Lichtstimmung im Garten

Flexibel erweiterbare Gartenbeleuchtung

Bunte und schön angelegte Gärten und Terrassen liegen nicht erst seit der Corona-Pandemie im Trend, und Gartencenter bzw. Gartenplaner werden regelrecht gestürmt. Die Gartenbeleuchtung hinkt diesem Trend ein wenig hinterher. Hier findet man häufig preisgünstige Solarleuchten, die oft ein Jahr nicht überstehen und dann wegen der defekten Akkus im Sondermüll landen. Oder man entdeckt Gartenleuchten, die zwar über eine Bluetooth-App steuerbar, aber nicht in Hausautomatisierungslösungen wie Homematic IP integrierbar sind. In diesem Leserwettbewerb-Beitrag von Dr. Peter Tschulik zeigen wir Ihnen, wie Sie eine individuelle Beleuchtung für Ihren Garten im Eigenbau realisieren und komfortabel über mediolas AIO Creator NEO für Homematic IP, Smartphone oder PC steuern und automatisieren können.

Peter Tschulik
hat für seinen Beitrag zum Leserwettbewerb einen Gutscheincode* über 200,– Euro erhalten!

*Siehe Seite 112

Den Garten erleuchten

Ziel meines Projekts war es, eine flexible und erweiterbare Beleuchtungslösung zu schaffen. Dabei sollten verschiedene Gartenleuchten aus dem Baumarkt mittels leicht erhältlicher Neopixel-WS2812-Ringe so umgebaut werden, dass sie auch verteilt auf mehrere Terrassen oder mehrere weiter entfernte Plätze im Garten synchron in einer Farbe oder in Farbprogrammen den Garten erleuchten. Das Ganze gesteuert über eine Webschnittstelle oder Homematic IP über AIO Creator NEO und NEO Server. Die Anforderungsliste für mein Projekt war entsprechend lang:

- Anbindung an das Heimnetzwerk über WLAN oder Ethernet (LAN)
- Einfacher Aufbau auf mit in Eagle entwickelten PCBs
- Verteilte Struktur über einen Master basierend auf einem ESP32 und bis zu drei Slaves basierend auf einem Arduino Leonardo Micro, die über eine DMX-Schnittstelle miteinander vernetzt sind
- Autonomer Betrieb über eine Zeitsteuerung mit integriertem NTP-Server und automatischer Anpassung der monatlichen Einschaltzeiten an den Sonnenuntergang
- Parametersteuerung über einfache HTPP-Schnittstelle
- Komfortables Webinterface zur Steuerung und Konfiguration aller Parameter mit flexibler Anpassung der Webseiten an PC oder Mobiltelefone
- Vollständige Integration in Homematic IP über den AIO Creator NEO und NEO Server
- Eingebauter Bootloader für Upgrade über WLAN/Ethernet für den Master
- Umfangreiche Beleuchtungsmöglichkeiten: zehn fixe Farben mit neun Dimmstufen, beliebige Farbenauswahl über ein Farbrad, 13 animierte Programme mit wählbarer Helligkeit und Geschwindigkeit und fünf frei belegbare Speicherpositionen
- Einzelne Slaves können aktiviert und deaktiviert werden
- Erstkonfiguration über INI-Text-File oder Konsoleneingabe

In Bild 1 ist der generelle Aufbau der Gartenbeleuchtung in der WLAN-Version mit den dazugehörigen Blockschaltbildern dargestellt. Die LAN-Version unterscheidet sich nur in der Anbindung des ESP32.

Der ESP32 im Master stellt die Bedienschnittstelle über ein LAN oder WLAN zur Verfügung, übernimmt die komplexe Berechnung der Farbverläufe und steuert die Slaves über eine DMX-Schnittstelle an. Diese recht alte, aber bewährte Schnittstelle kommt mit drei Leitungen (GND, DMX- und DMX+) aus und ist noch heute in der Bühnen- und Lichttechnik weit verbreitet.

Durch den Einsatz einer symmetrischen Verbindung und geringen Datenraten können mit guten Kabeln bis zu 100 m überbrückt werden. In meiner Installation kommt ein Outdoor-geeignetes Ether-

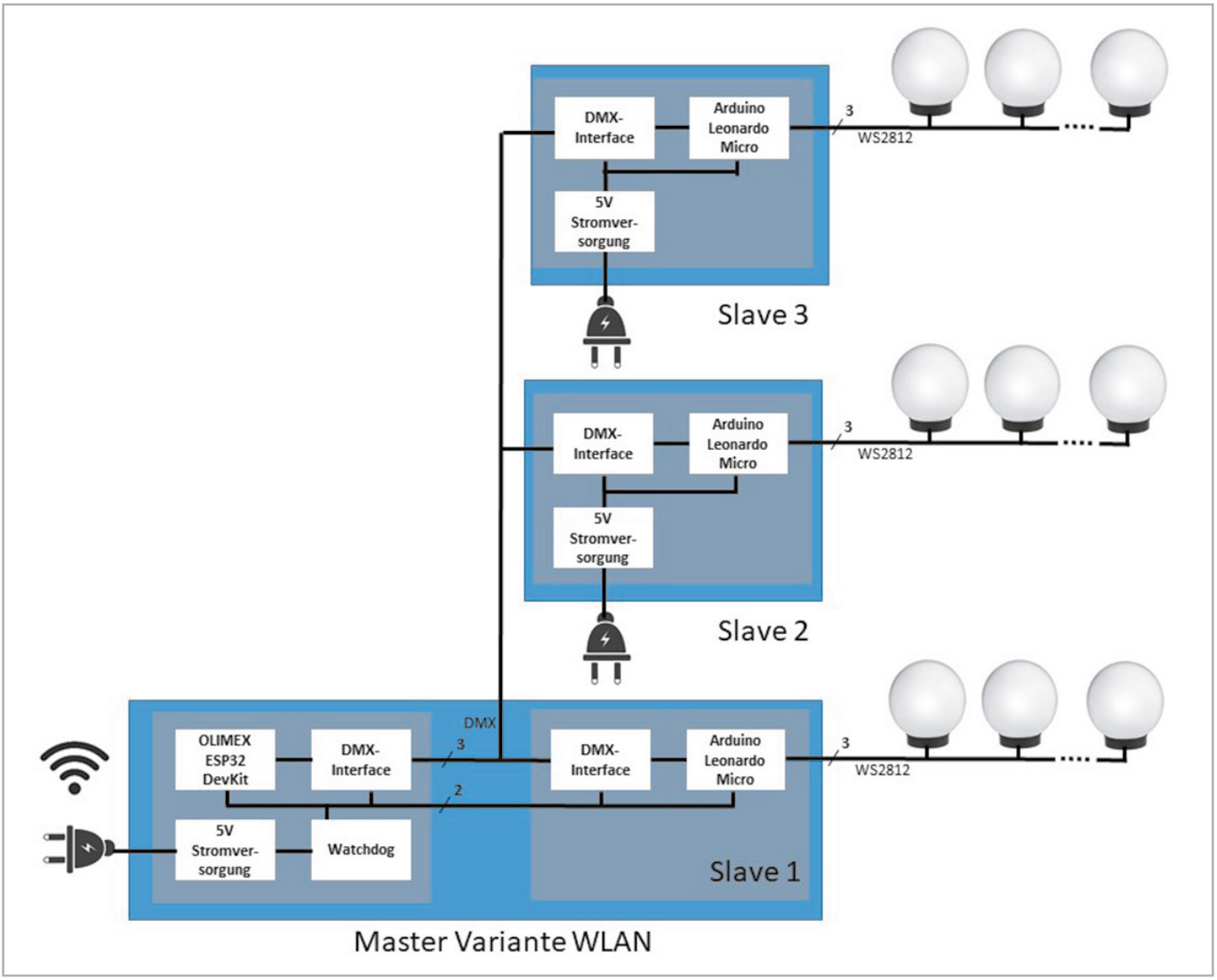

Bild 1: WLAN-Version

net-Kabel zur Anwendung. Weitere Informationen zum DMX-Protokoll findet man unter [1].

