Final Project Report: Smart Greenhouse

Student:
Ali Kashi Pazha (40121723)

Instructor:
Dr. Yarmand

Course:
Microprocessor Systems - 4041

Simulation Software:
Proteus 8.13

Development Environment:
STM32CubeIDE / STM32CubeMX

GitHub Repository:
Link

1. Introduction and Objectives

The objective of this project is to design and simulate an intelligent environmental monitoring and control system tailored for a greenhouse application. The system is designed to monitor critical parameters—temperature, light intensity, and soil moisture—in real-time and execute autonomous control actions based on predefined thresholds. The system features two operational modes: **AUTO** (Autonomous) and **MANUAL** (User-controlled), providing both efficiency and flexibility.

---

2. Hardware Specifications and Components

The following components were utilized for the hardware design and simulation:

- **Central Processing Unit (MCU):** STM32F103C8T6 (ARM Cortex-M3 core) operating at a 72 MHz clock frequency.
- **Temperature Sensor:** DS18B20 digital sensor using the One-Wire protocol for high-precision thermal measurement.
- **Light Sensor:** LDR (Light Dependent Resistor) in a voltage divider configuration for analog light detection.
- **Soil Moisture Simulator:** A 10K Potentiometer (POT-HG) generating a variable analog voltage (0 to 5V).
- **Display Unit:** SSD1306 OLED module using the I2C protocol for local data visualization.
- **Actuators (Simulation via LEDs):**
  - **Ventilation Fan:** Green LED (Digital Output).
  - **Irrigation Pump:** Blue LED (Digital Output).
  - **Heating System:** Red LED (PWM-controlled intensity).
  - **Lighting Adjustment:** Yellow LED (PWM-controlled intensity).
- **Communication:** UART protocol via Virtual Terminal for remote monitoring and manual override.

