Dokumentation

B211 Drahtlose Netzwerke

Entwicklung eines sensorbasierten Smart-Garden-Systems mit Echtzeitdatenübertragung via MQTT und Datenverarbeitung via Node-RED

Autor
Ole Marwede (591748)

Abgabe
07.07.2025

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Hochschule für Technik und Wirtschaft Berlin
Wilhelminenhofstraße 75A
12459 Berlin

Inhaltsverzeichnis

1 Einleitung

Das folgende Dokument beschreibt die Realisierung eines sensorbasierten Smart-Garden-Systems mit Echtzeitdatenübertragung via MQTT und der Datenverarbeitung via Node-RED.
Die vorliegende Arbeit entstand im Rahmen der Lehrveranstaltung *Drahtlose Netzwerke* (des Studiengangs Angewandte Informatik) im vierten Semester des Bachelor-Studiengangs Internationale Medieninformatik an der HTW Berlin.

1.1 Projektziel

Ziel des Projektes war es, ein automatisiertes Gewächshaus zu bauen, welches relevante Umweltdaten für das Wachstum von Pflanzen regelt und überwacht. Außerdem soll das System bei nicht optimalen Bedingungen eingreifen, um diese zu verbessern.

Dabei waren zentrale Anforderungen zu erfüllen:

Integration von Sensoren und Aktoren:

Verschiedene Sensoren und Aktoren mussten miteinander abgestimmt werden, sodass das Smart Garden System mithilfe der Daten, eigenständig auf sich verändernde Umweltbedingungen reagieren kann.

Datenverarbeitung:

Die gesammelten Daten der Sensoren mussten verarbeitet und ausgewertet werden, um eine situationsabhängige Reaktion der Aktoren zu ermöglichen.

Fernüberwachung und Steuerung:

Es sollte dem Benutzer des Smart Garden Systems die Möglichkeit geboten werden, die gesammelten Daten und die Steuerung des Systems auch aus der Ferne zu überwachen.

Autonomie:

Das System sollte bestenfalls, wenn es einmal in Betrieb genommen wurde, ohne menschliche Eingriffe funktionieren.

2 Stand des Projektes

Dieses Projekt baut auf einem Vorgängerprojekt von Herr Lissé und Herr Hundertmark aus dem SoSe 2024 auf, welches bereits den kompletten Aufbau des Gewächshauses, sowie die Sensoren und Aktoren bereitstellte. Es ist jedoch hervorzuheben, dass der gesamte Programmcode und die komplette Logik neu entwickelt und umgesetzt werden musste.
Das gesamte Kabelsystem stellte eine mögliche Fehlerquelle dar, da es sehr unübersichtlich gestaltet war, deshalb wurde das alte Kabelsystem entfernt und durch eigens gelötete Neuverkabelungen ersetzt.

Somit wurde von mir, abgesehen vom physischen Aufbau des Gewächshauses, ein komplett neues System implementiert.

3 Projektaufbau

Der Projektaufbau sieht im Groben wie folgt aus:

Das Gewächshaus besteht aus einem abnehmbaren Deckel, in welchem 2 große Löcher für die Lüftung und 4 kleine für die Bewässerung geschnitten wurden. Für die Bewässerung reichen jedoch 2 Löcher aus, weshalb die anderen noch gestopft werden könnten. Die großen Löcher sind auf der einen Seite für den Lüfter und auf der anderen Seite für eine, mithilfe eines SG-90 Servomotors, öffenbare Klappe für die Luftzufuhr vorgesehen. Am oberen Dach des Deckels ist ein DHT-22 Sensor angebracht, welcher die Temperatur und Luftfeuchtigkeit misst. Der Boden wurde nicht weiter angepasst.