In jedem Slave wird das DMX-Signal in ein Signal zur Ansteuerung der WS2812-Neopixel umgewandelt. Jeder Slave verfügt über eine DMX-Adresse, die über Jumper einstellbar ist, wobei pro Slave jeweils drei DMX-Adressen verwendet werden. Der Master kann natürlich auch so ausgeführt werden, dass Slave 1 ebenfalls abgesetzt wie Slave 2 und 3 betrieben wird.

Bild 2 zeigt den Schaltplan für den Master der WLAN-Variante aus Bild 1. Das Herzstück der Schaltung ist ein ESP32-Board der Firma Olimex (s. Material [2]). Zum Einsatz kommt ein ESP32-DevKit-Lipo-EA-Board mit einem U.FL-Verbinder und einer externen WLAN-Antenne, um den Empfang in nicht so gut versorgten Gartenbereichen sicherzustellen.

Zwei weitere fertig aufgebaute Boards kommen zum Einsatz – ein RS485-Modul (s. Material [2]) und ein MOSFET-Modul zum Schalten der Versorgungsspannung (s. Material [2]). Als Watchdog wird ein Atmel ATtiny13a verwendet.

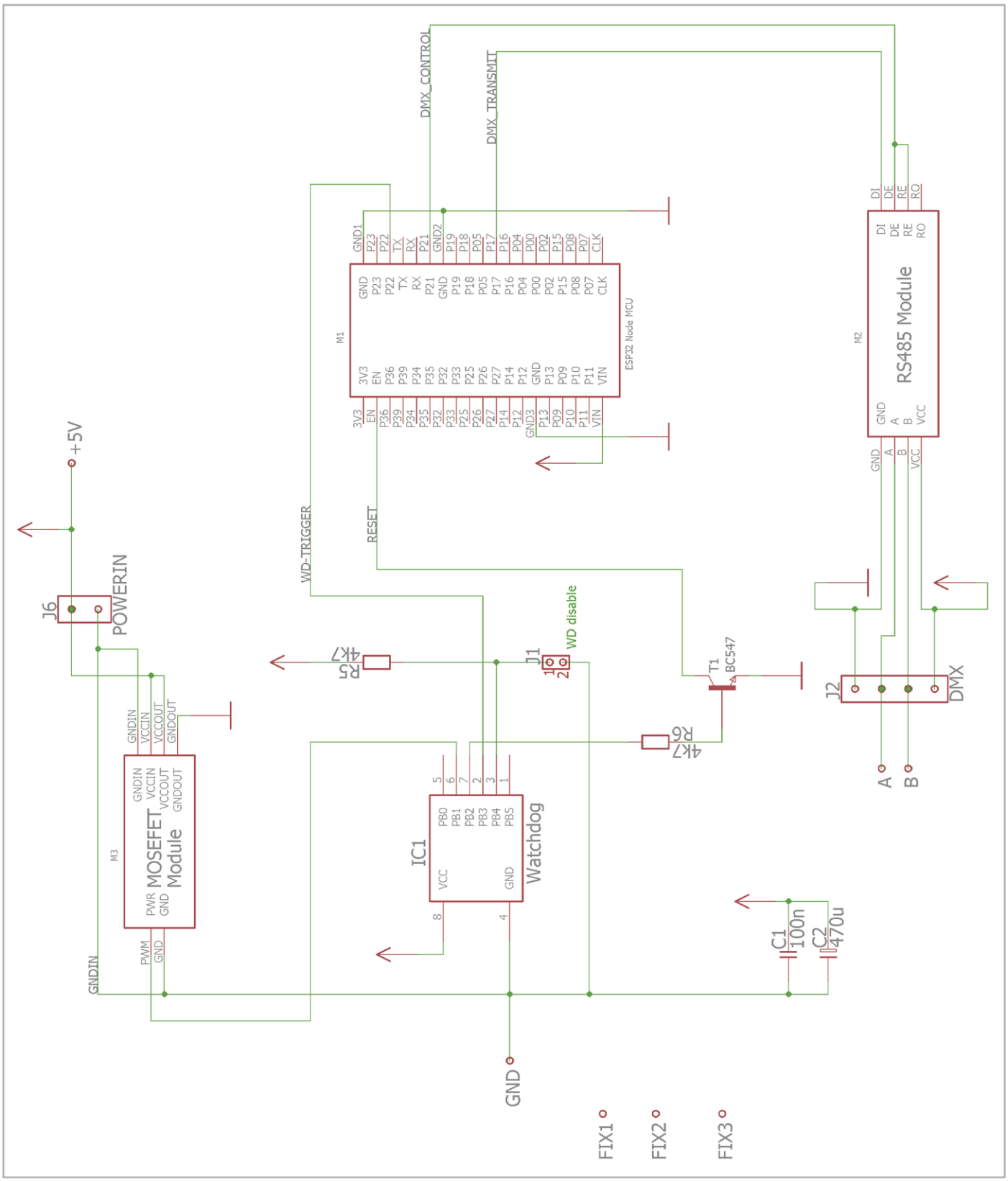

Bild 2: Schaltplan Master WLAN-Variante

Die Beschreibung und Programmierung des Mikrocontrollers sind Thema im Kapitel Nachbau und Inbetriebnahme. Über den Jumper J1 kann der Watchdog deaktiviert werden.

Im Gegensatz zur WLAN-Variante kommt bei der LAN-Variante ein ESP32-Gateway-Board zum Einsatz, das einen Ethernet-Anschluss anbietet. Die Beschreibung der LAN-Variante findet man im Zusatzdokument „LAN-Version“ [2].

Bild 3 zeigt den Schaltplan für den Slave. Beim Slave kommen lediglich zwei fertig aufgebaute Boards zum Einsatz, ein RS485-Modul und ein Arduino Leonardo Micro (s. Material [2]). Das RS485-Modul wandelt die symmetrischen DMX-Signale in ein unsymmetrisches Signal für den Mikrocontroller des Arduino Leonardo Micro um. Dieser konvertiert die empfangenen DMX-Signale in WS2812-konforme Signale. Über die Jumper J3, J4 und J5 kann die DMX-Adresse – wie im folgenden Kapitel beschrieben –eingestellt werden.

Nachbau und Inbetriebnahme

Für den einfachen Aufbau wurden mit dem Programm Eagle drei PCBs (Master WLAN, Master LAN und Slave) erstellt, die unter [2] im Downloadbereich des Beitrags verfügbar sind und beliebig modifiziert werden können. Neben den Eagle Files sind auch der Schaltplan, der Bestückungsplan sowie das Layout als JPEG-Bilder verfügbar. Die eigens erstellten Symbole kann man ebenfalls im Downloadbereich des Beitrags herunterladen. Meine Platinen habe ich auf meiner Fräse hergestellt, als Alternative können diese auch in Eigenregie geätzt bzw. bei einem PCB-Hersteller in Auftrag gegeben werden.