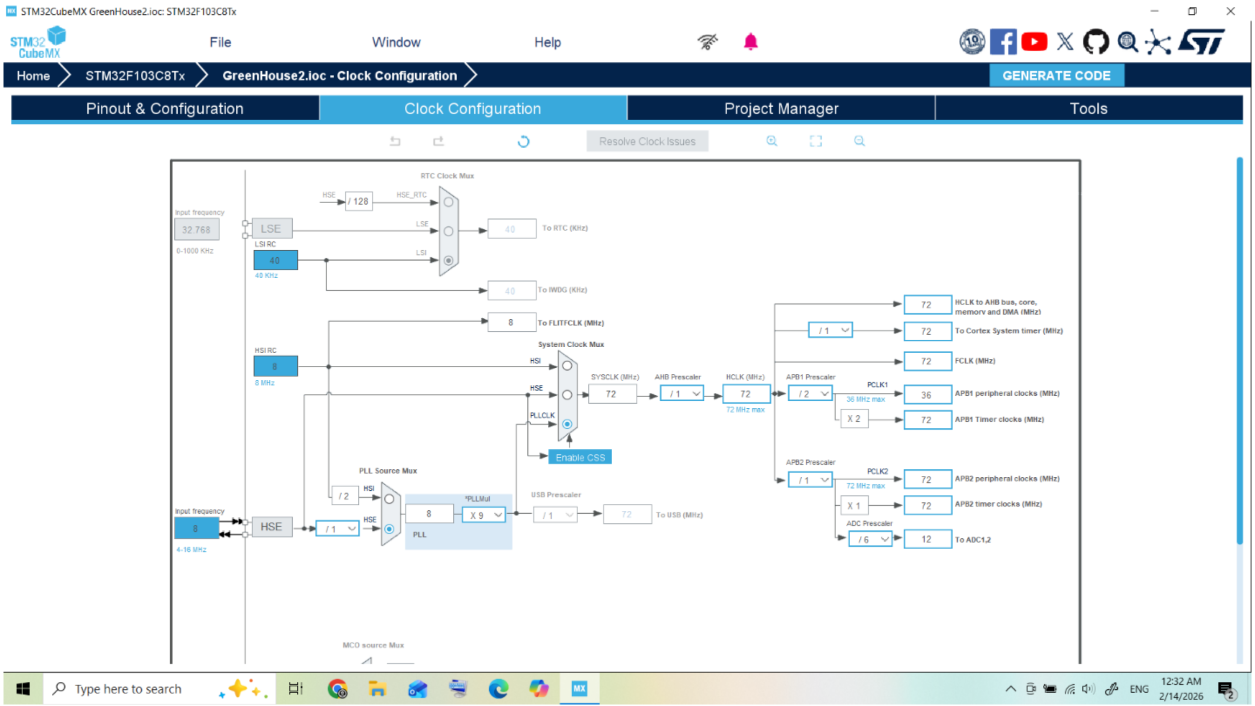
---

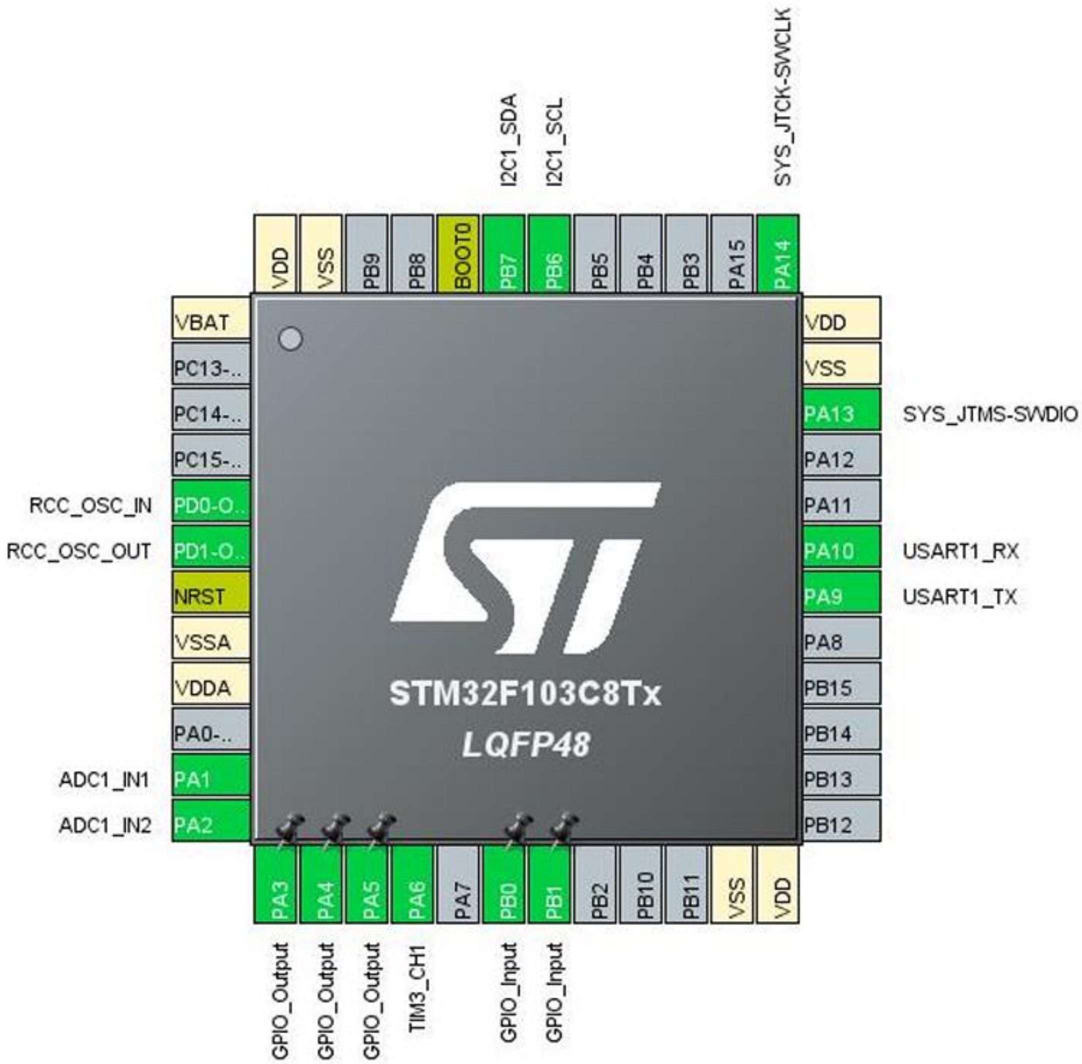
3. Peripheral and System Configuration

The firmware was configured using STM32CubeMX with the following parameters:

- **RCC:** System clock set to 72 MHz using an external High-Speed Crystal (HSE).

- **ADC1:** Two analog channels enabled (IN1 for light, IN2 for soil moisture) in Continuous Conversion Mode with 12-bit resolution.
- **TIM3:** Configured in PWM Generation mode (Channels 1 and 2) to control the power intensity of the heater and lighting system.
- **USART1:** Configured at a Baud Rate of 115200 bps for serial data transmission.
- **I2C1:** Configured in Standard Mode (100 kHz) for OLED communication.
- **GPIO:** Digital pins defined with internal Pull-up resistors for push-buttons (Mode and Door sensors).





4. Software Design and Control Logic

The firmware was developed in C using the STM32 HAL (Hardware Abstraction Layer) libraries. The logic is divided into four primary modules:

A. Data Acquisition

Analog values are captured via the ADC, and temperature data is extracted through a custom One-Wire driver.

B. Autonomous Logic (AUTO Mode)

The MCU executes threshold-based decisions:

1. **Ventilation:** If Temp > 30.0°C, the Fan (PA3) is toggled ON.
2. **Irrigation:** If Soil Moisture (ADC value) < 1500, the Pump (PA4) is toggled ON.
3. **Heating:** If Temp < 20.0°C, the Heater (PA6) is activated via PWM to provide proportional heating.
4. **Lighting:** If ambient light drops below a certain level, the Lighting LED (PA7) brightness increases via PWM.