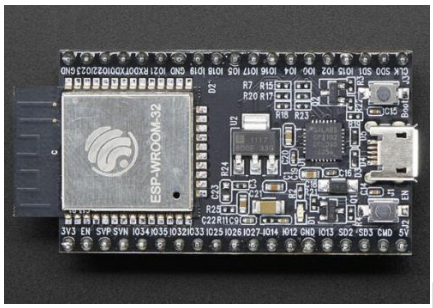
Zur Wasserversorgung ist ein externer Wasserkanister vorhanden, welcher 3 zusätzliche Löcher in seinen Deckel bekommen hat. Diese sind für den HC-SR04 Ultra Sonic Sensor, welcher den Wasserstand misst und für die Pumpe mit den Wasserschläuchen vorgesehen.

Im Inneren des Gewächshauses befindet sich ein Capacitive Soil Moisture Sensor 2.0, der für die Messung der Bodenfeuchtigkeit zuständig ist.

Das gesamte Gewächshaus steht auf einem Holzkasten mit einer Schublade, in welcher sich der Rest der Technik mit dem ESP-32, einer Relais Station und einem Breadboard befindet, welche sämtlichen Teile mit Strom versorgt und selber einem 12V Netzteil hängt.

4 Komponenten

4.1 ESP-32 Microcontroller



[11]

Der ESP-32 ist ein 32-Bit Mikrokontroller, der von der chinesischen Firma Espressif entwickelt wurde. Er ist WLAN und Bluetooth fähig und wird oft für Anwendungen im IoT Bereich verwendet [7].

Der ESP-32 dient als zentrales Bindeglied zwischen allen Sensoren und Aktoren des Smart Garden Systems. Er wurde verwendet, da er einen integrierten WiFi-Chip enthält und somit eine Verbindung mit dem Internet bereitstellt, was für die Verwendung von MQTT essentiell ist.

Der ESP-32 beinhaltet die Logik zum Auslesen der Sensordaten und zur Aktivierung der Aktoren. Die Programmierung erfolgte in Micropython, wobei als IDE Thonny verwendet wurde.

Der ESP sitzt außerdem auf einem Erweiterungsboard um die Verkabelung zu erleichtern und mehr Pins zu bieten.

4.2 Raspberry Pi 4

Der Raspberry Pi ist ein kleiner Einplatinencomputer, der mit einem Quad-Cores Prozessor, bis zu 8GB RAM und einer Vielzahl von Anschlüssen angeboten wird und damit in einer großen Anzahl von Anwendungen zu gebrauch kommt. [3,15,16]

Der Raspberry Pi dient lediglich der Ausführung des Node-RED Flowcharts und kann sehr leicht ersetzt werden.

4.3 Sensoren

4.3.1 DHT-22 Sensor


[4]

Der DHT-22 Sensor misst die Luftfeuchtigkeit und Temperatur innerhalb des Gewächshauses und befindet sich an dessen Decke.
Das Programmcodeschnipsel für den DHT-22 stammt von der offiziellen Micropython Website [12].

4.3.2 Capacitive Soil Moisture Sensor 2.0

Der Capacitive Soil Moisture Sensor 2.0 misst die Bodenfeuchtigkeit der Erde im Gewächshaus und steckt in der Erde.
Die Logik im Programmcode hinter dem Bodenfeuchtigkeitssensor und eine gute Erklärung der Ausgabewerte gibt es hier [8].

4.3.3 HC-SR04 Ultraschall Sensor

Der Ultraschall Sensor ist im Deckel des externen Wasserkanisters angebracht und misst den aktuellen Wasserstand. Die Programmlogik dahinter und die externe verwendete Bibliothek stammt von hier [10].

4.4 Aktoren

4.4.1 SG-90 Servomotor

[17]

Der SG-90 Servomotor wird verwendet um die Klappe des Gewächshauses zu öffnen und zu schließen. Er wird zusammen mit dem Lüfter betrieben und reguliert die Frischluftzufuhr für das Gewächshaus und regelt damit die Temperatur und Luftfeuchtigkeit.
Der Code kommt auch hier von der Mikropython Website [13].

