Politechnika Rzeszowska im. Ignacego Łukasiewicza

Wydział Elektrotechniki i Informatyki

Projekt na zaliczenie przedmiotu

Elektronika dla Informatyków (EDI)

Automatyczny system zarządzania klimatem (Growbox)

Kontrola parametrów rośliny z interfejsem WWW

Wykonał:
Jakub Jaworski
Nr albumu: 183848
Grupa: L04 1FDI

Rzeszów, 10 czerwca 2026

Spis treści

Spis listingów

1 Cel i założenia projektu

Celem projektu było zbudowanie układu, który sam dba o warunki w zamkniętej skrzyni do uprawy roślin (tzw. Growbox). Całością steruje mikrokontroler ESP32, który dodatkowo udostępnia prosty panel WWW do podglądu i zmiany ustawień.

Założenia projektu:

- Regulacja temperatury w komorze za pomocą grzałki i wentylatora w tunelu.
- Automatyczne podlewanie pompą perystaltyczną, tak aby nie przelać gleby.
- Panel WWW do podglądu odczytów na żywo i zmiany ustawień.
- Utrzymanie wilgotności powietrza generatorem mgły (część programowa i sterowanie przekaźnikiem działają; płytkę piezo spaliłem podczas pomiaru multimetrem — na obwód atomizera trafiło +24 V przez odwróconą polaryzację pompy, opis w rozdz. 7).

Rysunek 1: Zmontowany i działający prototyp Growboxa

2 Architektura systemu i wykorzystany sprzęt

Układ podzieliłem na część logiczną (pomiary i sterowanie) oraz część siłową (elementy wykonawcze). Głównym elementem jest mikrokontroler z wbudowanym WiFi.

Najważniejsze komponenty:

- **Mikrokontroler ESP32:** steruje całością i działa jako serwer WWW. Wybrałem go głównie ze względu na wbudowane WiFi.
- **Czujniki pomiarowe:** pojemnościowy czujnik wilgotności gleby oraz cyfrowe czujniki temperatury i wilgotności powietrza.
- **Elementy wykonawcze:** pompa perystaltyczna DC zasilana 24 V (podlewanie), generator mgły 5 V (nawilżanie) oraz grzałka i wentylator tunelu, oba na 12 V.
- **Moduły zasilania i sterowania:** przetwornica step-up XL6009 podbija napięcie dla pompy do 24 V. Atomizerem steruje moduł przekaźnika, a pompę, grzałkę i wentylator załączają moduły MOSFET.

2.1 Mikrokontroler ESP32

Głównym układem jest moduł ESP32-WROOM-32 (płytka DevKit V1). Ma wystarczającą moc do obsługi automatyki i serwera WWW, a dzięki wbudowanemu WiFi nie trzeba było dokładać osobnego modułu sieciowego.

Specyfikacja (wg dokumentacji producenta):

- Moduł: ESP32-WROOM-32 (DevKit V1), procesor dwurdzeniowy Xtensa LX6 32-bit, taktowanie do 240 MHz.
- Pamięć: 520 kB SRAM, 4 MB Flash (bez zewnętrznego PSRAM).
- Łączność: WiFi 802.11 b/g/n (2,4 GHz) oraz Bluetooth.
- Wyprowadzenia: GPIO z obsługą ADC, I^2C, SPI, UART, PWM.
- Napięcie logiki: 3,3 V (zasilanie modułu 5 V przez wbudowany stabilizator).

2.2 Mapowanie pinów mikrokontrolera ESP32

Poniższa tabela zestawia przypisanie funkcji do konkretnych wyprowadzeń GPIO. W projekcie celowo unikano pinów **GPIO 34–39**, które na ESP32 obsługują wyłącznie wejścia (ADC-only) i nie nadają się do sterowania elementami wykonawczymi.

GPIO	Funkcja	Interfejs	Uwagi
18	DHT22 (temp., wilg. powietrza)	Protokół jednoprzewodowy	Zasilanie 3,3 V
4	DS18B20 (temp. rury)	OneWire	Magistrala z pull-up
32	Wilgotność gleby	ADC1 (analogowy)	12 bitów, średnia z 15 próbek
5	Pompa perystaltyczna	MOSFET, Active HIGH	Silnik 24 V
19	Wentylator tunelu	MOSFET, Active HIGH	Zasilanie 12 V
23	Grzałka	MOSFET, Active HIGH	Zasilanie 12 V
15	Atomizer (przekaźnik)	Open-drain	VCC modułu 3,3 V, IN Active LOW
21	LCD SDA	I^2C (SDA)	Domyślna magistrala `Wire`
22	LCD SCL	I^2C (SCL)	Linia zegarowa magistrali
25	Enkoder CLK	INPUT_PULLUP	ISR, zbocze opadające
26	Enkoder DT	INPUT_PULLUP	Odczyt stanu w ISR
27	Enkoder SW	INPUT_PULLUP	Aktywny stan LOW

Tabela 1: Mapowanie pinów ESP32 w projekcie Growbox

2.3 Czujniki pomiarowe

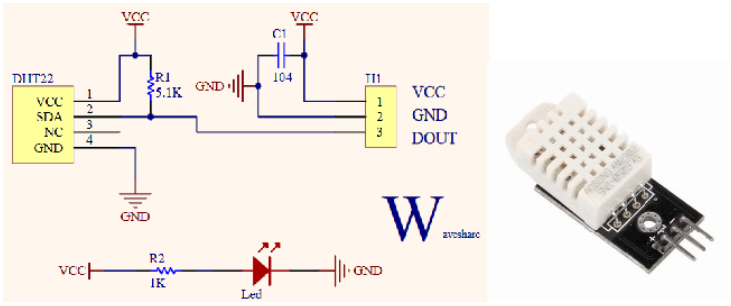
Dla każdego modułu poniżej zamieszczono schemat ideowy z dokumentacji producenta. Pełny układ połączeń w projekcie widoczny jest w schemacie Multisim (rozdz. 3).

2.3.1 Czujnik DHT22 (GPIO 18)

DHT22 to cyfrowy czujnik temperatury i wilgotności względnej powietrza. Mikrokontroler wysyła pojedynczy impuls startowy, a następnie odczytuje ramkę danych zawierającą wilgotność i temperaturę. Wartości są wykorzystywane w automatyce klimatu (grzałka, nawilżacz) oraz wyświetlane na LCD i w panelu WWW. Odczyt odbywa się cyklicznie co 2 s.

Specyfikacja (wg dokumentacji producenta):

- Napięcie zasilania: 3,3–6 V DC.
- Zakres temperatury: -40 do $+80\,^{\circ}$C, dokładność $\pm 0,5\,^{\circ}$C.
- Zakres wilgotności: 0–100% RH, dokładność ±2–5% RH.
- Interfejs: jednoprzewodowy protokół cyfrowy (proprietary 1-wire).
- Maksymalna częstotliwość odczytu: 0,5 Hz (co 2 s).



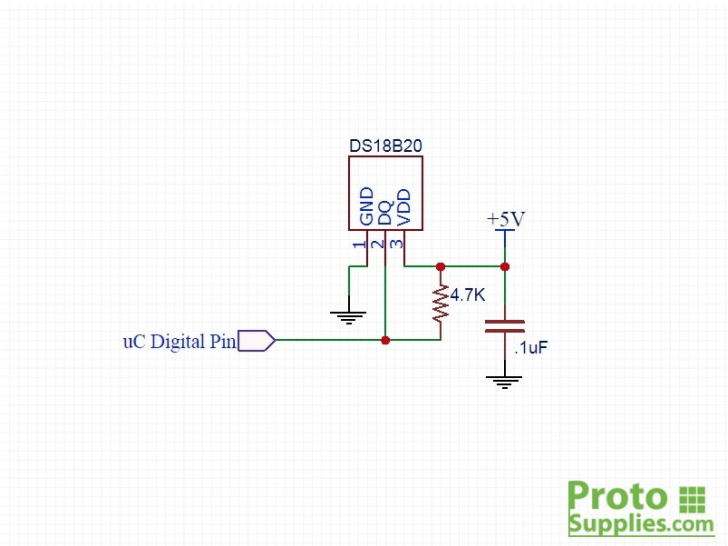
Rysunek 2: Schemat modułu DHT22 (źródło: `https://www.researchgate.net/figure/a-Schematic-diagram-of-DHT22-b-DHT22-module_fig2_332672437`)

2.3.2 Czujnik DS18B20 (GPIO 4)

Termometr cyfrowy pracujący na magistrali OneWire mierzy temperaturę w tunelu wentylacyjnym. Pozwala to oddzielić temperaturę wewnątrz komory od temperatury w kanale obiegu powietrza. Biblioteka `DallasTemperature` zarządza konwersją i odczytem wartości w stopniach Celsjusza.

Specyfikacja (wg dokumentacji producenta):

- Napięcie zasilania: 3,0–5,5 V DC.
- Zakres temperatury: -55 do $+125\,^{\circ}\mathrm{C}$, dokładność $\pm 0,5\,^{\circ}\mathrm{C}$ (w zakresie -10 do $+85\,^{\circ}\mathrm{C}$).
- Rozdzielczość: programowalna 9–12 bitów.
- Interfejs: OneWire (jedna linia danych z rezystorem pull-up); unikalny 64-bitowy adres.
- Możliwość pracy wielu czujników na jednej magistrali.



Rysunek 3: Schemat modułu DS18B20 (źródło: `https://protosupplies.com/product/ds18b20-waterproof/`)

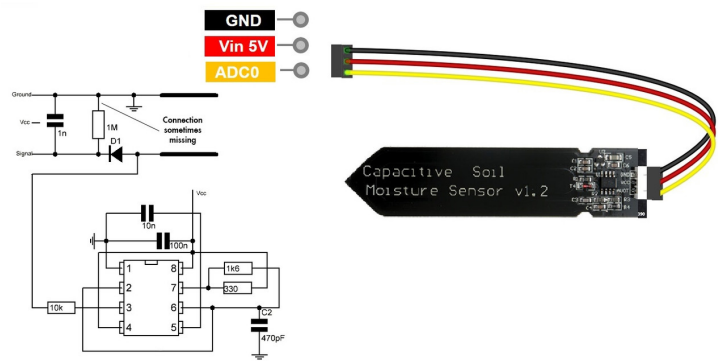
2.3.3 Czujnik pojemnościowy wilgotności gleby (GPIO 32)

Moduł mierzy przewodność pojemnościową podłoża i zwraca napięcie analogowe na wejście ADC. Im wyższa wilgotność gleby, tym niższa wartość odczytu ADC. W firmware zastosowano:

- **Średnią kroczącą** z 15 próbek — wygładza szumy i zakłócenia WiFi na ADC1.
- **Mapowanie liniowe** (`map`) z kalibracji: suche powietrze $\rightarrow$ 2699, woda $\rightarrow$ 1107, wynik 0–100%.
- **Debouncing czasowy** 5 s przed startem pompy – wilgotność musi utrzymywać się poniżej progu, zanim uruchomi się podlewanie.