Slave

Beginnen wir mit der Slave-Platine: Nach der Bestückung der beiden Drahtbrücken wird der Widerstand sowie der Entstörkondensator aufgelötet. Danach wird der Arduino Leonardo Micro über Stiftleisten in der korrekten Orientierung aufgelötet. Vom RS485-Modul wird der grüne Schraubanschluss wegen der Einbauhöhe ausgelötet und dann das Modul über Stiftleisten in der korrekten Orientierung aufgelötet. Mit der Bestückung der Schraubanschlüsse ist der Aufbau abgeschlossen. Bild 4 zeigt die fertig bestückte Platine.

Schließlich muss noch die DMX-Slave-Adresse mittels Lötbrücken oder Jumper wie folgt festgelegt werden:

	DMX-Adresse	J3	J4	J5
Slave 1	1	geschlossen	geschlossen	geschlossen
Slave 2	4	offen	geschlossen	geschlossen
Slave 3	7	geschlossen	offen	geschlossen

Es folgt nun das Flashen der Software. Diese wurde in der Arduino IDE (Version 1.8.12) erstellt. Auf eine genaue Beschreibung der ausführlich dokumentierten, relativ einfachen Software wird an dieser Stelle verzichtet.

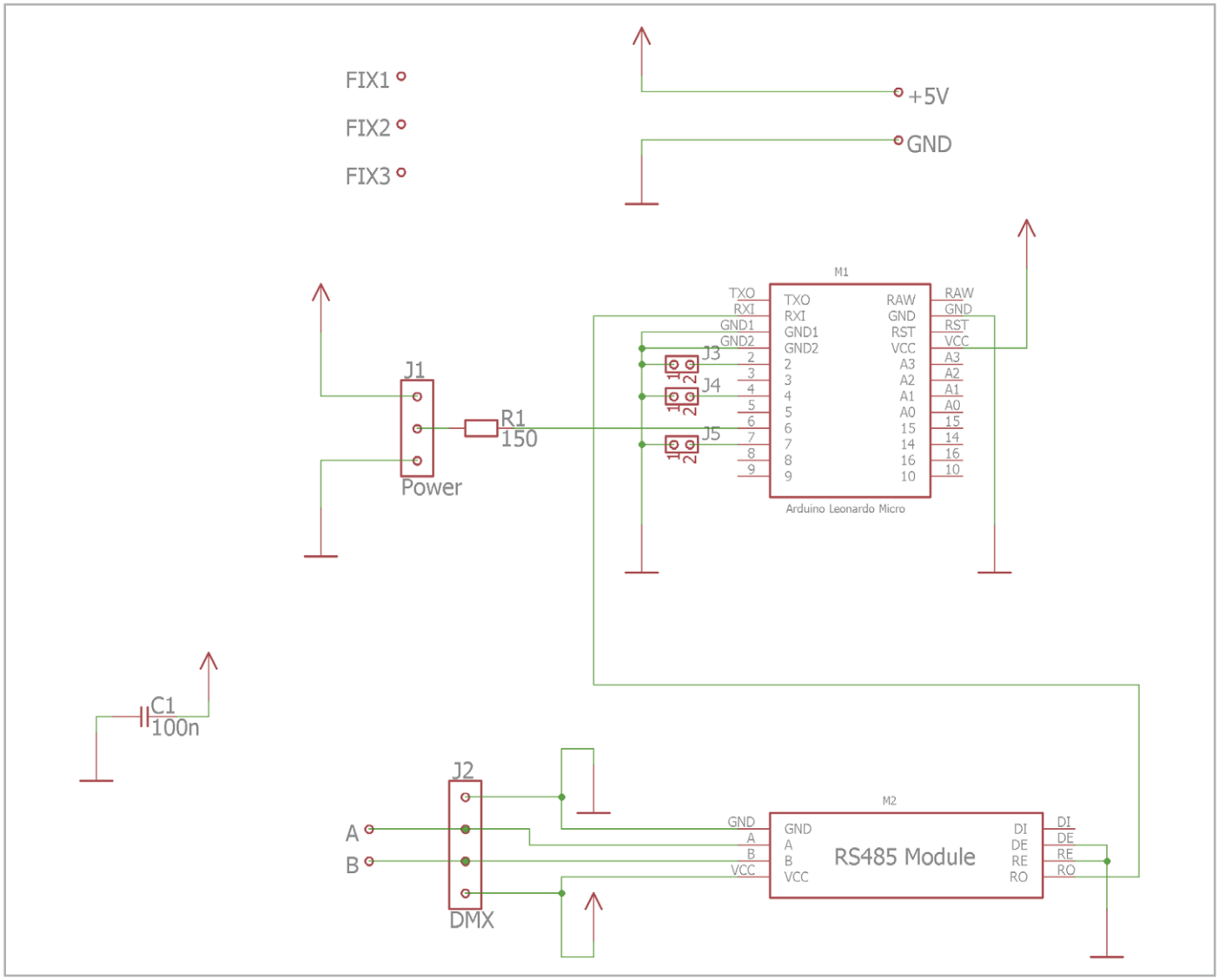

Bild 3: Schaltplan Slave

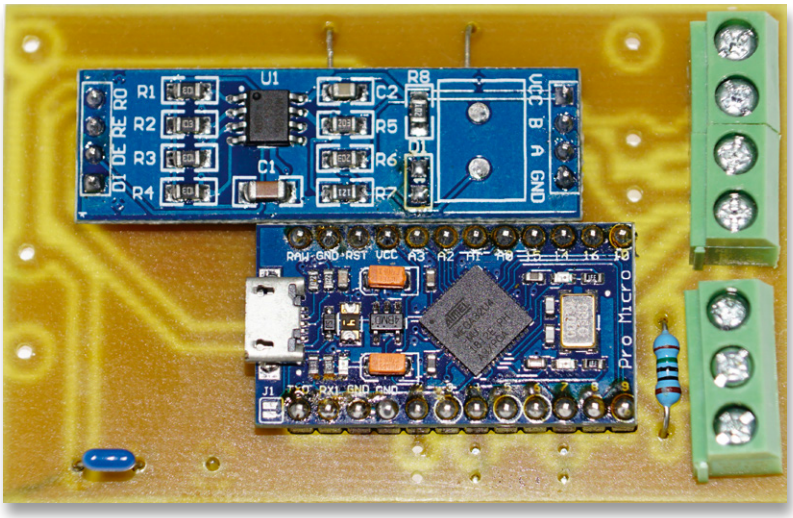

Bild 4: Bestückte Slave-Platine

Folgende Vorbereitungsarbeiten sind vor dem Übertragen der Software notwendig:

Die beiden Libraries „Fast LED" und „DMX-Serial" müssen installiert werden. Die Links zu den Libraries finden sich in Zeile 19-20 von „Gartenbeleuchtung_Slave.ino".

Der Sourcecode ist unter [2] im Directory

Arduino Source\ Gartenbeleuchtung_Slave,

die Libraries im Directory

Arduino Libraries\ Gartenbeleuchtung_Slave zu finden.