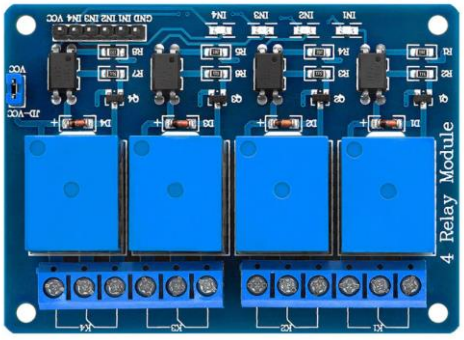
4.4.2 Lüfter

Der Lüfter ist laut alter Dokumentation ein Lüfter einer alten AMD-Grafikkarte. Er bläst Frischluft in das Gewächshaus rein und ist, wie schon erwähnt, mit dem Servomotor für die Frischluftzufuhr verantwortlich.
Der Lüfter wird über ein Relais angesteuert. Die Stromzufuhr ist das 12V Netzteil, was alle Komponenten versorgt.

4.4.3 Pumpe

Die Pumpe ist für die Bewässerung zuständig und pumpt Wasser aus dem externen Kanister durch einen Schlauch der Löcher hat, damit das Wasser gleichmäßig im Gewächshaus verteilt wird.
Die Pumpe ist zwar für 5V ausgelegt, kann aber auch mehr ab und wird somit mit allen Komponenten über das 12V Kabel über ein Relais angeschlossen.

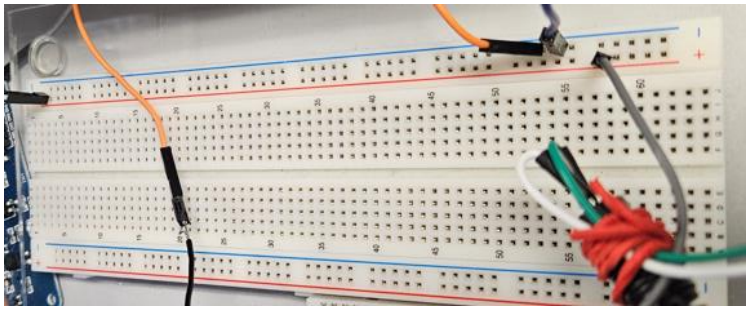
4.4.4 Relais



[1]

Das Relais stellt die Pumpe und den Lüfter an oder aus.
Der Programmcode kommt ebenfalls von der offiziellen Micropython Website [14].

4.4.5 Breadboard



Über das Breadboard sind alle Komponenten die direkt Strom beziehen müssen verbunden. Dazu gehören der ESP-32, der Lüfter und die Pumpe. Die restlichen Komponenten beziehen indirekt über den ESP-32 Strom.
Das gesamte System wird hierbei über ein 12V Netzteil versorgt.

4.5 Node-RED

Node-RED ist ein von IBM-Entwickeltes Browser basiertes, visuelles Programmierwerkzeug. Es basiert auf der Grundlage von Node.js und wird oft im IoT Bereich eingesetzt. Es ermöglicht Geräte, APIs und Online-Dienste miteinander zu verbinden [2,19].
Node-RED implementiert in meinem Projekt die gesamte Systemlogik.

4.6 MQTT

MQTT ist ein Nachrichten Protokoll für eingeschränkte Netzwerke mit geringer Bandbreite und IoT-Geräte mit hoher Latenz. Es wird meist für Machine-to-Machine Kommunikation verwendet [18].
In diesem Projekt werden sämtliche Daten die zwischen dem ESP-32 und dem Raspberry-Pi (also Node-RED) ausgetauscht werden, über MQTT verschickt.

5 Umsetzung

Der Gesamte Code, eine Startanleitung und die Dokumentation sind auf diesem GitHub Repo einsehbar: https://github.com/Elociraptor/dln-smartgarden

5.1 Netzwerk und Kommunikation

Die Systemkommunikation läuft vollständig über WLAN unter Verwendung des MQTT-Protokolls.

WLAN-Kommunikation:
Beim Booten verbindet sich der ESP-32 Automatisch mit dem Netzwerk

„Rechnernetze“. Auch der Raspberry-Pi ist über dieses Netzwerk verbunden.