Specyfikacja (wg dokumentacji producenta):

- Napięcie zasilania: 3,3–5,5 V DC.
- Wyjście: analogowe (napięciowe), odczyt przez 12-bitowy ADC ESP32.
- Zasada pomiaru: pojemnościowa (brak elektrod stykających się z glebą — odporność na korozję).
- Wartości kalibracyjne w projekcie: sucho $\approx$ 2699, woda $\approx$ 1107 (surowy odczyt ADC).



Rysunek 4: Schemat modułu czujnika wilgotności gleby (źródło: `https://www.techmaze.ae/shop/99187584-soil-capacitive-moisture-sensor-v12-21036`)

2.4 Elementy wykonawcze

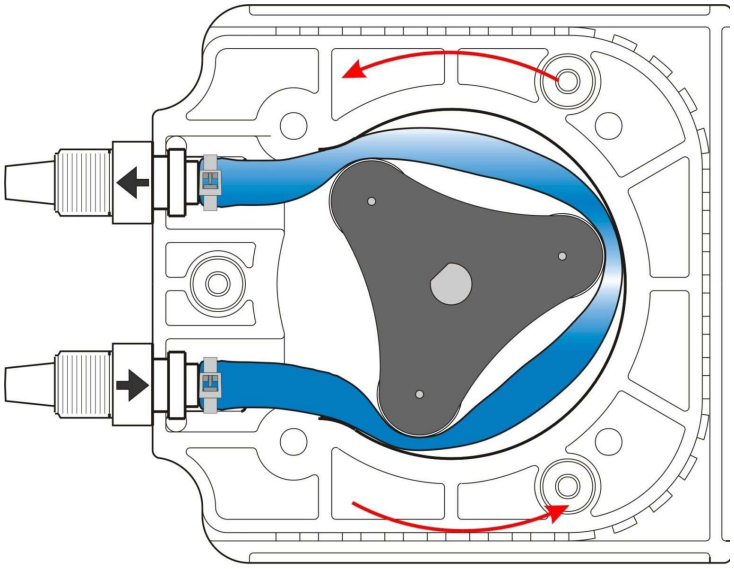
Analogicznie do czujników — poniżej schematy poszczególnych modułów wykonawczych; połączenia w całym systemie w Multisim (rozdz. 3).

2.4.1 Pompa perystaltyczna (GPIO 5, MOSFET)

Pompa DC pracuje w układzie jednokierunkowym. Tranzystor MOSFET kluczuje zasilanie 24 V silnika: stan `HIGH` na GPIO 5 włącza pompę, `LOW` ją wyłącza. W trybie AUTO firmware generuje krótkie impulsy podlewania (konfigurowalna długość w sekundach), z histerezą wilgotności gleby i cooldownem 60 s między cyklami.

Specyfikacja:

- Typ: pompa perystaltyczna z silnikiem DC (dozowanie jednokierunkowe).
- Napięcie pracy w projekcie: 24 V (z przetwornicy step-up XL6009).
- Sterowanie: moduł MOSFET D4184 (logic-level, Active HIGH) z pinu GPIO 5.



Rysunek 5: Widok modułu pompy perystaltycznej (źródło: `https://industriflo.com/collections/peristaltic-pumps`)

2.4.2 Grzałka i wentylator (GPIO 23 / 19, MOSFET)

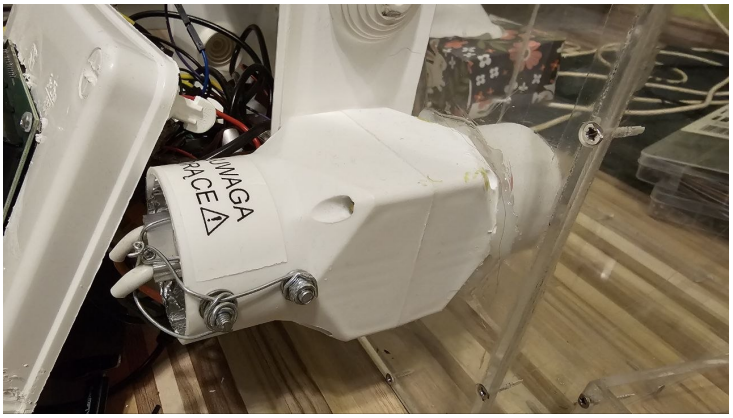
Oba elementy zasilane są z magistrali 12 V i sterowane identycznie jak pompa (Active HIGH). Grzałka włącza się, gdy temperatura spadnie poniżej progu docelowego z uwzględnieniem histerezy. Wentylator wymuszony jest przy pracy grzałki (interlock bezpieczeństwa) oraz realizuje cykliczną wymianę powietrza (2 h przerwy, 5 min pracy).

Specyfikacja:

- Napięcie zasilania (grzałka i wentylator): 12 V DC.
- Sterowanie: moduły MOSFET D4184 (logic-level, Active HIGH) z pinów GPIO 23 (grzałka) i GPIO 19 (wentylator).

- Logika: termostat z histerezą (grzałka), wymuszenie wentylatora przy grzaniu (interlock) oraz cykl wietrzenia 5 min co 2 h.

Rysunek 6: Zdjęcie modułu grzałki PTC (źródło: `https://nettigo.pl/products/grzalka-ptc-70c-12v`)



Rysunek 7: Zdjęcie modułu wentylatora (źródło: zdjęcie własne)

2.4.3 Atomizer ultradźwiękowy (GPIO 15, przekaźnik)

Do sterowania generatorem mgły wykorzystałem popularny, 1-kanałowy moduł przekaźnika. Jest to moduł typu Active LOW, w którym cewka nominalnie zasilana jest napięciem 5 V.

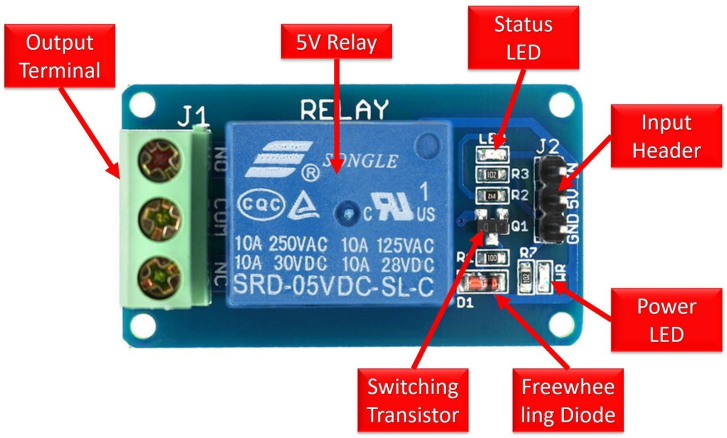
Podczas prób uruchomienia układu z mikrokontrolerem ESP32 pojawił się problem z napięciami. ESP32 pracuje na logice 3,3 V. Gdy mikrokontroler wystawiał stan wysoki (3,3 V), aby wyłączyć przekaźnik (który był zasilany z 5 V), układ wcale się nie wyłączał. Różnica napięć (pozostałe 1,7 V) była na tyle duża, że moduł wciąż interpretował to jako sygnał do załączenia. Aby szybko rozwiązać ten problem bez lutowania dodatkowych konwerterów napięć, po prostu przepiąłem zasilanie modułu przekaźnika na linię 3,3 V. Choć cewka przekaźnika jest przystosowana do 5 V, okazało się, że niższe napięcie w zupełności wystarcza, aby styki pewnie się zamykały, a problem z wyłączaniem całkowicie zniknął.

Płytkę samego przetwornika piezo niestety spaliłem podczas testów i pomiarów multimetrem. Mierzyłem napięcie na przekaźniku, ale miałem w tym czasie zamienioną polaryzację na pompie wody (podłączyłem ją odwrotnie, bo zasysała wodę zamiast tłoczyć).

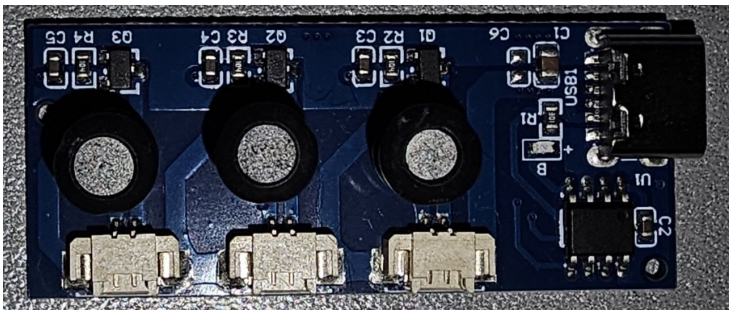
Przez ten błąd w plątaninie kabli, na 5-woltowy obwód atomizera trafiło nagle +24 V z zasilacza pompy, uszkadzając elektronikę generatora (więcej o tym w rozdz. 7). Sama warstwa sterowania przekaźnikiem z poziomu ESP32 po obniżeniu napięcia działa jednak całkowicie stabilnie.

Specyfikacja:

- Atomizer: ultradźwiękowy generator mgły na 5 V DC.
- Przekaźnik: prosty moduł 1-kanałowy, Active LOW (cewka 5 V).
- Sterowanie z ESP32 (GPIO 15): zasilanie modułu przekaźnika zostało obniżone do 3,3 V, co naturalnie zrównało poziomy logiczne i pozwoliło na poprawne wyłączanie obwodu.



Rysunek 8: Zdjęcie modułu przekaźnika optoizolowanego (źródło: `https://microcontrollerslab.com/5v-single-channel-relay-module-pinout-working-interfacing-applications-datasheet/`)



Rysunek 9: Zdjęcie modułu atomizera ultradźwiękowego (źródło: zdjęcie własne)

2.5 Interfejs lokalny (LCD i enkoder)

2.5.1 Wyświetlacz LCD 20×4 (I^2C)

Ekran łączy się z ESP32 przez konwerter I^2C–PCF8574 zamontowany na tyle wyświetlacza. Dzięki temu zamiast wielu pinów GPIO wystarczą dwie linie magistrali: **SDA (GPIO 21)** i **SCL (GPIO 22)**.

Moduł LCD w projekcie posiada sprzętowy adres `0x27`. Wartość ta została zdefiniowana w pliku `config.h` jako `PIN_LCD_ADDR` i jest bezpośrednio przekazywana do biblioteki obsługującej wyświetlacz.