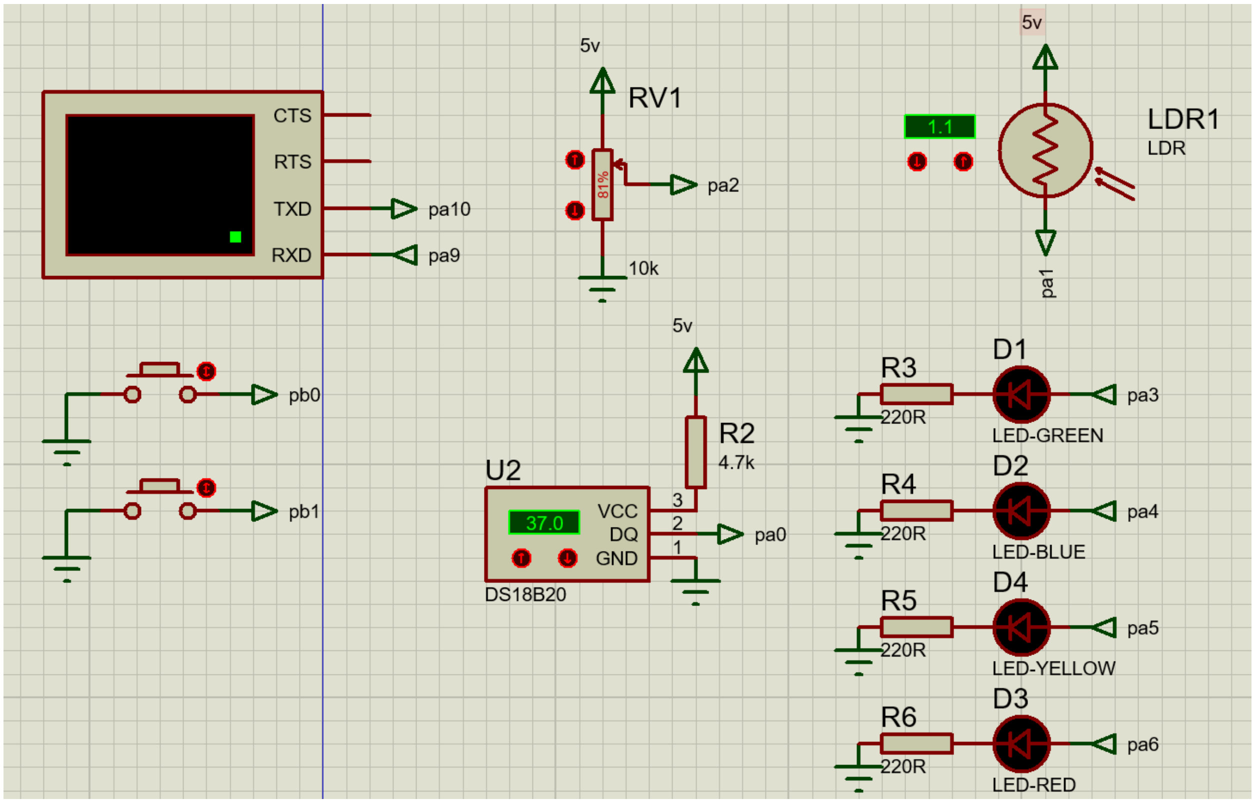
C. Manual Logic (MANUAL Mode)

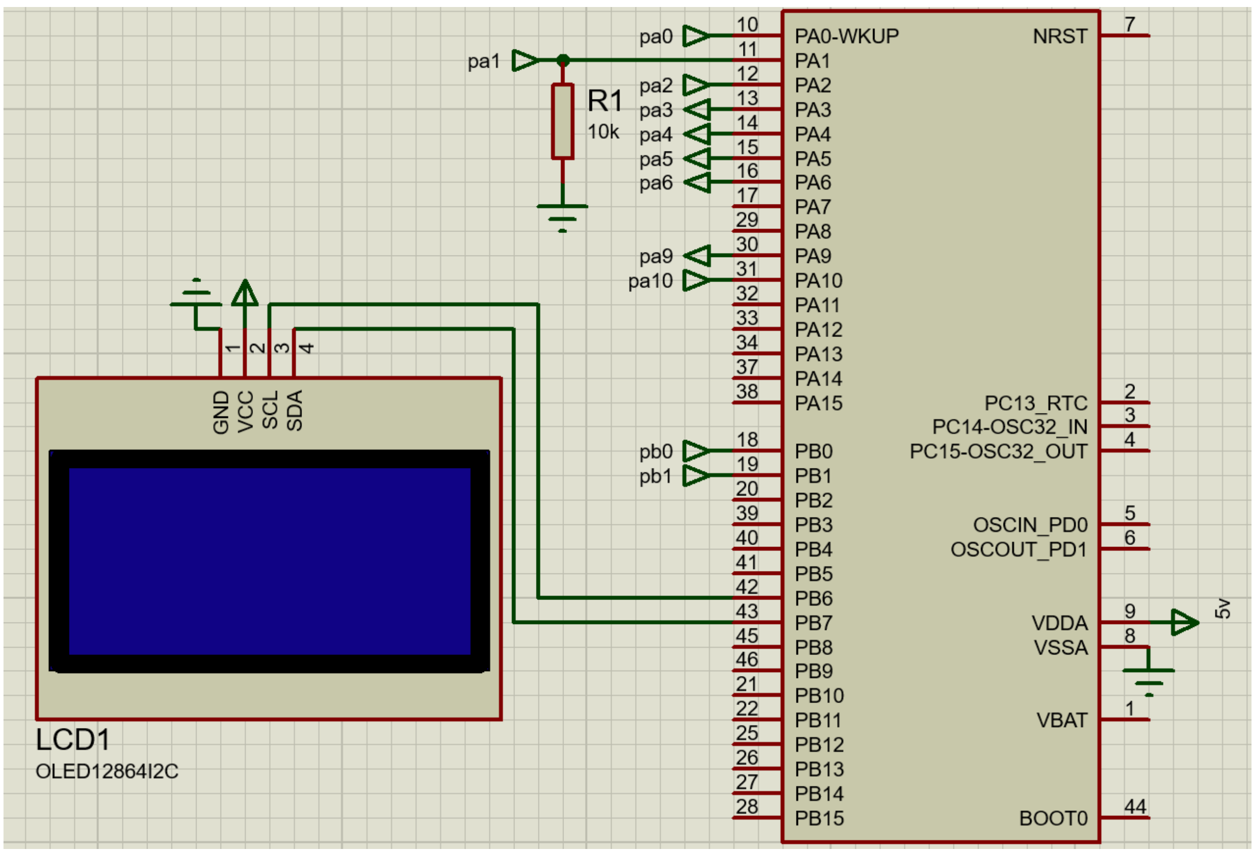
Users can override the system via the UART terminal. By sending specific characters (e.g., 'F' for Fan, 'P' for Pump), the user can directly toggle actuators while monitoring sensor feedback on the terminal screen.