MQTT-Client:
Daraufhin wird auf dem ESP-32 ein MQTT-Client erstellt, der sich mit dem MQTT-Broker der Hochschule (broker.f4.htw-berlin.de) verbindet. Die Bibliothek umqttsimple.py [9] wird für die Erstellung des Clients und die Verbindung zum Broker benötigt.

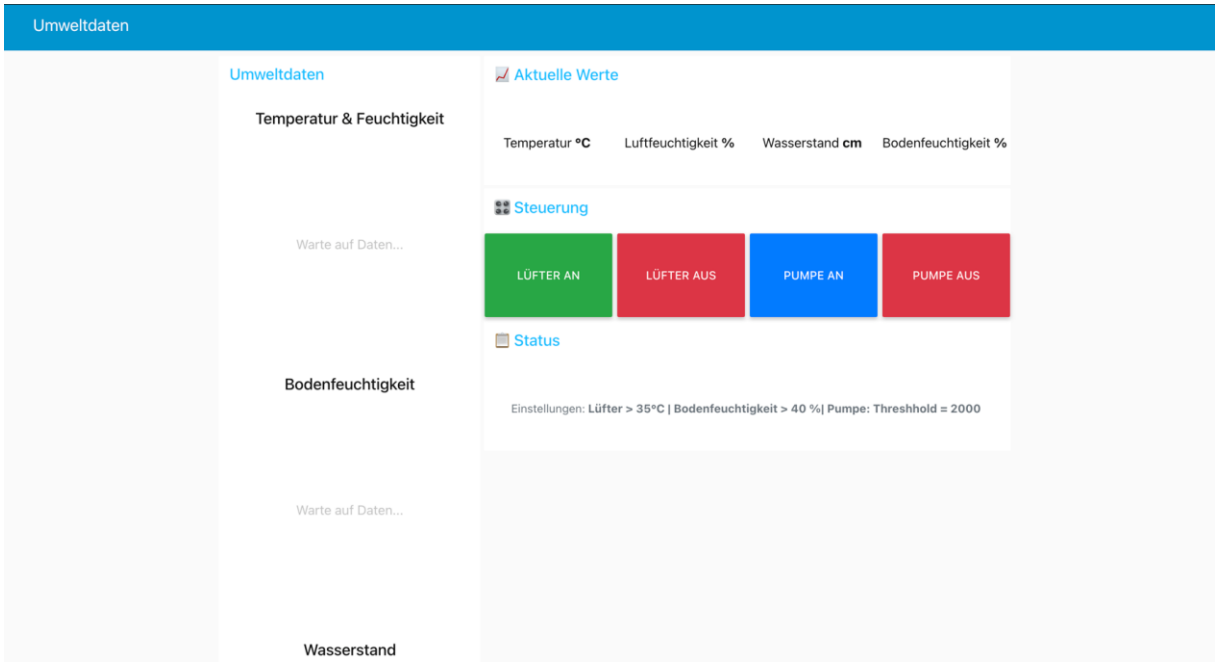
Datenversand:
Der ESP-32 sendet alle 10 Sekunden sämtliche Sensordaten an das jeweilige MQTT-Topic z.B. DLN/test/temp für die Temperatur. Alle einzelnen Topics sind im Anhang aufgelistet.

Dateninterpretation und Steuerung:

Der Raspberry-Pi abonniert die entsprechenden Topics mithilfe seines Node-RED Flowcharts und interpretiert und speichert die Werte dann in 3 verschiedenen Diagrammen (eins für Temperatur und Luftfeuchtigkeit, eins für die Bodenfeuchtigkeit, eins für den Wasserstand). Werden bestimmte Schwellwerte erreicht, wird eine Reaktion ausgelöst und es wird eine Nachricht, an ein auf die Aktoren zugeschnittene MQTT-Topic (siehe Anhang) gesendet. Der ESP hat dieses Topic abonniert und reagiert dann entsprechend, indem er z.B. die Pumpe anschaltet.