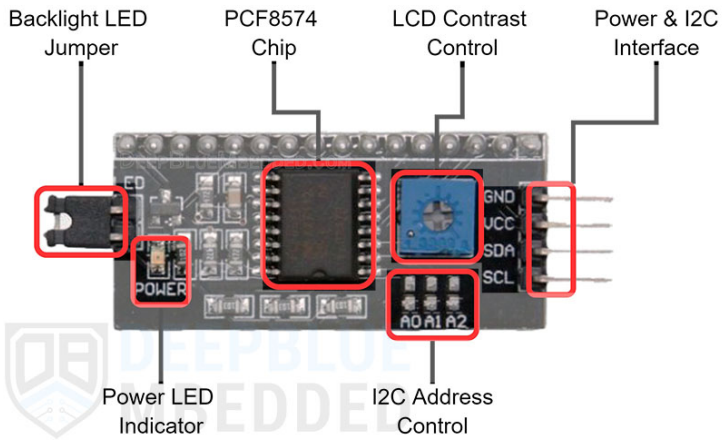
Specyfikacja:

- Wyświetlacz: alfanumeryczny 20×4 znaki, sterownik kompatybilny z HD44780.
- Konwerter: PCF8574 (I^2C), domyślny adres magistrali `0x27`.
- Napięcie zasilania: 5 V DC; linie sygnałowe podpięte pod sprzętowe piny I^2C mikrokontrolera.

Interfejs lokalny pracuje w trzech stanach logicznych: `DASHBOARD` (główny pulpit), `MENU` (przeglądanie opcji) oraz `EDIT` (edycja parametrów). Główny pulpit automatycznie sekwencyjnie przełącza trzy ekrany informacyjne co 15 s:

- **Ekran 0** — bieżąca temperatura i wilgotność powietrza z czujnika DHT22 wraz z wartościami progowymi.
- **Ekran 1** — odczyt temperatury kanału wentylacyjnego z czujnika DS18B20 oraz aktualny stan pracy grzałki, wentylatora i atomizera.
- **Ekran 2** — wilgotność gleby, zdefiniowany cel nawadniania oraz status pompy perystaltycznej.

Po 5 minutach bezczynności ze strony użytkownika podświetlenie ekranu zostaje automatycznie wyłączone w celu oszczędzania energii, a wybudzenie następuje po wykryciu jakiejkolwiek akcji na enkoderze.



Rysunek 10: Zdjęcie modułu konwertera magistrali I^2C dla wyświetlacza alfanumerycznego (źródło: `https://deepbluembedded.com/arduino-lcd-20x4-i2c-code-example-tutorial/`)

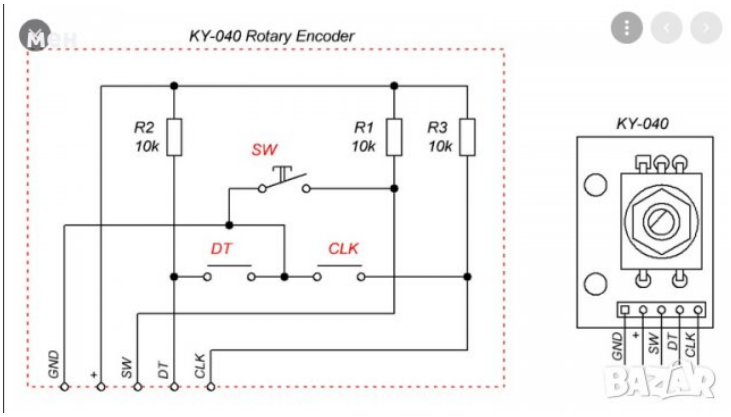
2.5.2 Enkoder obrotowy KY-040 (GPIO 25–27)

Enkoder służy do nawigacji po menu ustawień i zmiany progów automatyki. Żeby reagował od razu i nie obciążał pętli głównej (która zajmuje się też serwerem WWW), impulsy obrotu z pinu CLK obsługuję w przerwaniu sprzętowym wyzwalanym zboczem opadającym.

Wewnątrz funkcji obsługi przerwania sprawdzany jest stan pinu DT, co pozwala jednoznacznie określić kierunek obrotu gałki. W oprogramowaniu zaimplementowano debouncing programowy o długości 60 ms chroniący przed drganiami styków. Przycisk wbudowany w oś enkodera (SW) jest odczytywany cyklicznie w pętli głównej programu. Krótkie kliknięcie obsługuje potwierdzenia i natychmiastowe przełączanie ekranów, a przytrzymanie przez 3 sekundy pozwala na wejście lub wyjście z menu konfiguracyjnego.

Specyfikacja:

- Typ: inkrementalny enkoder obrotowy z przyciskiem (KY-040).
- Napięcie zasilania: 3,3–5 V DC.
- Wyprowadzenia: CLK (GPIO 25), DT (GPIO 26), SW (GPIO 27) — skonfigurowane jako `INPUT_PULLUP`.
- Obsługa obrotu: przerwanie zewnętrzne na pinie CLK, filtracja drgań styków (debouncing).



Rysunek 11: Schemat modułu enkodera obrotowego KY-040 (źródło: `https://deepblue mbedded.com/arduino-lcd-20x4-i2c-code-example-tutorial/`)

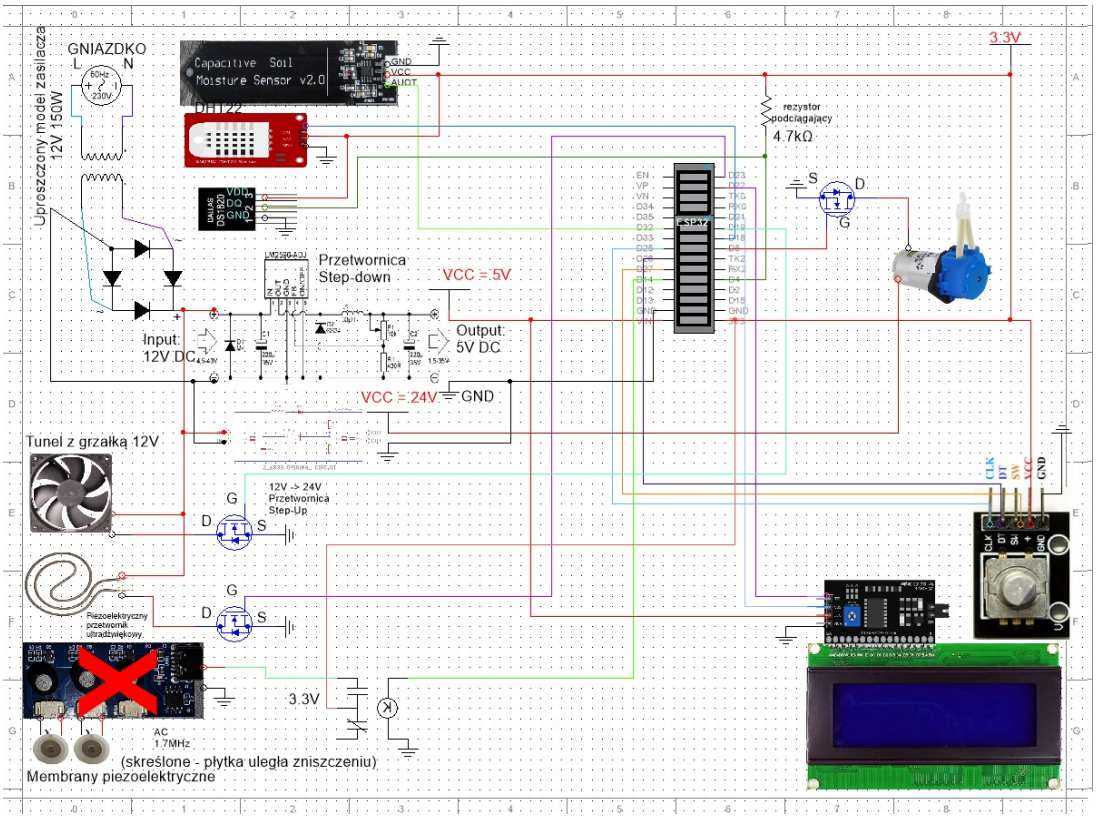
3 Schemat ideowy i dystrybucja zasilania

Głównym wyzwaniem przy zasilaniu tego układu był fakt, że poszczególne elementy wymagają zupełnie innych napięć roboczych. Tanie płytki ESP32 nie są przystosowane do zasilania bezpośrednio z 12 V, ponieważ wbudowany stabilizator napięcia uległby natychmiastowemu przegrzaniu i uszkodzeniu. Z tego powodu zasilanie musiało zostać rozdzielone na kilka osobnych szyn.

Głównym źródłem energii w systemie jest zasilacz 12 V o mocy 150 W. Z niego napięcie dystrybuowane jest w następujący sposób:

- **Szyna 12 V (główna):** bezpośrednie zasilanie elementów o największym poborze prądu — grzałki PTC oraz wentylatora kanałowego.
- **Szyna 24 V (przetwornica podwyższająca):** napięcie podbijane z 12 V osobnym modułem XL6009, przeznaczone wyłącznie dla silnika pompy. Przy standardowym zasilaniu 12 V silnik nie miał wystarczającego momentu obrotowego, by przełamać opór wężyka w pompie perystaltycznej w zadowalającej prędkości.

- **Szyna 5 V (przetwornica obniżająca):** oparta na module LM2596, który obniża główne napięcie z 12 V do 5 V. Z tej linii zasilana jest płytka ESP32 oraz wyświetlacz LCD.
- **Szyna 3,3 V (logika):** wyprowadzona z wbudowanego stabilizatora na płytce ESP32 (który zasilany jest z szyny 5 V). Linia ta zasila enkoder, czujniki oraz – co wynikało z problemów ze sterowaniem – moduł przekaźnika atomizera.



Rysunek 12: Schemat ideowy układu wykonawczego opracowany w programie Multisim (link do pobrania: `https://cloud.jjaworski.pl/s/6P4PjybjoPbEBYS`)

4 Etapy budowy i montaż prototypu

Proces fizycznej realizacji projektu podzielono na kilka kluczowych etapów, które łączyły obróbkę mechaniczną materiałów z precyzyjnym montażem elektroniki sterującej.

1. **Kompletowanie podzespołów i narzędzi:** Pierwszym etapem było zgromadzenie wszystkich niezbędnych układów elektronicznych, modułów wykonawczych, okablowania oraz narzędzi warsztatowych wymaganych do integracji systemu.