Als Board wird in der Arduino IDE das Arduino Leonardo ausgewählt. Im Programm muss lediglich in Zeile 35 die Konstante NUM_NEO_PIXELS mit der tatsächlich angeschlossenen Anzahl an WS2812-Neopixel angepasst werden. Im mitgelieferten Sourcecode beträgt der Wert 48, da drei Lampen mit jeweils einem LED-Ring (s. Material [2]) mit 16 LEDs zum Einsatz kommen.

Der Arduino Leonardo Micro wird über ein passendes USB-Kabel an den PC angeschlossen und die Firmware übertragen. War das erfolgreich, erfolgt der erste Funktionstest: Die Schaltung wird mit 5 V über ein Labornetzteil versorgt, und ein LED-Ring wird, wie in Bild 5 gezeigt, angeschlossen.

Da keine Verbindung zum Master hergestellt werden kann, sollten alle LEDs rasch rot blinken. Ist dies der Fall, ist der Funktionstest erfolgreich abgeschlossen, und es geht an den Aufbau des Masters.

Hinweis: Im Folgenden wird beispielhaft die Beschreibung für die Variante mit dem Anschluss an das WLAN gezeigt. Die Dokumentation für die LAN-Variante befindet sich im Zusatzdokument „LAN-Version" [2].

Variante Master WLAN

Zuerst erfolgt die Bestückung von drei Drahtbrücken, der Widerstände, der Kondensatoren und des Transistors. Bei der Bestückung des Kondensators C2 und des Transistors T1 ist auf die korrekte Polarität zu achten. Schließlich wird Jumper J1 bestückt. Danach erfolgt die Programmierung und Bestückung des Watchdog IC1. Für die Watchdog-Schaltung wird kein eigener Reset-Schaltkreis verwendet, sondern eine flexiblere Lösung auf Basis eines günstigen Atmel ATtiny13a.

Den in der Arduino IDE geschriebenen Sourcecode findet man unter [2] im Directory

Arduino Source\ Atiny_Watchdog.

Der Watchdog spielt auch eine entscheidende Rolle beim Aktivieren des Bootloaders für ein Remote-Software-Update.

Nach einem Power-On schaltet die Watchdog-Schaltung die Stromversorgung über das MOSFET-Modul ein und setzt den ESP32 zurück. Danach wird auf die erste positive Flanke am Watchdog-Triggereingang gewartet, die den Watchdog scharfschaltet. Nun muss jeweils innerhalb von 8 Sekunden eine negative Flanke erkannt werden, damit der Watchdog nicht abläuft. Bleibt das Triggern des Watchdogs wegen eines Software-Fehlers aus, so wird eine Resetprozedur ausgelöst. Wie man mithilfe eines Arduino UNOs und der Arduino IDE den Atmel ATtiny13a programmiert, ist in [3] beschrieben.

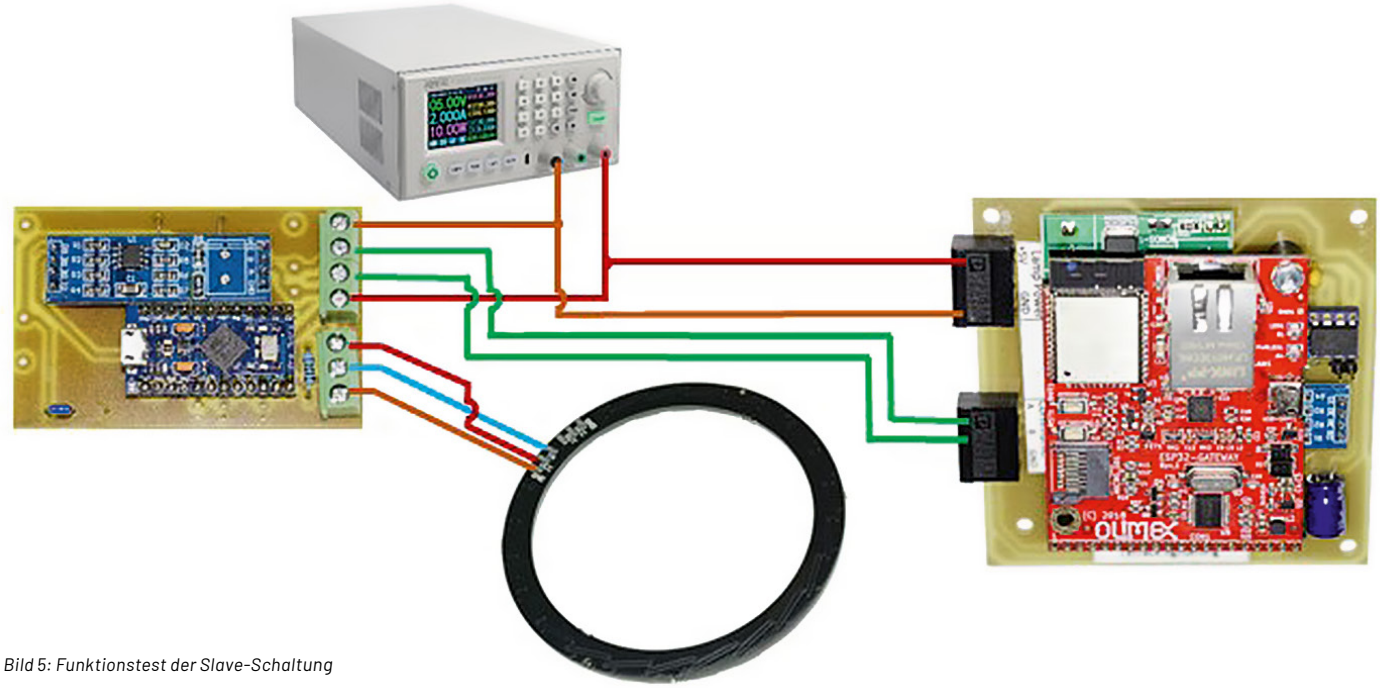

Bild 5: Funktionstest der Slave-Schaltung

Nach der Programmierung wird für den Watchdog IC1 ein 8-poliger IC-Sockel bestückt und der Watchdog IC in der korrekten Orientierung eingesetzt.

Bevor das ESP32-DevKit-Lipo-EA Modul bestückt wird, sollte zuerst die Software erstmalig aufgespielt werden. Diese wurde in der Arduino IDE (Version 1.8.12) erstellt und ist in folgende Blöcke geteilt:

- „Gartenbeleuchtung_Master_LAN.ino" enthält neben den Definitionen die Setup- und Main-Routine.
- „Basis.ino" enthält Basisroutinen wie beispielsweise das Lesen des Init-Files, die Debug-Routinen, die Watchdog-Routinen, den Bootloader und die DMX-Basisroutine.
- „Lan.ino" enthält die Ethernet-Routinen sowie die NTP-Routinen.
- „Led.ino" enthält die Routinen zur Anzeige von statischen Farben sowie Programmen und das Ein- und Auslesen der fünf Speicher.
- „Web.ino" schließlich beinhaltet die Behandlung und den Aufruf der einzelnen Webseiten, die über die Bibliothek ESP32 Async Web Server erfolgt.