Benutzeroberfläche:

Um die eben erwähnten Diagramme und die aktuellen Daten einzusehen, muss man mit dem Rechnernetzte WLAN verbunden sein und folgende IP im Browser aufrufen: http://10.10.4.156:1880/ui.



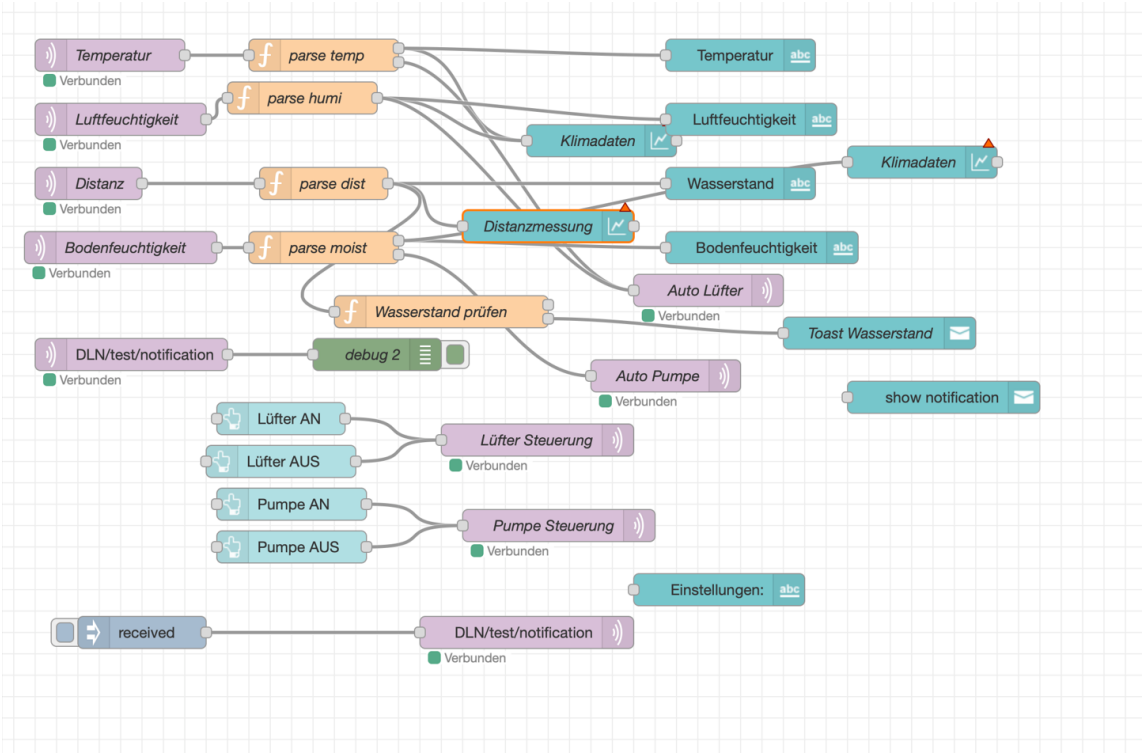
Ein Bild der aktuellen UI mit 3 Diagrammen (eins ist unten abgeschnitten), den aktuellen Werten, 4 Buttons und den aktuellen Einstellungen. (Ohne Werte, da gerade keinen Zugriff auf Rechnernetze WLAN)

5.2 Steuerung der Komponenten mit Micropython

Der Code, der auf dem ESP-32 läuft, ist mit Micropython in der Thonny IDE verfasst worden.
Der Code ist gut kommentiert und kann auf meinem GitHub [5] eingesehen werden.

5.3 Datenverarbeitung und Logik mit Node-RED

Die Gesamte Logik des Systems wird in dem Node-RED Flowchart implementiert, welches auf dem Raspberry-Pi läuft.
Eine Json Datei mit dem gesamten Flowchart, ist auf meinem GitHub einsehbar [6].