Rysunek 13: Zgromadzone podzespoły elektroniczne, okablowanie oraz narzędzia przed rozpoczęciem prac montażowych

2. **Konstrukcja komory uprawowej (Growbox):** Obudowę wykonano z przezroczystych płyt pleksi o grubości 0,5 cm, co zapewnia dostęp światła do rośliny. Elementy zostały precyzyjnie docięte do następujących wymiarów:

   - Ściany boczne węższe: 2 szt., 24 x 40 cm
   - Ściany boczne szersze: 2 szt., 25 x 40 cm
   - Podstawa: 1 szt., 24 x 24 cm
   - Pokrywa górna: 1 szt., 25 x 25 cm
   - Obramowanie pokrywy: 2 szt. 26,3 x 4 cm oraz 2 szt. 25,1 x 4 cm

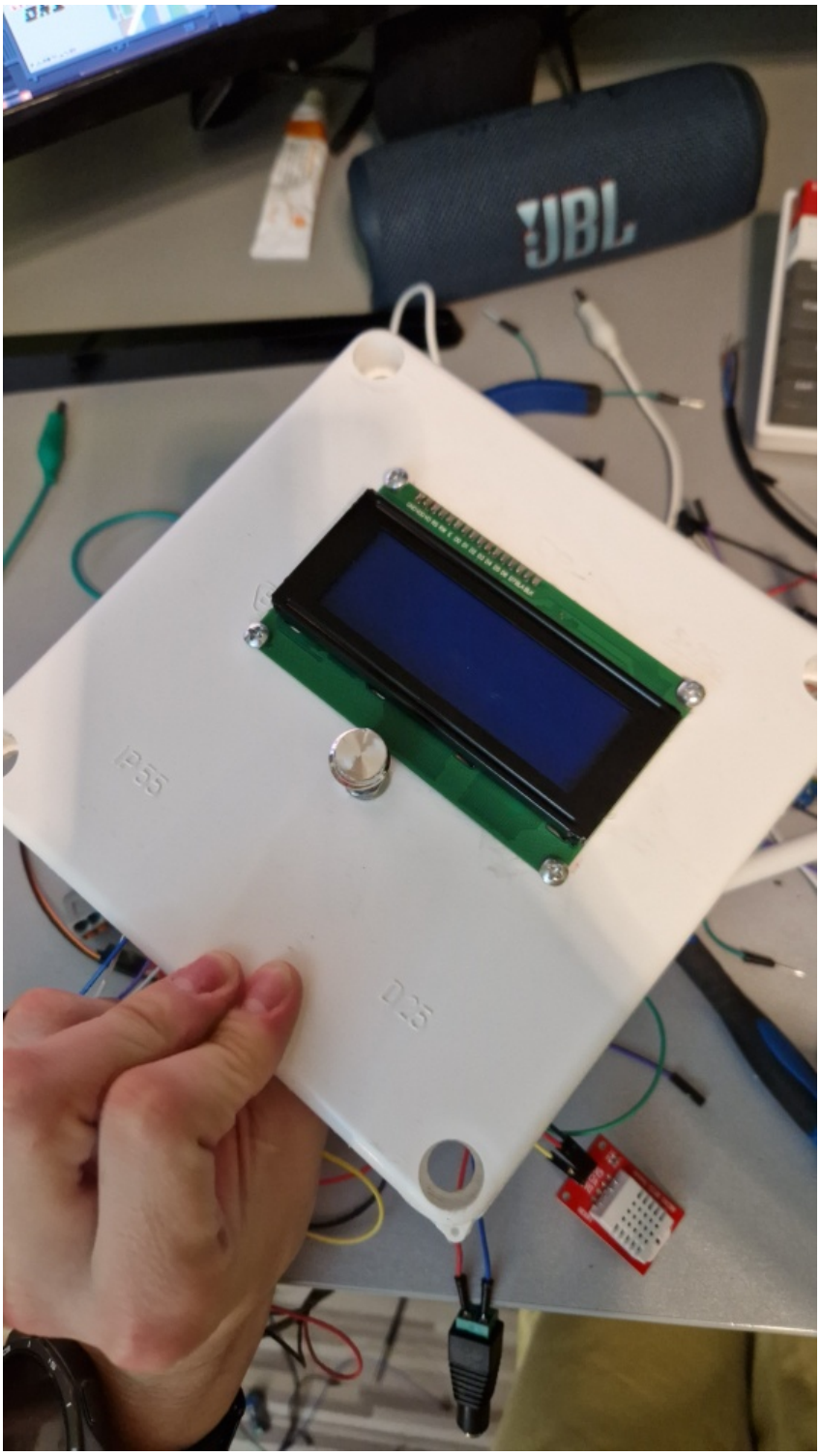
   Dla zapewnienia stabilności i szczelności konstrukcji, w krawędziach płyt nawiercono otwory i nagwintowano je. Całość skręcono niewielkimi wkrętami. Zastosowano również uszczelnienie klejem — proces ten wykonano etapami, pozwalając spoiwu wyschnąć przy jednoczesnym ustabilizowaniu mechanicznym ścian przez wkręty.

Rysunek 14: Zmontowana, szczelna obudowa główna wykonana z płyt pleksi

3. **Budowa tunelu grzewczo-wentylacyjnego:** Kolejnym krokiem było przygotowanie izolowanego kanału wymuszającego obieg powietrza. Wewnątrz zintegrowano grzałkę PTC oraz wentylator kanałowy.

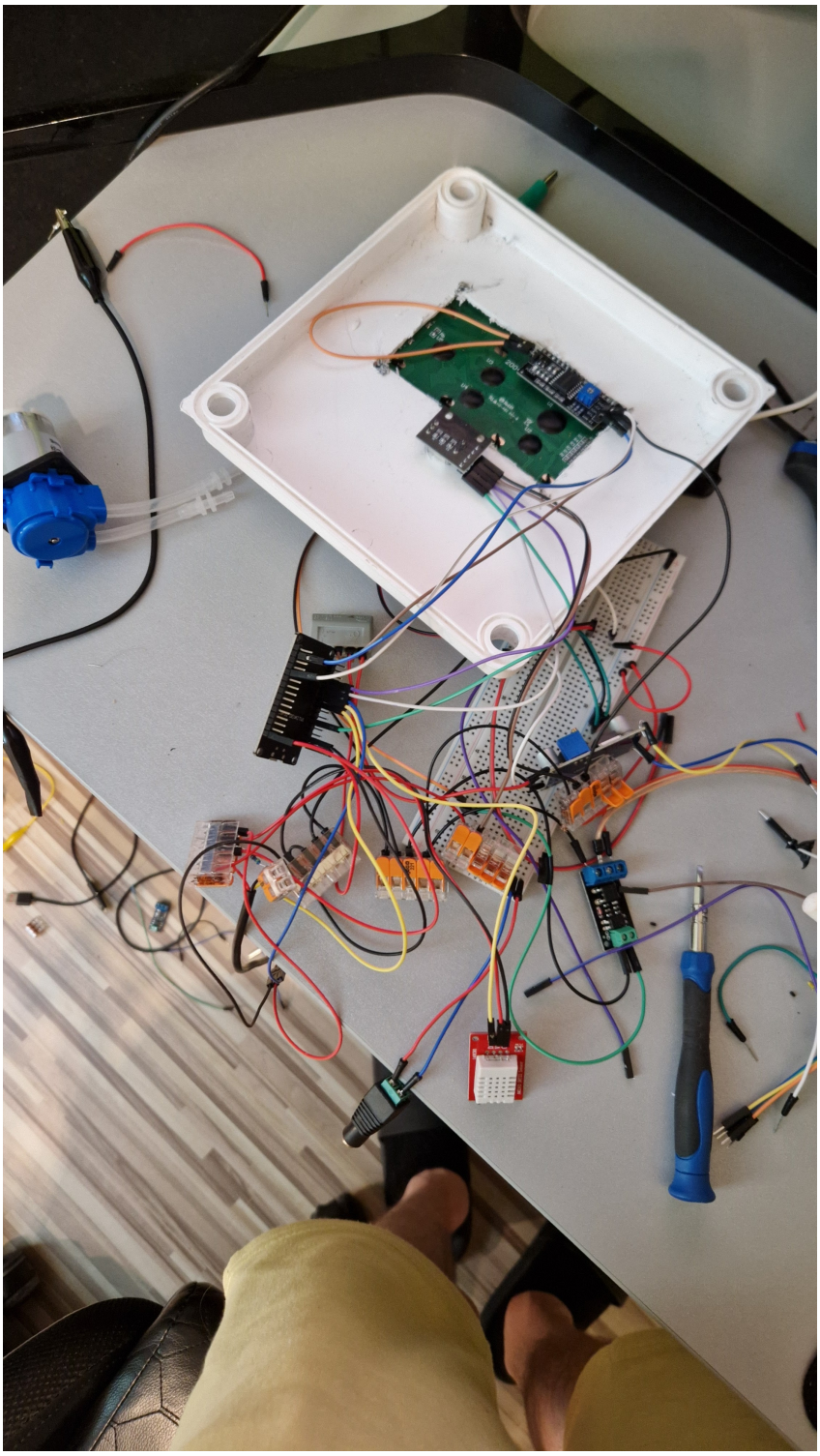
Rysunek 15: Wnętrze zmontowanego tunelu wentylacyjnego z widoczną grzałką PTC i okablowaniem

4. **Przygotowanie panelu frontowego:** W pokrywie natynkowej puszki instalacyjnej (pełniącej rolę obudowy dla jednostki centralnej) wycięto otwory montażowe. Zamocowano w nich interfejs lokalny użytkownika: wyświetlacz LCD oraz enkoder obrotowy.



Rysunek 16: Panel frontowy elektroniki sterującej z osadzonym ekranem LCD oraz gałką enkodera

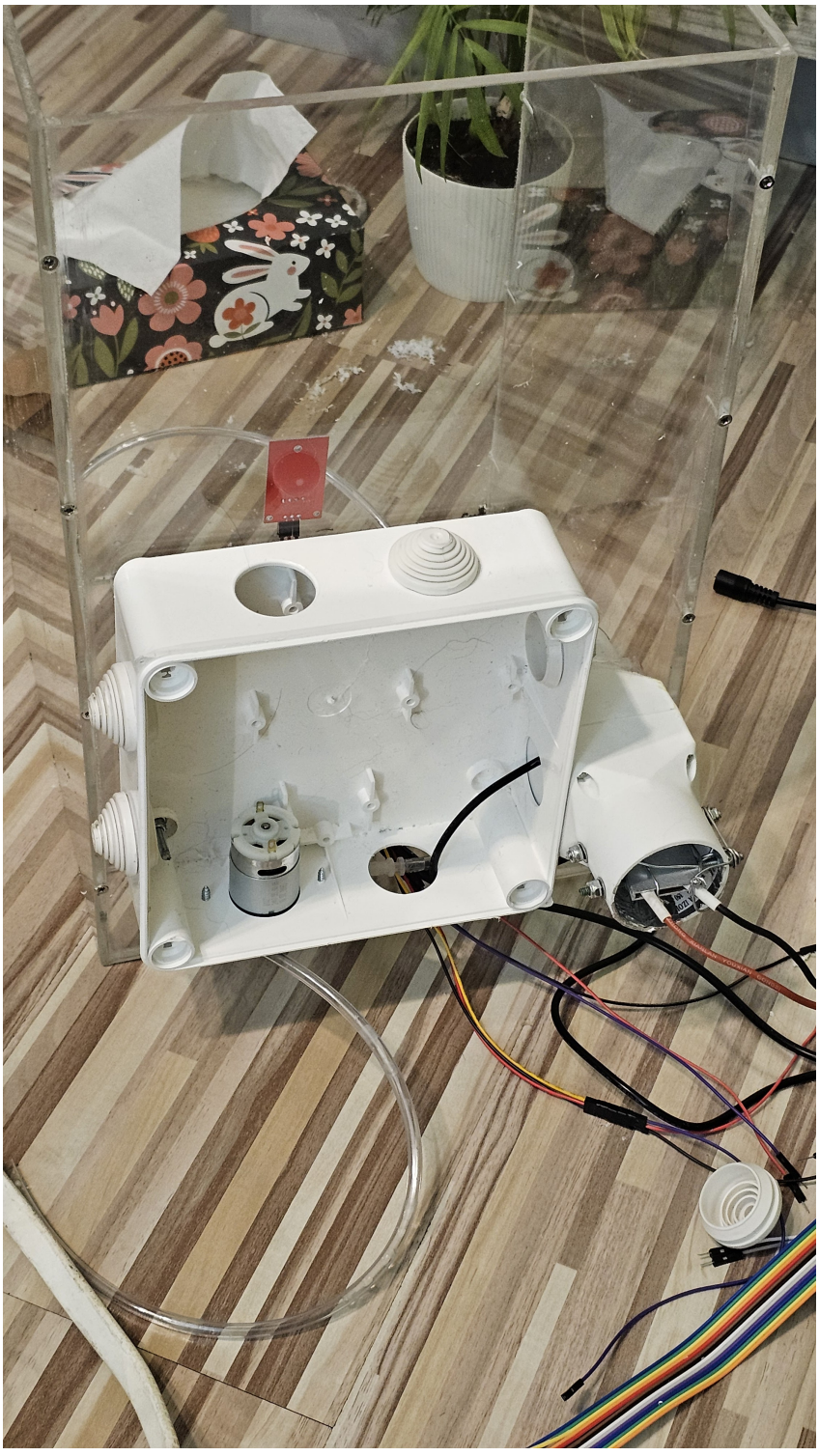
5. **Rozwój oprogramowania i migracja połączeń:** Ostatnim etapem przed ostatecznym zamknięciem obudowy było tworzenie i testowanie wczesnych wersji oprogramowania. Po weryfikacji logiki na płytkach stykowych, układ przeniesiono na docelowe okablowanie, wykorzystując niezawodne szybkozłączki instalacyjne typu WAGO.