In Zeile 271 von „Gartenbeleuchtung_Master_WLAN.ino" kann der Befehl "#define _DEBUG_" durch Entfernung der beiden // am Beginn der Zeile aktiviert werden. Dann werden viele zusätzliche Informationen über den Serial Monitor der Arduino IDE mit 115.200 Baud ausgegeben. Das ist sehr hilfreich, wenn etwas nicht funktioniert oder der Code erweitert oder geändert werden soll. Ist das Debugging aktiviert, besteht in der Startsequenz des Programms auch die Möglichkeit, über den Seriellen Monitor der Arduino-Programmumgebung die Parameter zeilenweise zu ändern.

In Zeile 332 sollte in LOGIN = "admin:password/" das Wort *admin* durch ein selbst definiertes Login und das Wort *password* durch ein selbst gewähltes Passwort ersetzt werden. Dies hat eine Auswirkung auf die Ansteuerung über den AOI Creator NEO über HTTP-Befehle der Form

http://<IP-Adresse>/?login=<Login>:<Passwort>/c=148

wie sie später im Text beschrieben wird.

Ein Eintrag in Zeile 328 in der Form LOGIN = "garten:gt1501/" würde einen HTTP-Befehl in der Form

http://<IP-Adresse>/?login=garten:gt1501 /c=148

ergeben, wobei die IP-Adresse noch entsprechend angepasst werden muss. Bevor jedoch der Code auf den ESP32 übertragen wird, müssen noch die Daten im Unterverzeichnis „data" angepasst und übertragen werden. Die Datei „Init.txt" enthält die folgenden Konfigurationsparameter:

- Zeile 01: SSID des WLAN-Netzes: Default=SSID
- Zeile 02: Passwort des WLAN-Netzes: Default=1234
- Zeile 03: Hostname: Default=Gartenleuchte
- Zeile 04: fixe IP-Adresse: Default=192.168.123.225
- Zeile 05: Subnet-Maske: Default=255.255.255.0
- Zeile 06: Gateway IP-Adresse: Default=192.168.123.1
- Zeile 07: DNS-Server 1: Default=195.58.161.123
- Zeile 08: DNS-Server 2: Default=212.186.211.21
- Zeile 09: NTP-Server 1: Default=time.nist.gov
- Zeile 10: NTP-Server 2: Default=pool.ntp.org
- Zeile 11: NTP-Server 3: Default=europe.pool.ntp.org

Eine genauere Beschreibung der einzelnen Parameter befindet sich im Zusatzdokument „Bedienung über den Webbrowser" [2].

Die einzelnen Zeilen sind mit einem CRLF (Enter) geteilt, nach der letzten Zeile darf kein Zeilenumbruch mehr stehen!

In dem Unterverzeichnis „data" sind weiterhin die Styledatei „Style.css" und die zehn Webseiten abgelegt. Bei Bedarf können auch diese angepasst werden. Zudem müssen noch die Libraries „ESP32 Ping", „Time library", „ESP32 Async Web Server", „Async TCP" und „ESP32 DMX Library" installiert werden. Die Links zu den Libraries finden sich in Zeile 20–24 von „Gartenbeleuchtung_Master_WLAN.ino".

Der Sourcecode ist unter [2] im Directory

Arduino Source\ Gartenbeleuchtung_ ↩
Master_WLAN,

die Libraries im Directory

Arduino Libraries\ Gartenbeleuchtung_ ↩
Master_WLAN

zu finden.

Nun geht es darum, die Daten und das Programm zu übertragen. Das ESP32-DevKit-Lipo-EA-Modul wird mit dem PC über ein passendes USB-Kabel verbunden. In der Arduino IDE muss sowohl die ESP32-Erweiterung (ich verwende aktuell ESP32 arduino extension 1.0.4) als auch der ESP32 Sketch Data Uploader (Installationsanleitung siehe [4] oder Informationen in Zeile 26–28 von „Gartenbeleuchtung_Master_WLAN.ino") installiert sein. Als Board wird das „OLIMEX ESP32 DevKit-Lipo" ausgewählt, danach die korrekte Schnittstelle.

Zuerst müssen die Daten, die sich im Unterverzeichnis „data" befinden (Init-file, Webseiten, Stylesheet) unter

Tools → ESP32 Sketch Data Upload

in den Flash-Speicher des ESP32 übertragen werden. Danach wird der Sketch mittels

Sketch → Upload

kompiliert und übertragen. Hat das funktioniert, wird das ESP32-DevKit-Lipo-EA-Modul aufgelötet. Bild 6 zeigt die teilbestückte Platine.

Um ein möglich kompaktes Design zu erreichen, werden das RS485- und das MOSFET-Modul auf der Platinen-Rückseite bestückt. Beide Module werden von den Schraubenanschlüssen befreit und über Stiftleisten auf der Platinen-Rückseite verlötet. Die fertig aufgebaute Platine ist in Bild 7 dargestellt.