Node-RED Flowchart des Smart Garden Systems

Luftfeuchtigkeit und Temperatur:

Bei einer Luftfeuchtigkeit von mehr als 40 % und/oder bei einer Temperatur von mehr als 35 Grad Celsius wird der Lüfter aktiviert und die Klappe geöffnet. Sinkt der Wert wieder unter diese Werte, wird der Lüfter wieder ausgeschaltet und die Klappe wieder geschlossen.

Bodenfeuchtigkeitsregelung:

Ist der Wert des Sensors über 2000 wird die Pumpe angeschaltet, ist der Wert wieder darunter schaltet die Pumpe sich wieder aus.

Wasserstand:

Ist der Abstand zur Wasseroberfläche höher als 20cm, wird ein Popup aktiviert welches dem Benutzter mitteilt, dass der Wasserstand zu niedrig ist.

6 Zukunftsausblick

Kurzfristig:

Die Schwellwerte für die An/Ausschaltung der Aktoren müssten in einer echten

Testumgebung überprüft werden, da ich bisher noch keine Zeit hatte eine Pflanze einzupflanzen. Diese können dann einfach im Node-RED Flow angepasst werden.

Es könnte ein einfacher an/aus Schalter draußen an die Box angebaut werden, der das System ohne große Mühe aktivieren und deaktivieren könnte.

Außerdem könnte eine kleine Status LED auch außen an die Box angebracht werden, die bei zu geringem Wasserstand Rot leuchtet oder auch andere Funktionen anzeigt.

Langfristig:

Es könnte eine Kamera verbaut werden die Fotos von den Pflanzen macht, welche dann per Telegram verschickt werden könnte.

Außerdem wäre es möglich eine Datenbank anzulegen die alle Messdaten langfristig speichert, daraus könnte man in Zukunft schließen welche Schwellwerte am besten Funktionieren.
Eine letzte Idee wäre, dass System über Solar oder eine Batterie zu betreiben, was es komplett autonom machen würde.

7 Fazit

7.1 Fachliches Fazit

Das Smart-Garden-System erfüllt erfolgreich die im Projekt definierten Ziele: Die Integration von Sensoren und Aktoren, die drahtlose Kommunikation über MQTT, sowie die Steuerung und Datenverarbeitung mit Node-RED wurden vollständig umgesetzt. Durch den Einsatz des ESP-32 als zentraler Controller und die Nutzung von Micropython konnte eine kompakte, flexible Steuerung entwickelt werden. Die Verbindung zu einem externen MQTT-Broker sowie die visuelle Darstellung und Logiksteuerung in Node-RED machen das System gut skalierbar und wartbar. Besonders positiv ist die Echtzeitverarbeitung der Sensordaten und die automatische Reaktion des Systems auf Umweltveränderungen.

Die Umverkabelung und eigenständige Programmierung zeigen, dass das Projekt nicht nur auf bestehender Hardware aufbaute, sondern inhaltlich komplett neu entwickelt wurde. Die klare Trennung von Sensorik, Logik und Aktorik macht das System nachvollziehbar und modular. Verbesserungspotenzial liegt in der langfristigen Datenspeicherung, der Optimierung der Schwellenwerte und der Erhöhung der Energieautonomie.

7.2 Persönliches Fazit

Das Projekt hat mir insgesamt sehr viel Freude bereitet – insbesondere das kreative Zusammensetzen der verschiedenen Sensoren und Aktoren sowie das Herumprobieren beim Aufbau des Systems. Das Löten der Kabel war für mich eine völlig neue Erfahrung, die ich jedoch als sehr bereichernd empfand.

Ich bin zufrieden mit dem Ergebnis und stolz, das System vollständig überarbeitet und funktionsfähig gemacht zu haben und auch, dass ich die Probleme, die in der Vorgängerversion mit Node-RED auftraten, erfolgreich lösen konnte.

Etwas schade ist, dass ich es zeitlich nicht mehr geschafft habe, die geplante Status-LED oder die Kamera zu integrieren – beides waren Features, die ich mir ursprünglich fest vorgenommen hatte. Die umfassende Neuverkabelung samt Lötarbeiten hat sich als deutlich zeitintensiver herausgestellt als gedacht.