Rysunek 17: Wczesny etap testowania firmware'u i migracja przewodów z płytki stykowej na złączki WAGO

6. **Montaż mechaniczny obudowy elektroniki i elementów wykonawczych:** Główna puszka natynkowa została trwale przymocowana do bocznej ściany z pleksi. Wykonano obok niej niezbędne otwory: pod kanał wentylacyjny (który po osadzeniu uszczelniono i ustabilizowano klejem na gorąco) oraz przepusty na okablowanie. Wewnątrz puszki zamontowano silnik pompy perystaltycznej, dbając o to, by same

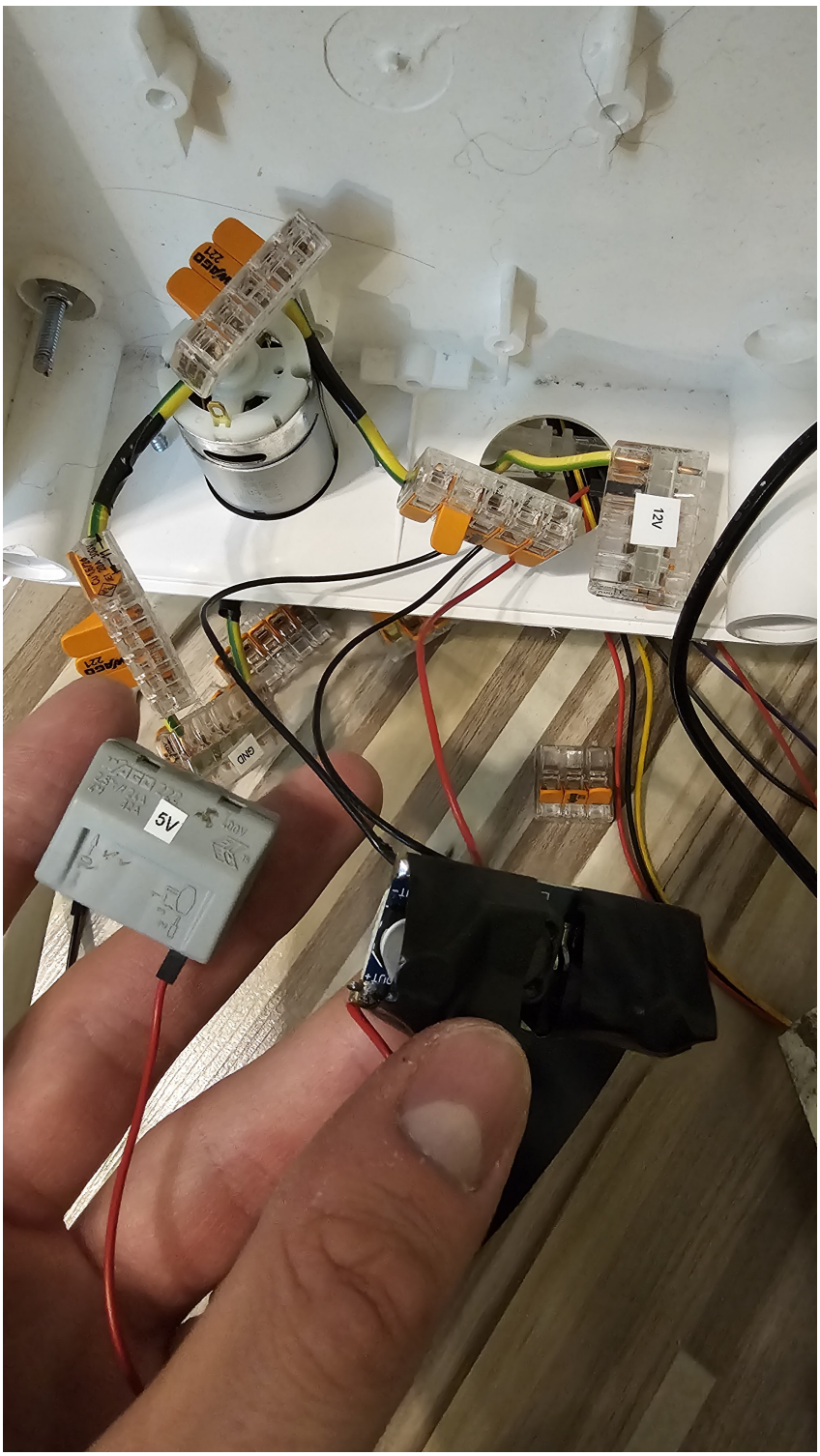
wężyki transportujące wodę poprowadzić wyłącznie na zewnątrz. Takie rozwiązanie całkowicie eliminuje ryzyko zalania wrażliwej elektroniki w przypadku ewentualnego rozszczelnienia układu nawadniania.



Rysunek 18: Puszka natynkowa przymocowana do obudowy z widocznym silnikiem pompy wewnątrz oraz wężykami wyprowadzonymi na zewnątrz

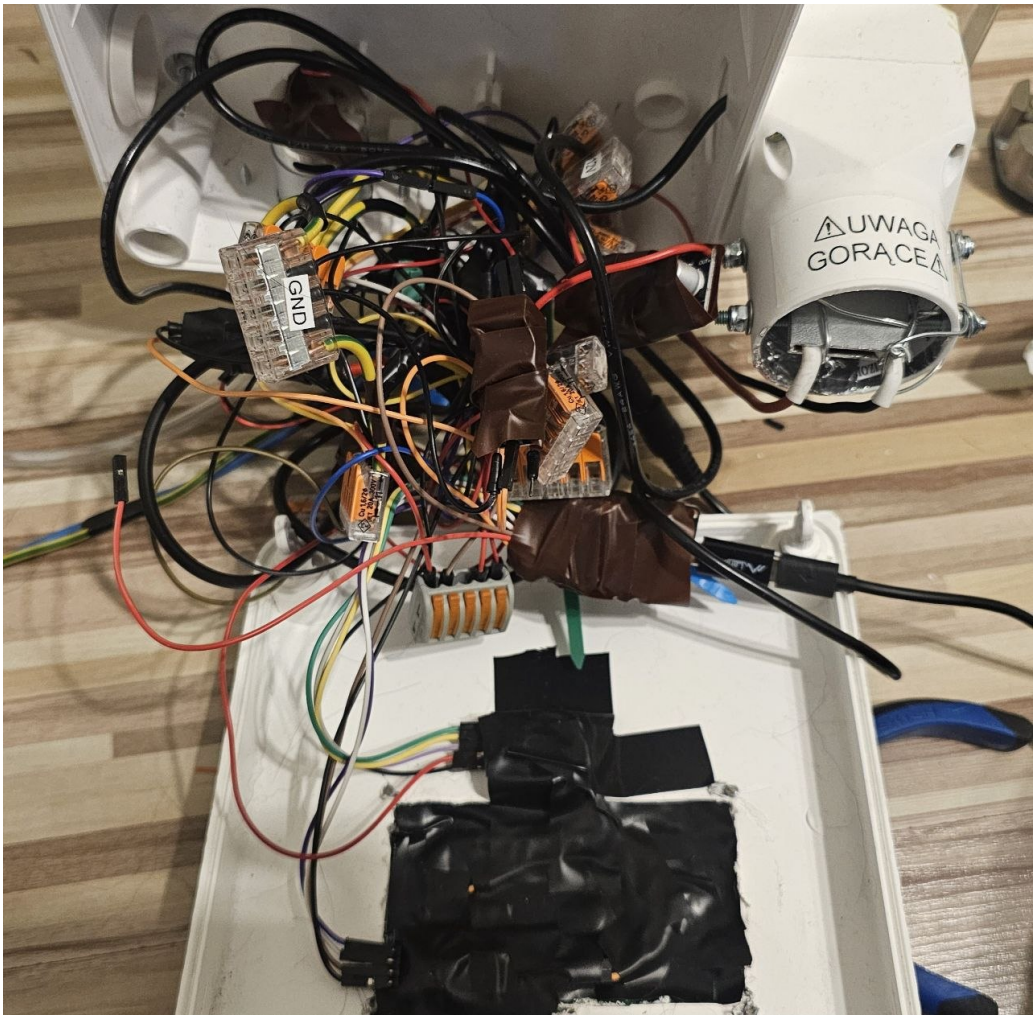
7. **Integracja zasilania i organizacja docelowego okablowania:** Prototyp zdemontowano z płytek stykowych, a wszystkie obwody przeniesiono bezpośrednio do

wnętrza docelowej puszki. Ze względu na stopień skomplikowania układu i obecność kilku szyn napięciowych (3,3 V, 5 V, 12 V, 24 V), do dystrybucji zasilania wykorzystano złączki instalacyjne WAGO. Każdy blok rozdzielczy został wyraźnie oznaczony dedykowanymi etykietami (np. `GND`, `5V`, `12V`), co pozwoliło zapanować nad okablowaniem i bezpiecznie wpiąć zaizolowane moduły przetwornic napięcia (step-up oraz step-down).