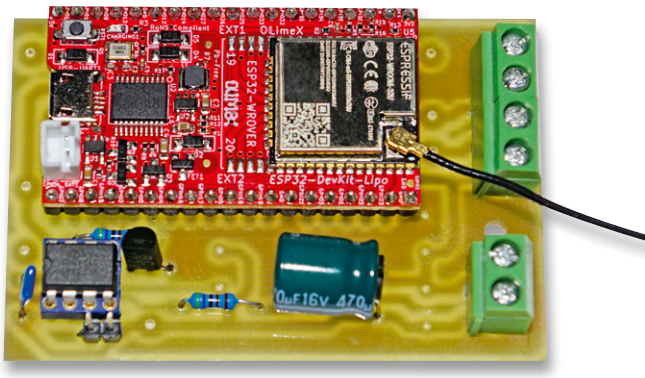

Bild 6: Teilbestückung Master WLAN-Platine Oberseite

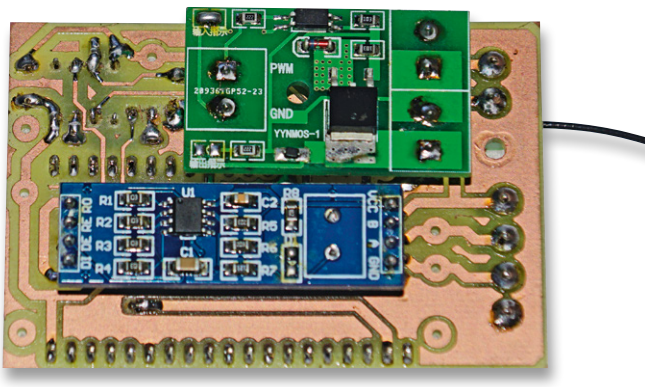

Bild 7: Fertig bestückte Master LAN-Platine mit ESP32-Modul

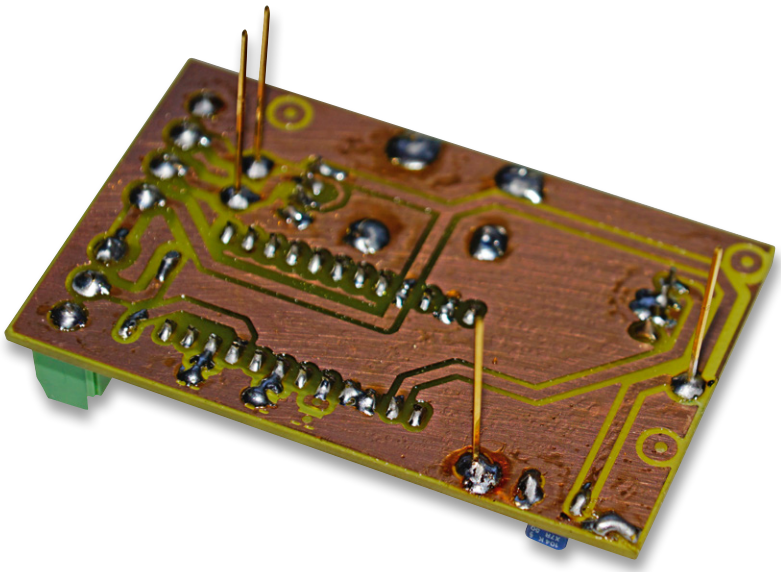

Bild 8: Bestückung der vier Stiftleisten auf der Unterseite der Slave-Platine

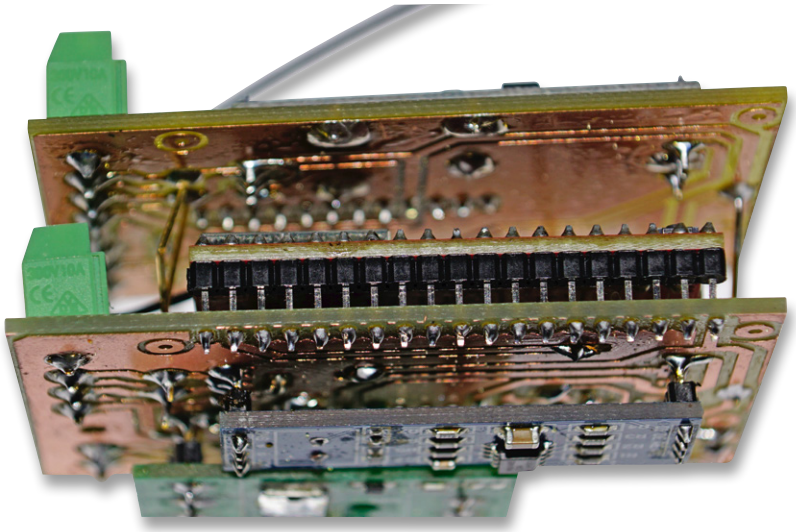

Bild 9: Fertiges Sandwich aus Master und Slave

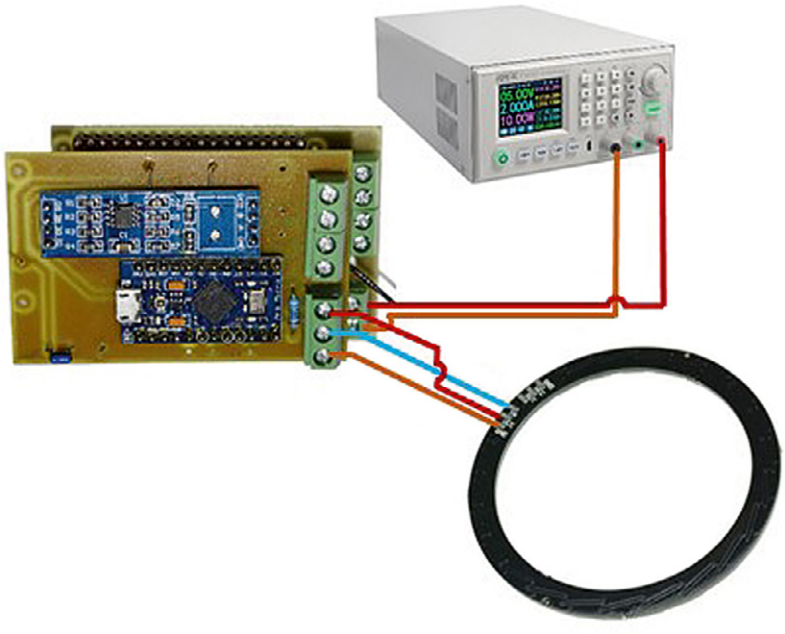

Bild 10: Funktionstest Master WLAN-Platine mit Slave

Für einen besonders kompakten Aufbau kann der Slave huckepack auf die Masterplatine aufgelötet werden. Dazu werden auf der Slave-Unterseite vier längere Stiftleisten angelötet (Bild 8).

Danach wird die Slave-Platine deckungsgleich auf die Masterplatine aufgesetzt, und beide Platinen werden miteinander verbunden (Bild 9).

Nun erfolgt der erste Funktionstest. Master und Slave werden, wie in Bild 10 gezeigt, verschaltet und über ein Labornetzteil mit 5 V versorgt.

Achtung: Für den Funktionstest muss der Slave auf Slave 1 (DMX-Adresse 1, alle Jumper geschlossen) eingestellt werden! Fehler werden immer mit einem Blinken von 0,5 Sekunden angezeigt. Zuerst sollte der LED-Ring einige Sekunden rot blinken und danach 2 Sekunden gelb aufleuchten, was eine erfolgreiche Initialisierung des SPIFFS anzeigt. Danach wird eine Verbindung mit dem WLAN versucht. Ein erfolgreicher Versuch wird mit einem violetten Aufleuchten für 2 Sekunden angezeigt, bei einem Fehler leuchtet der LED-Ring 2 Sekunden rot, bevor der Watchdog die Schaltung zurücksetzt.

Danach wird versucht, die Zeit über einen NTP-Server abzufragen. Ein erfolgreicher Versuch wird mit einem cyanen Aufleuchten für 2 Sekunden angezeigt, bei einem Fehler leuchtet der LED-Ring 2 Sekunden rot, bevor der Watchdog die Schaltung zurücksetzt. Schließlich wird der Server gestartet und die Konfiguration geladen.

Ein erfolgreicher Versuch wird mit einem grünen Aufleuchten für 2 Sekunden angezeigt, bei einem Fehler leuchtet der LED-Ring 2 Sekunden rot, bevor der Watchdog die Schaltung zurücksetzt. So kann ein erfolgreicher Hochlauf durch Beobachtung des LED-Rings mitverfolgt werden. Bei einem Fehler sollten die Debug-Meldungen wie weiter oben beschrieben aktiviert und im Serial Monitor der Arduino IDE analysiert werden.

Umrüstung der Gartenlampen

Ich zeige hier beispielhaft die Umrüstung einer Kugellampe (Bild 11). Der Umbau einer Solarleuchte und Lounge-Lampe wird im Zusatzdokument „Umrüstung weiterer Gartenlampen“ [2] beschrieben. Im Beispiel werden drei Kugellampen – wie in Bild 12 dargestellt – umgerüstet:

Als Verbindungskabel wurde ein 3-poliges, flexibles Stromkabel mit 1 mm² Durchmesser, wie es in jedem Baumarkt erhältlich ist, benutzt. Die Entfernung zwischen den einzelnen Lampen sollte ein paar Meter nicht überschreiten. Zur Stabilisierung der Stromspitzen kommt in jede Lampe ein Elektrolytkondensator mit 470–1000 µF.

Zur Verwendung kommen RGB-Ringe mit 16 LEDs (s. Material [2]), die die Kugellampe schön homogen ausleuchten. Der Einbau des LED-Rings ist in den Bildern 13 und 14 dargestellt.

In den schwarzen Sockel werden zwei Löcher links und rechts gebohrt, um die Kabel in die Lampe zu führen. Zwei weitere Löcher links und rechts werden für Kabelbinder genutzt, die als Zugentlastung

dienen. Vier Gummipuffer stellen den Abstand sicher, sodass die Lampe trotz der Kabel gerade steht.

