScheduler()`.

2. **ADS131M08 ADC Handling**
   - SPI communication for reading and writing register values.
   - ADC sampling and RMS calculations are performed in dedicated tasks.
   - Post-processing is implemented for anomaly detection.

3. **Cloud Integration**
   - Arduino IoT Cloud integration allows remote data access.
   - Cloud updates are managed through FreeRTOS tasks.

4. **Peripheral Handling**
   - LED control with various blink states.
   - Push button handling with debounce mechanism.
   - DAC waveform generation for analog signal output.

5. **Error Handling**
   - Uses `vSetErrorLed()` for error indication.
   - Implements FreeRTOS error handling routines.

© *Schneider Electric, France*