Rysunek 19: Organizacja szyn zasilających za pomocą oznaczonych złączek WAGO oraz moduł Step-Down

8. **Ostateczne zamknięcie obwodów w obudowie:** Po fizycznym podłączeniu wszystkich elementów wykonawczych, czujników oraz zasilania, cała wiązka przewodów wraz z modułami sterującymi została ciasno upakowana wewnątrz puszki. Zastosowanie elastycznych przewodów oraz kompaktowych złączek WAGO pozwoliło na ostateczne, bezpieczne domknięcie panelu frontowego z wyświetlaczem.



Rysunek 20: Kompletne okablowanie upakowane wewnątrz głównej obudowy docelowej przed jej zamknięciem

9. **Końcowy montaż i uruchomienie w pełni zmontowanego prototypu:** Ostatnim etapem prac było scalenie wszystkich przygotowanych wcześniej podsystemów w jedną, autonomiczną strukturę. Wewnątrz przezroczystej komory z pleksi umieszczono docelową roślinę wraz z osadzonym w podłożu pojemnościowym czujnikiem wilgotności gleby. Puszka instalacyjna mieszcząca kompletną elektronikę sterującą oraz interfejs lokalny została zamknięta i zabezpieczona. Zewnętrzne przewody pompy perystaltycznej wprowadzono do dedykowanego zbiornika z wodą, a cały system podłączono do głównego zasilacza impulsowego. Po zainicjalizowaniu mikrokontrolera ESP32 układ przeszedł w tryb ciągłego, automatycznego nadzoru nad mikroklimatem, poprawnie realizując zaprogramowaną logikę sterowania temperaturą, cyklicznym wietrzeniem oraz inteligentnym nawadnianiem.

Rysunek 21: W pełni zmontowany, uruchomiony i funkcjonujący prototyp automatycznego systemu zarządzania mikroklimatem (Growbox)

5 Oprogramowanie i logika sterująca

Oprogramowanie mikrokontrolera zostało napisane w języku C++ z wykorzystaniem środowiska PlatformIO w Visual Studio Code.

Kluczowe rozwiązania programowe:

- **Architektura nieblokująca:** Automatyka, czujniki, WiFi i UI działają w jednej pętli `loop()` opartej na `millis()`; logika sterująca nie używa blokujących `delay()` (wyjątek stanowią krótkie opóźnienia przy inicjalizacji Serial i debouncingu przycisku w trybie uśpienia).
- **Modularyzacja kodu:** `main.cpp` (pętla główna), `hardware.cpp` (wyjścia), `climate_control.cpp` wraz z modułami `climate_fan_heater`, `humidifier`, `pump_ctrl`, `ui_lcd.cpp` (LCD i enkoder), `server_routing.cpp` (WiFi i REST API), `storage.cpp` (Preferences), `sensors.cpp` i pliki czujników.
- **Tryby OFF/AUTO/ON:** Jednolity model sterowania dla wiatraka, grzałki, nawilżacza i pompy.
- **Histereza klimatu:** Konfigurowalne progi temperatury i wilgotności zapisywane w pamięci Flash.
- **Interlocki bezpieczeństwa:** Grzałka wymusza wiatrak; wiatrak blokuje nawilżacz; walidacja sprzecznych trybów w menu LCD i API WWW.
- **Filtrowanie zakłóceń ADC:** Średnia krocząca (15 próbek) oraz debouncing czasowy 5 s przed startem pompy.
- **Kalibracja gleby:** Mapowanie liniowe ADC na procenty na podstawie pomiarów w powietrzu i wodzie.
- **Mieszana logika wyjść i programowy Open-Drain:** Bramki tranzystorów MOSFET sterowane są klasycznie w trybie Active HIGH. Z kolei moduł przekaźnika atomizera obsłużono na pinie GPIO 15 przy użyciu programowej emulacji Open-Drain. Polega ona na dynamicznej zmianie trybu pracy pinu: włączenie przekaźnika realizowane jest przez ustawienie pinu jako wyjście w stanie niskim (`pinMode(15, OUTPUT); digitalWrite(15, LOW)`), co zwiera obwód do masy. Wyłączenie polega na przełączeniu pinu w tryb wejścia (`pinMode(15, INPUT)`), co sprzętowo wprowadza go w stan wysokiej impedancji (Hi-Z), skutecznie odcinając przepływ prądu i zwalniając linię. Sam atomizer podłączony jest pod linię 5 V poprzez zwierny styk NO przekaźnika.
- **Interfejs LCD + enkoder:** `DASHBOARD`/`MENU`/`EDIT`, automatyczne przełączanie ekranów, usypianie podświetlenia, ISR enkodera z odczytem rejestrów GPIO.
- **Panel WWW i REST API:** Podgląd na żywo, zmiana nastaw i trybów przez `/api/status`, `/api/settings`, `/api/setmode`.
- **Stabilne WiFi:** Serwer HTTP uruchamiany po uzyskaniu adresu IP; automatyczne ponawianie połączenia.
- **Diagnostyka Serial:** Raport stanu czujników i automatyki co 30 s (115200 baud, funkcja `logSensorsToSerial()`).

5.1 Parametry konfiguracyjne

Kluczowe stałe zebrano w pliku `include/config.h`, co pozwala dostroić działanie systemu bez ingerencji w logikę. Najważniejsze z nich zestawiono w tabeli 2.

Parametr	Wartość	Znaczenie
SENSOR_INTERVAL_MS	2 s	Okres odczytu czujników
SERIAL_LOG_INTERVAL_MS	30 s	Okres raportu diagnostycznego Serial
DASH_SWITCH_MS	15 s	Auto-przełączanie ekranów pulpitu LCD
LCD_SLEEP_MS	5 min	Czas do uśpienia podświetlenia LCD
VENT_INTERVAL_MS	2 h	Odstęp między cyklami wietrzenia
VENT_DURATION_MS	5 min	Czas trwania wietrzenia
PUMP_COOLDOWN_MS	1 min	Minimalna przerwa między podlewaniami
PUMP_SOIL_HYSTERESIS	5 %	Histereza wilgotności gleby
PUMP_SOIL_CONFIRM_MS	5 s	Potwierdzenie niskiej wilgotności przed startem pompy
SOIL_ADC_SAMPLES	15	Liczba próbek średniej kroczącej ADC
SOIL_ADC_DRY / WET	2699 / 1107	Kalibracja ADC (sucho / woda)
HTTP_PORT	80	Port serwera WWW

Tabela 2: Wybrane parametry konfiguracyjne z pliku `config.h`

5.2 Wybrane fragmenty kodu

Poniżej przedstawiono najważniejsze fragmenty firmware ilustrujące opisane wyżej rozwiązania.

5.2.1 Pętla główna

Cała logika (sieć, automatyka, czujniki, UI) wykonywana jest w jednej pętli `loop()` sterowanej znacznikami czasu `millis()`. Brak blokujących `delay()` sprawia, że serwer WWW i enkoder pozostają responsywne niezależnie od pozostałych zadań.

```
void loop() {
  unsigned long now = millis();

  handleGrowboxNetwork(growboxServer);    // obsluga zadan HTTP
  controlClimate();                       // automatyka klimatu

  if (now - lastSensorReadTime >= SENSOR_INTERVAL_MS) {
    readAllSensors();
    lastSensorReadTime = now;
    if (currentState == DASHBOARD && !isSleeping) drawDashboard();
  }

  if (now - lastSerialLogTime >= SERIAL_LOG_INTERVAL_MS) {
    logSensorsToSerial();
    lastSerialLogTime = now;
  }

  // ... obsluga enkodera, przycisku i uspienia LCD

  if (currentState == DASHBOARD && !isSleeping) {
    if (now - lastDashSwitchTime >= DASH_SWITCH_MS) {
      dashIndex = (dashIndex + 1) % 3;    // auto-przelaczanie ekranow
      lastDashSwitchTime = now;
      drawDashboard();
```

```
    }
  }
  yield();
}
```

Listing 1: Pętla główna — fragment (main.cpp)

5.2.2 Sterowanie atomizerem w trybie open-drain

Inicjalizacja pinu GPIO 15 jako wyjścia open-drain oraz funkcja przełączająca przekaźnik. Stan wysoki zwalnia linię (atomizer OFF), stan niski zwiera ją do masy (ON). Filtr `lastOn` eliminuje zbędne przełączenia.

```
// atomizer - open drain, HIGH=off (zwalnia linie)
gpio_reset_pin((gpio_num_t)PIN_ATOMIZERA);
gpio_set_direction((gpio_num_t)PIN_ATOMIZERA, GPIO_MODE_OUTPUT_OD);
gpio_set_level((gpio_num_t)PIN_ATOMIZERA, 1);

void setAtomizer(bool on) {
  static bool lastOn = false;
  if (!digitalPinCanOutput(PIN_ATOMIZERA)) return;
  if (on == lastOn) return;          // bez zmiany - nic nie rob
  lastOn = on;
  gpio_set_level((gpio_num_t)PIN_ATOMIZERA, on ? 0 : 1);
}
```

Listing 2: Konfiguracja i sterowanie open-drain (hardware.cpp)

5.2.3 Termostat z histerezą i interlock wentylatora

Logika grzałki i wentylatora w trybie AUTO. Histereza zapobiega częstemu przełączaniu wokół progu, a interlock wymusza pracę wentylatora podczas grzania.

```
// grzalka - termostat z histereza
if (heaterMode == 0) {
  isHeaterOn = false;
} else if (heaterMode == 2) {
  isHeaterOn = true;                 // tryb ON (reczny)
} else {
  if (currentTemp < (float)targetTemp - tempHysteresis) {
    isHeaterOn = true;
  } else if (currentTemp >= (float)targetTemp) {
    isHeaterOn = false;
  }
}

// interlock - grzalka zawsze ciagnie wiatrak
if (isHeaterOn) isFanOn = true;
```

Listing 3: Histereza klimatu i interlock (climate_fan_heater.cpp)

5.2.4 Obsługa enkodera

Stan pinów odczytywany jest bezpośrednio z rejestrów `GPIO.in` zamiast `digitalRead`, ponieważ wywołania bibliotek Arduino wewnątrz ISR potrafią resetować ESP32. Funkcja oznaczona jest atrybutem `IRAM_ATTR`, co umieszcza ją w pamięci RAM.

```
static int IRAM_ATTR readPinLevel(uint8_t pin) {
  if (pin < 32) return (GPIO.in >> pin) & 1;
  return (GPIO.in1.val >> (pin - 32)) & 1;
}

void IRAM_ATTR isrEncoder() {
  static uint32_t lastMs = 0;
  uint32_t nowMs = (uint32_t)(esp_timer_get_time() / 1000ULL);
  if (nowMs - lastMs < 60) return;   // debouncing 60 ms
  lastMs = nowMs;
  if (readPinLevel(PIN_ENC_DT) != readPinLevel(PIN_ENC_CLK))
     encoderCount++;
  else encoderCount--;
  activityFlag = true;
}
```

Listing 4: ISR enkodera z odczytem rejestrów GPIO (ui_lcd.cpp)