Der Elektrolytkondensator wird in den Sockel mittels Heißkleber geklebt und die Kabel an diesen angelötet. An einen freien Anschluss am LED-Ring wird ein dickerer Draht angelötet und um 90 Grad gebogen. Dieser wird dann am Sockel mit ein paar Lagen Klebeband fixiert, sodass dieser quasi über dem Sockel schwebt. Da keine mechanische Beanspruchung vorliegt, reicht eine solche Befestigung völlig aus.

Ist der Umbau der Lampen erfolgt, wird die Elektronik gemeinsam mit einem passenden Netzteil in ein wetterfestes Gehäuse eingebaut. Alle Stecker und Durchführungen müssen ebenfalls wetterfest ausgeführt werden, wenn sich die Elektronik im Außenbereich befindet.

Die Stromversorgung muss wie folgt dimensioniert werden:

Maximaler Strom = (Anzahl der LEDs) * 60 mA + 300 mA (für ESP32)

Addiert man zum maximalen Strom 10–20 % als Reserve, so gelangt man zu der Stromstärke, die das Netzteil dauerhaft liefern können muss. In den oben gezeigten beiden Beispielen ist der Wert 48 (3 x 16) für die Anzahl der LEDs im Slave-Programm einzustellen.

Bild 11: Kugellampe

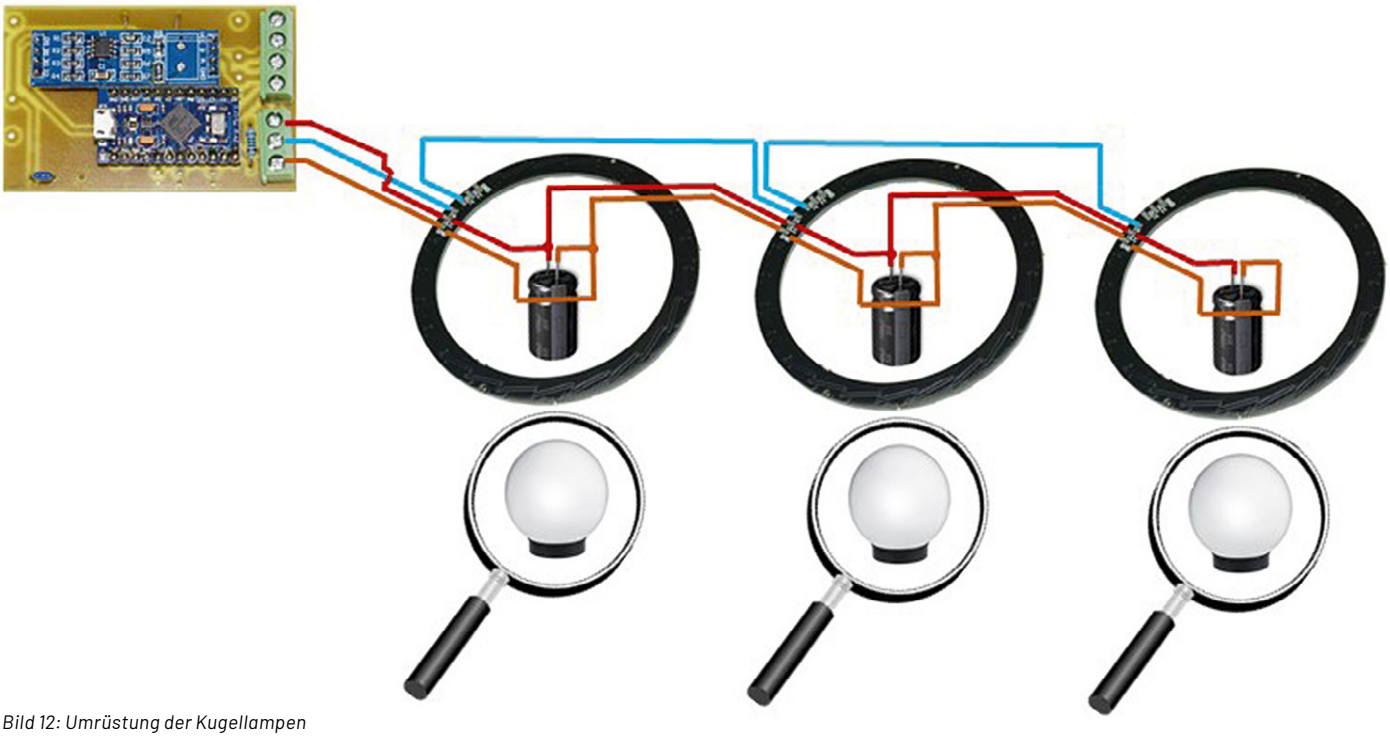

Bild 12: Umrüstung der Kugellampen

Bild 13: Anschlussleitungen und Zugentlastung

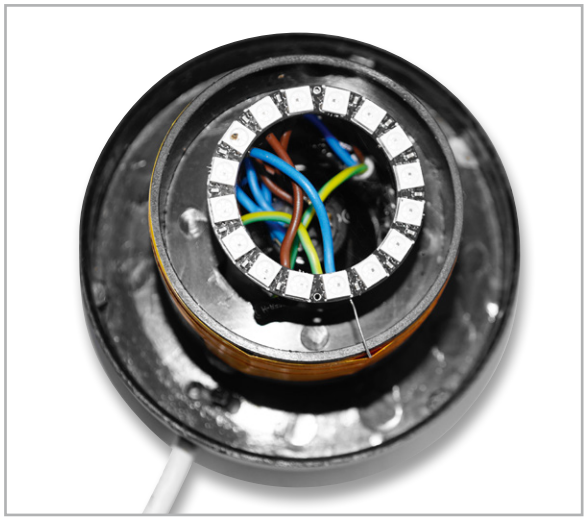

Bild 14: Montage des LED-Rings

Bedienung über den Webbrowser

Die Anleitung zur Bedienung über den Webbrowser finden Sie unter [2] im Dokument „Bedienung über den Webbrowser“.