---

5. Circuit Connection Mapping (Proteus)

- **PA0:** DS18B20 Data line (with 4.7K Pull-up resistor).
- **PA1 & PA2:** Analog inputs for LDR and Potentiometer.
- **PA3 & PA4:** Digital outputs for Fan and Pump LEDs.
- **PA6 & PA7:** PWM outputs for Heating and Lighting systems.
- **PB6 & PB7:** I2C lines (SCL & SDA) for the OLED Display.
- **PA9 & PA10:** UART TX and RX lines connected to the Virtual Terminal.
- **PB0 & PB1:** Push-buttons for Mode switching and Door status simulation.


CTS
RTS
TXD
RXD
pa10
pa9
5v
RV1
81%
pa2
10k
5v
1.1
LDR1
LDR
pa1
pb0
pb1
5v
U2
37.0
VCC
DQ
GND
DS18B20
R2
4.7k
pa0
R3
220R
D1
LED-GREEN
pa3
R4
220R
D2
LED-BLUE
pa4
R5
220R
D4
LED-YELLOW
pa5
R6
220R
D3
LED-RED
pa6


pa0
pa1
R1
10k
pa2
pa3
pa4
pa5
pa6
pa9
pa10
pb0
pb1
GND
VCC
SCL
SDA
LCD1
OLED12864I2C
PA0-WKUP
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
NRST
PC13_RTC
PC14-OSC32_IN
PC15-OSC32_OUT
OSCIN_PD0
OSCOUT_PD1
VDDA
VSSA
VBAT
BOOT0
5v

6. Results and Conclusion

During the final simulation phase in Proteus, the system encountered significant technical obstacles, and the expected visual output was not achieved on the OLED display. Post-simulation analysis revealed that the primary cause for the display failure was a protocol mismatch; the software functions implemented were designed for a 4-bit parallel Character LCD, whereas an I2C OLED module was utilized in the hardware schematic. Furthermore, the control actuators (LEDs) exhibited unstable behavior due to the precise timing requirements of the DS18B20 sensor within the simulated environment and conflicts in ADC polling priorities. Although the project did not produce a fully functional visual output, it provided critical insights into STM32 configuration and highlighted the vital importance of ensuring strict compatibility between software drivers and hardware communication protocols.