5.2.5 Filtrowanie odczytu wilgotności gleby

Surowy odczyt ADC jest uśredniany kroczącą tablicą (15 próbek), a następnie mapowany liniowo na procenty na podstawie kalibracji sucho/woda.

```
ring[ringIdx] = analogRead(PIN_GLEBA);
ringIdx = (ringIdx + 1) % SOIL_ADC_SAMPLES;
if (ringCount < SOIL_ADC_SAMPLES) ringCount++;

long sum = 0;
for (int i = 0; i < ringCount; i++) sum += ring[i];

lastRawSoil = (int)(sum / ringCount);
soilMoisture = map(lastRawSoil, SOIL_ADC_DRY, SOIL_ADC_WET, 0, 100);
soilMoisture = constrain(soilMoisture, 0, 100);
```

Listing 5: Średnia krocząca i kalibracja gleby (sensor_soil.cpp)

5.2.6 Logika podlewania — impuls z potwierdzeniem i cooldownem

W trybie AUTO pompa nie startuje natychmiast: niska wilgotność gleby musi utrzymać się przez 5 s (`PUMP_SOIL_CONFIRM_MS`), a po każdym impulsie obowiązuje przerwa 60 s (`PUMP_COOLDOWN_MS`). Zapobiega to przelaniu i drganiu stanu wokół progu.

```
// gleba ponizej progu musi sie utrzymac, zanim ruszy pompa
if (soilMoisture >= targetSoil) {
  soilDryConfirmStart = 0;
} else if (soilDryConfirmStart == 0) {
  soilDryConfirmStart = nowMs;
}

if (pumpPulseActive) {                  // trwa impuls podlewania
  isPumpOn = true;
  if (nowMs - pumpStartTime >= pumpDurationMs) {
    isPumpOn = false;
    pumpPulseActive = false;
    lastPumpCycleTime = nowMs;          // start cooldownu
  }
```

```
  return;
}

isPumpOn = false;
bool dryConfirmed = soilDryConfirmStart != 0 &&
                    (nowMs - soilDryConfirmStart >= PUMP_SOIL_CONFIRM_MS
                       );
if (dryConfirmed && soilMoisture < soilTrigger &&
    (lastPumpCycleTime == 0 || nowMs - lastPumpCycleTime >=
       PUMP_COOLDOWN_MS)) {
  pumpPulseActive = true;              // start nowego impulsu
  pumpStartTime = nowMs;
  isPumpOn = true;
  soilDryConfirmStart = 0;
}
```

Listing 6: Impuls podlewania w trybie AUTO — fragment (pump_ctrl.cpp)

5.2.7 Interlocki bezpieczeństwa

Aby uniknąć sprzecznych stanów urządzeń, menu LCD i API WWW przed zapisem trybu wymuszają reguły zależności: grzałka zawsze pociąga wentylator, a praca wentylatora wyklucza ręczne załączenie nawilżacza.

```
bool isHumidManualOnBlocked() {
  return isFanOn || fanMode == 2;      // wiatrak ON blokuje reczny
     nawilzacz
}

void enforceFanModeWithHeater() {      // grzalka wymusza wiatrak (min.
    AUTO)
  if ((heaterMode == 1 || heaterMode == 2) && fanMode == 0) {
    fanMode = 1;
    pamiec.putInt("trybWiatr", fanMode);
  }
}

void enforceHumidWithFan() {           // wiatrak wylacza reczny
    nawilzacz
  if (humidMode == 2 && isHumidManualOnBlocked()) {
    humidMode = 1;
    pamiec.putInt("trybNawilz", humidMode);
  }
}
```

Listing 7: Walidacja sprzecznych trybów (interlocks.cpp)

6 Wyzwania inżynieryjne i ich rozwiązania

Podczas budowy i uruchamiania prototypu napotkałem kilka problemów, które trzeba było rozwiązać:

- **Spalenie atomizera:** mierzyłem multimetrem napięcie na styku NO przekaźnika atomizera względem masy, ale miałem odwróconą polaryzację pompy — jej czarny przewód był wpięty do +24 V (inaczej silnik DC zasysał zamiast tłoczyć). Przez

to w trakcie pomiaru na 5-woltowy obwód atomizera trafiło +24 V i spaliło płytkę przetwornika piezo. Więcej w rozdz. 7.

- **Kapanie wody z pompy:** w spoczynku układ tracił podciśnienie i woda kapała. Początkowo chciałem to rozwiązać, tworząc funkcję odsysającą pozostałą wodę z wężyka za pomocą przekaźnika 2 kanałowego odwracając polaryzację pompki — zamiast tego sztywny wąż PVC 6 mm zamieniłem na ciaśniejszy wężyk silikonowy 4 mm.

- **Za mała moc pompy:** przy 12 V silnik działał wyjątkowo powoli. Jako iż na obudowie miał napisaną nominalną wartość napięcia 24V, dołożyłem przetwornicę step-up XL6009, która podbija napięcie do 24 V tylko dla pompy.

- **Uszkodzony moduł czujnika gleby:** przy pierwszym zamówieniu dostałem niesprawny pojemnościowy czujnik wilgotności do doniczki — odczyty były losowe albo moduł w ogóle nie reagował na zmianę wilgotności. Po wymianie na kolejny egzemplarz pomiar ADC i kalibracja zaczęły działać poprawnie.

- **Słabe szczotki w pompce perystaltycznej:** pierwsza pompa przyszła ze słabej jakości szczotkami — silnik praktycznie nie kręcił się, dopóki ręcznie nie nastawiłem ich na stojanie. Dopiero po tej regulacji mechanicznej pompa zaczęła tłoczyć wodę w sposób użyteczny do podlewania.

- **Różnica poziomów logicznych na przekaźniku:** przy VCC modułu z 5 V i sygnale sterującym 3,3 V z ESP32 przekaźnik (Active LOW) nie przełączał się pewnie — nie dało się go stabilnie wyłączyć. Zamiast konwertera poziomów przepiąłem zasilanie modułu na 3,3 V z ESP32 i ustawiłem GPIO 15 jako open-drain: stan wysoki zwalnia linię IN, stan niski zwiera ją do masy.

- **Zasilanie mikrokontrolera:** podpięcie ESP32 wprost pod główny zasilacz 12 V przegrzałoby stabilizator na płytce. Użyłem więc przetwornicy step-down LM2596, która obniża napięcie do 5 V dla ESP32 i peryferiów.

7 Znane ograniczenia prototypu

Poniższe ograniczenia nie dyskwalifikują projektu jako układu automatyki, ale powinny być jasno rozróżnione od założeń projektowych:

- **Nawilżacz (atomizer):** płytka przetwornika piezo została trwale uszkodzona podczas pomiaru multimetrem — przy odwróconej polaryzacji pompy (czarny przewód na +24 V, konieczne by silnik DC tłoczył wodę, a nie zasysał) na 5-woltowy obwód atomizera trafiło +24 V. Uszkodzony element usunąłem. Sam firmware, menu LCD, panel WWW i przekaźnik sterujący działają prawidłowo (układ fizycznie przełącza styki zgodnie z logiką) — pod kątem automatyki system jest sprawny, brakuje tylko końcowego efektu w postaci mgły.

- **Czujnik temperatury DS18B20:** system jest odporny na brak tego czujnika. Rozłączenie termometru na magistrali OneWire nie blokuje startu ani pętli głównej — na LCD i w logu Serial wartość rury pokazuje się wtedy jako FAIL, a DHT22 i czujnik gleby działają normalnie.

8 Kompilacja i uruchomienie układu

Proces budowania oraz kompilacji oprogramowania układowego realizowany jest w środowisku **PlatformIO** (zintegrowanym z edytorem Visual Studio Code) przy wykorzystaniu platformy `espressif32` oraz frameworku Arduino. Plik konfiguracyjny `platformio.ini` odpowiada za automatyczne pobranie i zdefiniowanie zależności zewnętrznych bibliotek (m.in. `DHT`, `DallasTemperature`, `LiquidCrystal_I2C`, `OneWire`).

Procedura wdrożenia oprogramowania na mikrokontroler przebiega według następujących kroków:

1. **Przygotowanie środowiska:** Kod źródłowy projektu należy otworzyć w edytorze z zainstalowanym i aktywnym rozszerzeniem PlatformIO.

2. **Konfiguracja sieciowa:** W strukturze katalogów wymagane jest utworzenie pliku nagłówkowego `include/secrets.h`, zawierającego definicje danych autoryzacyjnych lokalnej sieci WiFi:

   ```
   #define WIFI_SSID "nazwa_sieci"
   #define WIFI_PASS "haslo"
   ```

3. **Kompilacja i wgrywanie firmware:** Mikrokontroler ESP32 należy połączyć z komputerem programistycznym za pomocą interfejsu USB, a następnie uruchomić procedurę kompilacji i automatycznego transferu binarnego komendą: `pio run -t upload`.

4. **Weryfikacja diagnostyczna:** Uruchomienie wbudowanego monitora szeregowego o prędkości transmisji 115200 baud realizuje się poleceniem: `pio device monitor`. Po starcie systemu, cyklicznie co 30 s w terminalu generowany jest log diagnostyczny `GROWBOX dump`, obrazujący bieżące stany peryferiów oraz wartości pomiarowe.

5. **Dostęp do interfejsu sieciowego:** Po połączeniu modułu ESP32 z siecią WiFi, dostęp do panelu sterowania uzyskuje się poprzez wpisanie przypisanego adresu IP mikrokontrolera (port 80) w oknie przeglądarki internetowej. Wizualizacja danych odświeżana jest automatycznie w interwale 3 s.

9 Testy i działanie

Działanie prototypu sprawdzałem, obserwując stan na LCD, w panelu WWW oraz w logu na monitorze szeregowym (`logSensorsToSerial()`). Przetestowałem następujące przypadki:

- **Czujniki:** Odczyt DHT22 (temperatura, wilgotność powietrza) i czujnika gleby (ADC $\rightarrow$ procent wilgotności) co 2 s; wartości zgodne z warunkami w growboxie.
- **Automatyka klimatu:** Grzałka i wiatrak reagują na progi z histerezą; przy włączonej grzałce wiatrak jest wymuszony (interlock).
- **Podlewanie:** W trybie AUTO pompa uruchamia impuls po utrzymaniu niskiej wilgotności gleby przez 5 s; cooldown 60 s między cyklami.