Nichtsdestotrotz war das Projekt eine sehr spannende und lehrreiche Erfahrung für mich und ich würde das Smart-Garden-System gerne im Rahmen des IoT-Kurses weiterentwickeln.

8 Anhang

8.1 MQTT-Topics

MQTT-Topic	Beschreibung	Datentyp
DLN/test/temp	Temperatur in °C	Float
DLN/test/humi	Luftfeuchtigkeit in %	Float
DLN/test/moist	Bodenfeuchtigkeit 0-4095	Integer
DLN/test/dist	Distanz zum Wasser in cm	Float
DLN/test/luefter	Lüfter/Servomotor Steuerung	String(“on“/“off“)
DLN/test/pumpe	Pumpensteuerung	String(“on“/“off“)

8.2 Literaturverzeichnis

[1] „4-channel relay module 5V with optocoupler low-level trigger compatible with Arduino and Raspberry Pi“, AZ-Delivery. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.az-delivery.de/en/products/4-relais-modul

[2] „About : Node-RED“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://nodered.org/about/

[3] R. P. Ltd, „Buy a Raspberry Pi 4 Model B“, Raspberry Pi. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[4] „DHT 22 | Joy-IT“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://joy-it.net/de/products/SEN-DHT22

[5] „dln-smartgarden/CodeForESP-32 at main · Elociraptor/dln-smartgarden“, GitHub. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://github.com/Elociraptor/dln-smartgarden/tree/main/CodeForESP-32

[6] „dln-smartgarden/NodeRed-flow at main · Elociraptor/dln-smartgarden“, GitHub. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://github.com/Elociraptor/dln-

smartgarden/tree/main/NodeRed-flow
[7] „ESP32 Grundlagen (Basiswissen) - digitalewelt“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://digitalewelt.at/esp32-grundlagen/
[8] „ESP32 MicroPython Soil Moisture Sensor | ESP32 MicroPython Tutorial“, Tutorials for Newbies. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://newbiely.com/tutorials/esp32-micropython/esp32-micropython-soil-moisture-sensor
[9] „https://raw.githubusercontent.com/RuiSantosdotme/ESP-MicroPython/master/code/MQTT/umqttsimple.py“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://raw.githubusercontent.com/RuiSantosdotme/ESP-MicroPython/master/code/MQTT/umqttsimple.py
[10] S. Santos, „MicroPython: HC-SR04 Ultrasonic Sensor ESP32/ESP8266 | Random Nerd Tutorials“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/
[11] „Quick reference for the ESP32 — MicroPython latest documentation“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://docs.micropython.org/en/latest/esp32/quickref.html
[12] „Quick reference for the ESP32 — MicroPython latest documentation“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://docs.micropython.org/en/latest/esp32/quickref.html#dht-driver
[13] „Quick reference for the ESP32 — MicroPython latest documentation“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://docs.micropython.org/en/latest/esp32/quickref.html#pwm-pulse-width-modulation
[14] „Quick reference for the ESP32 — MicroPython latest documentation“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://docs.micropython.org/en/latest/esp32/quickref.html#pins-and-gpio
[15] „Raspberry Pi 4“, Elektor. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.elektor.de/collections/raspberry-pi-4
[16] „Raspberry Pi 4 Modell B“. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.elektronik-kompendium.de/sites/raspberry-pi/2407251.htm
[17] „SG90 9g Micro Servomotor“, Roboter-Bausatz.de. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.roboter-bausatz.de/p/sg90-9g-micro-servomotor
[18] M. Krohn, „Was ist MQTT? Erklärung mit industriellem Fokus“, OPC Router - The Communication Middleware. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.opc-router.de/was-ist-mqtt/
[19] „Was ist Node-RED? | Softwareentwicklung | PI-Lexikon“, pi-informatik. Zugegriffen: 6. Juli 2025. [Online]. Verfügbar unter: https://www.pi-informatik.berlin/pi-lexikon/softwareentwicklung/was-ist-node-red/