Bild 15: Bedienseite Gartenbeleuchtung über AIO Creator NEO

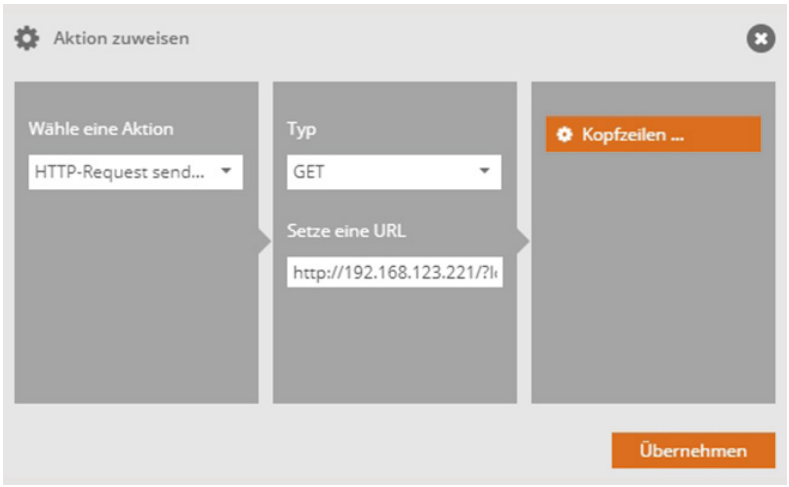

Bild 16: Aktion zuweisen für einen Button im AIO Creator NEO

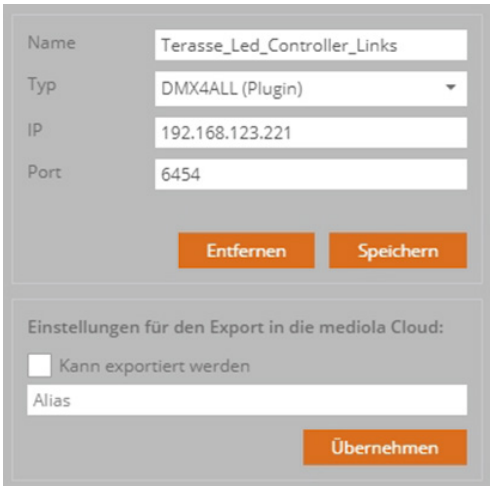

Bild 17: Neues Gerät anlegen im AIO Creator NEO

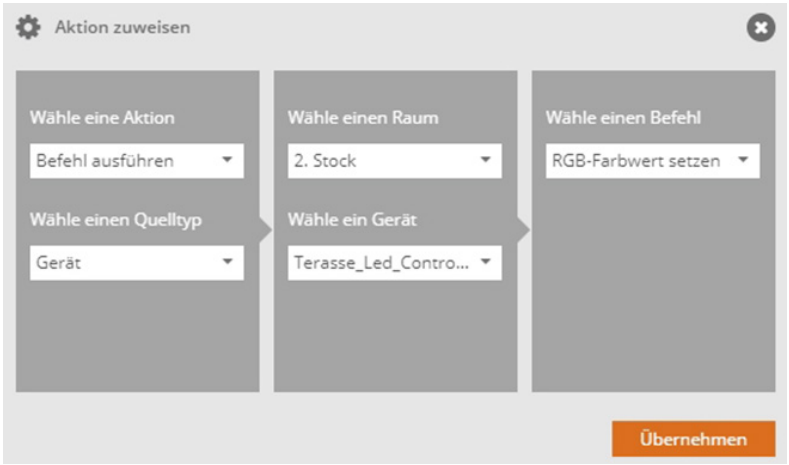

Bild 18: Aktion zuweisen für das Farbrad im AIO Creator NEO

Einbindung in Homematic IP über den AIO Creator NEO

Für die Steuerung der Gartenbeleuchtung werden zwei Seiten im AIO Creator NEO angelegt. Über die erste Seite erfolgt die Grundbedienung aller Funktionen (Bild 15).

Am Beispiel der Taste „Warmweiss“ wird die Funktion erklärt. Nachdem die Taste mit der Funktion „Button“ angelegt wurde, wird unter den Eigenschaften am rechten Rand der Button „Aktion zuweisen“ angeklickt. Es erscheint ein Pop-up-Fenster „Aktion zuweisen“. Unter „Wähle eine Aktion“ wählt man den Eintrag „HTTP-Request senden“, der Typ „GET“ bleibt wie voreingestellt, und unter „Setze eine URL“ wird Folgendes eingetragen:

Bild 19: Konfigurationsseite Gartenbeleuchtung über AIO Creator NEO

http://<IP-Adresse>/?login=<Login>:<Passwort>/c=001. Angenommen, die IP-Adresse der Gartensteuerung lautet 192.168.123.221 und das Login und Passwort wurden im Programm – wie zu Beginn des Artikels beschrieben – auf *LOGIN = "admin:password/"* gesetzt, dann lautet der einzutragende Befehl

http://192.168.123.221/?login=admin:password/c=001,

wie in Bild 16 gezeigt.

Mit „Übernehmen" wird die Zuteilung des HTTP-Befehls zum Button gespeichert und kann auch gleich mittels der Vorschau-Funktion des AIO Creator NEO getestet werden. Für alle Tasten können die entsprechenden Einträge unter [2] im Directory *Mediola\Steuerung.txt* gefunden werden.

Nun fehlt noch das Farbwählrad. Dieses wird mit der Funktion „Farbwähler" eingefügt. Damit die Funktion mit der Gartensteuerung funktioniert, muss das Plugin DMX4All [5] von Mediola erworben und installiert werden. Danach muss ein neues Gerät angelegt werden. Dazu ruft man den Gerätemanager auf und wählt die Funktion „Gerät anlegen" aus. Man vergibt einen beliebigen Namen, wählt als Typ „DMX4ALL (Plugin)" aus, trägt die IP-Adresse des Geräts und den Port 6454 ein. Mit „Speichern" wird das Gerät angelegt. Klickt man das Gerät im linken Fenster an, so wird es wie in Bild 17 angezeigt.

Danach klickt man auf das angelegte Farbrad, und unter den Eigenschaften am rechten Rand wird der Button „Aktion zuweisen" angeklickt. Es erscheint ein Pop-up-Fenster „Aktion zuweisen". Unter „Wähle eine Aktion" wählt man den Eintrag „Befehl ausführen" und unter „Wähle einen Quelltyp" den Eintrag „Gerät". Danach wählt man das zuvor angelegte Gerät aus und wählt unter „Wähle einen Befehl" auf der rechten Seite den Eintrag „RGB-Farbwert setzen" (Bild 18).

Über die zweite Seite, die im AIO Creator NEO angelegt wird, kann die Gartenbeleuchtung konfiguriert werden, wie es Bild 19 zeigt. Auch hier können für alle Tasten die entsprechenden Einträge unter [2] im Directory *Mediola\Einstellung.txt* gefunden werden, wobei das Anlegen wie oben beschrieben funktioniert.

Die Taste „Webseite aufrufen" verweist auf eine weitere Seite, auf der die Hauptseite der Steuerung der Gartenbeleuchtung hinterlegt ist.

Weitere Infos

[1] DMX-Schnittstelle: https://de.wikipedia.org/wiki/DMX (Lichttechnik)
[2] Beitrag Downloads: Artikel-Nr. 252719
[3] ATTINY Programmierung mit Arduino: https://arduino-projekte.webnode.at/attiny-programmierung/ und https://www.kollino.de/arduino/attiny-support-unter-arduino-1-6-installieren/
[4] ESP32 Filesystem Uploader: https://randomnerdtutorials.com/install-esp32-filesystem-up2loader-arduino-ide
[5] DMX4All Plugin: https://shop.mediola.com/aio-creator-neo/plugins/lichtschalter/35/neo-plugin-dmx4all-artnet-led-dimmer-4?c=97

Die Materialliste kann unter [2] heruntergeladen werden.

Alle Links finden Sie auch online unter: de.elv.com/elvjournal-links

Fazit

Günstige Lampen aus dem Baumarkt können mithilfe von Neopixel-LEDs und eigener Elektronik aufgepeppt werden und ein schickes Lichtambiente im Garten erzeugen. Das Ganze lässt sich per Mediola und den AIO Creator NEO über Homematic IP und das Smartphone oder den PC komfortabel steuern und automatisieren. So wird die kurzlebige Solarlampe zu einem intelligenten und schicken Accessoire im Garten und bietet für das Verweilen am Abend oder bei der Party interessante Lichteffekte. ELV