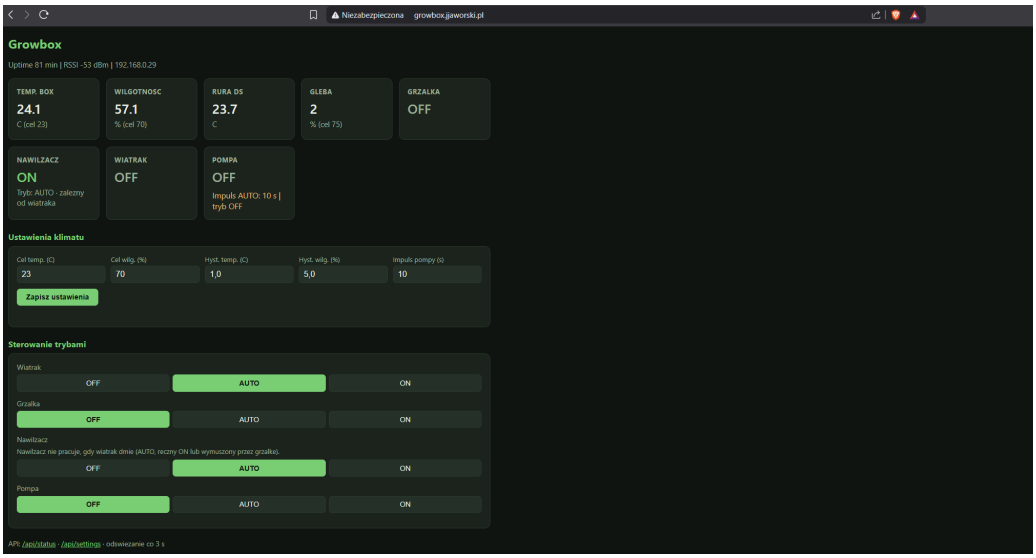
- **Interfejs lokalny:** Enkoder — nawigacja menu, edycja parametrów, zapis do Flash; LCD — trzy ekrany pulpitu, auto-przełączanie co 15 s, uśpienie podświetlenia po 5 min.
- **Panel WWW:** Endpointy `/api/status` i `/api/settings` zwracają aktualne dane JSON; zmiana trybów przez `/api/setmode` odzwierciedla się na LCD i w logu wyjść.
- **Nawilżacz:** Żądanie włączenia z automatyki lub WWW przełącza przekaźnik (potwierdzone na LCD i w linii `naw:ON` w Serial).

9.1 Dokumentacja wizualna

Poniższe zrzuty ekranu i zdjęcia pokazują działające interfejsy oraz zmontowany układ.

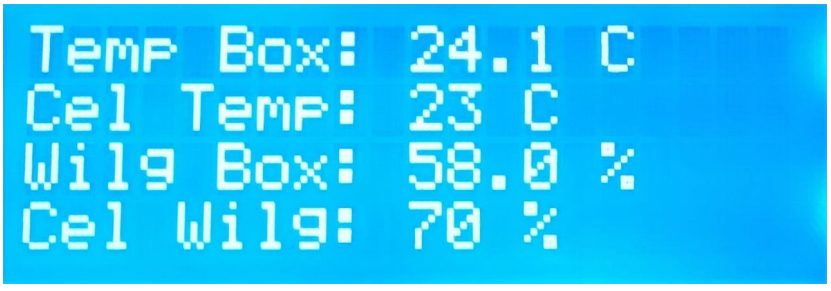
9.1.1 Panel WWW

Panel dostępny jest pod adresem IP ESP32 (port 80). Widać na nim bieżące odczyty czujników, tryby urządzeń (OFF/AUTO/ON) oraz formularz ustawień.

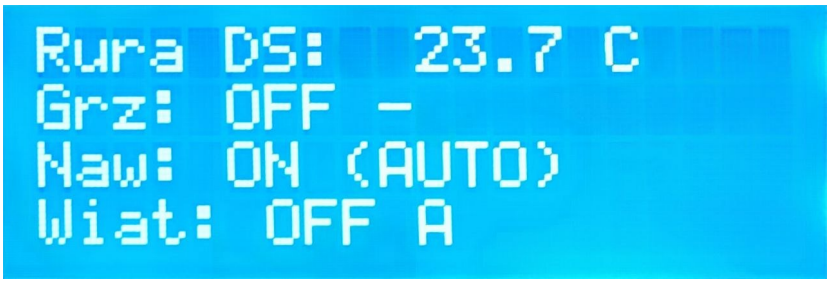


Rysunek 22: Panel WWW — podgląd na żywo i sterowanie urządzeniami

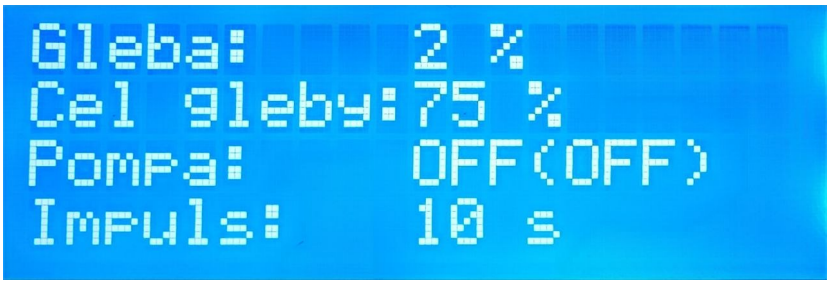
9.1.2 Wyświetlacz LCD



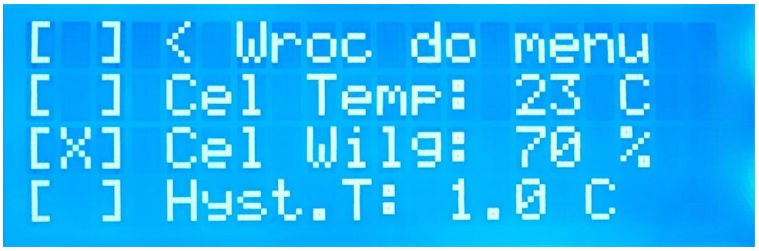
Rysunek 23: LCD — ekran pulpitu 0: temperatura i wilgotność powietrza z celami



Rysunek 24: LCD — ekran pulpitu 1: temperatura rury i stan grzałki, nawilżacza, wiatraka



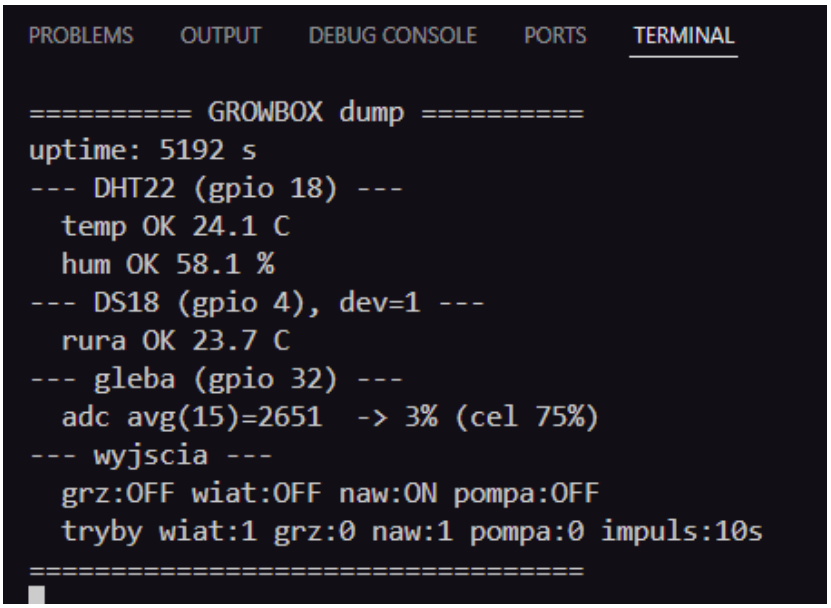
Rysunek 25: LCD — ekran pulpitu 2: wilgotność gleby, tryb i stan pompy



Rysunek 26: LCD — menu ustawień (nawigacja enkoderem)

9.1.3 Monitor szeregowy

Fragment logu z funkcji `logSensorsToSerial()` — widoczne odczyty DHT22, gleby, stan wyjść (`grz`, `wiat`, `naw`, `pompa`).



Rysunek 27: Monitor szeregowy (115200 baud) — zrzut GROWBOX dump

10 Podsumowanie

Udało mi się zbudować działający system, który realizuje większość założeń: reguluje temperaturę, automatycznie podlewa, pokazuje odczyty na LCD i w panelu WWW, a kod jest podzielony na moduły. Główne odstępstwo od pełnej automatyki to uszkodzony atomizer (rozdz. 7) — część sterująca i oprogramowanie są gotowe, wystarczy podłączyć sprawny przetwornik.

Projekt sporo mnie nauczył w praktyce — od podziału zasilania na osobne szyny (5 V / 12 V / 24 V), przez filtrowanie odczytów z ADC, po drobne problemy mechaniczne jak uszczelnienie pompy. Najwięcej dała mi nauka na własnych błędach, np. spalenie atomizera podczas pomiaru multimetrem przy odwróconej polaryzacji pompy.

Dzięki podziałowi kodu na moduły system łatwo rozbudować w przyszłości (wymiana atomizera, zapisywanie historii pomiarów czy integracja ze Smart Home).

Źródła

Literatura

[1] Espressif Systems, *ESP32 Series Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2] Aosong Electronics, *AM2302 (DHT22) Digital Temperature and Humidity Sensor — Datasheet*. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[3] Analog Devices (Maxim Integrated), *DS18B20 Programmable Resolution 1-Wire Digital Thermometer — Datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf

[4] *Capacitive Soil Moisture Sensor v1.2 — opis i podłączenie*. https://botland.com.pl/

[5] *XL6009 — przetwornica step-up (datasheet)*. https://www.haoyuelectronics.com/Attachment/XL6009/XL6009-DC-DC-Converter-Datasheet.pdf

[6] *D4184 — moduł MOSFET (logic-level, opis i parametry)*. https://botland.com.pl/

[7] Texas Instruments, *LM2596 Simple Switcher Power Converter — Datasheet*. https://www.ti.com/lit/ds/symlink/lm2596.pdf

[8] M. Burton, *Arduino-Temperature-Control-Library (DallasTemperature)*. https://github.com/milesburton/Arduino-Temperature-Control-Library

[9] P. Stoffregen, *OneWire Library*. https://github.com/PaulStoffregen/OneWire

[10] Adafruit, *DHT sensor library*. https://github.com/adafruit/DHT-sensor-library

[11] *LiquidCrystal_I2C Library*. https://github.com/johnrickman/LiquidCrystal_I2C

[12] *PlatformIO — dokumentacja platformy espressif32*. https://docs.platformio.org/en/latest/platforms/espressif32.html

[13] Espressif Systems, *ESP-IDF Programming Guide: GPIO & RTC GPIO*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html

[14] Espressif Systems, *Arduino Core for the ESP32 — Official Documentation*. https://docs.espressif.com/projects/arduino-esp32/en/latest/

[15] *Kurs elektroniki II — spis treści, praktyczne projekty*, Forbot. https://forbot.pl/blog/kurs-elektroniki-ii-spis-tresci-praktyczne-projekty-id10746

[16] RS Elektronika, *Przekaźniki - [RS Elektronika] # 16* (wideo). https://www.youtube.com/watch?v=0ca539ZJs50

[17] Electronics Tutorials, *MOSFET as a Switch*. https://www.electronics-tutorials.ws/transistor/tran_7.html