Externe Datenkommunikation mit Calliope mini

Serielle Übertragung von Daten und mittels WLAN

Thomas Meinike

Seit der ersten *Calliope*-Version lassen sich serielle Daten Byte für Byte an eine Gegenstelle senden. In diesem Projekt wird die direkte Übertragung von Daten an einen via USB-Kabel angeschlossenen Computer behandelt. Zudem wird die Datenkommunikation mit einer externen Internet-of-Things-Plattform (IoT-Plattform) unter Verwendung eines separat installierbaren WLAN-Moduls untersucht.

1 Einstieg

Der Einplatinen-Computer *Calliope mini* (Calliope gGmbH, 2025) kommt seit seiner Einführung im Jahr 2017 auch an der Hochschule Merseburg zum Einsatz, konkret in Kursen des Autors zum Thema Web-Entwicklung (Meinike, 2018). Kenntnisse allgemeiner Grundlagen zu Aufbau und Nutzung dieser Geräte werden hier vorausgesetzt. Abbildung 1 zeigt den grundsätzlichen technischen Aufbau.

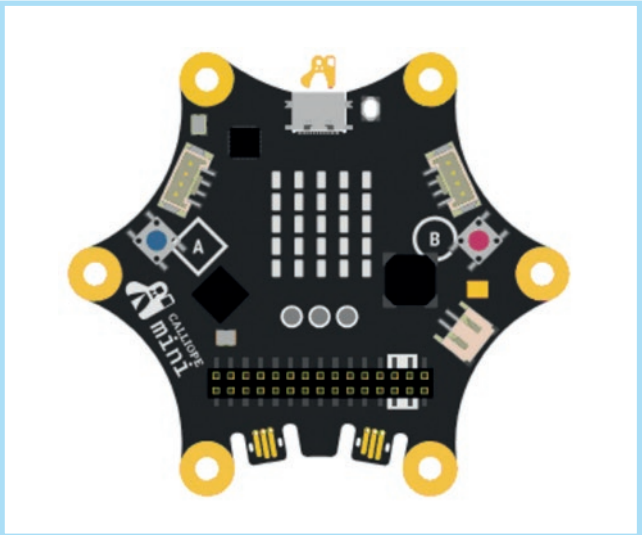

Abb. 1. Calliope mini im Überblick

2 Serielle Kommunikation

In diesem Projekt werden gesendete Daten direkt im Web-Browser empfangen. Dazu dient die von Google im *Chrome*-Browser etablierte *Web Serial API* (Beaufort, 2020; MDN Web Docs, 2025; W3C, 2025). Aktuell funktioniert das hier Gezeigte auch nur in *Chrome* und *Chromium*-Derivaten wie *Edge* und *Opera* (Can I use, 2025).

Auf einem *Calliope mini* läuft ein Programm, welches periodisch die internen Sensoren für Lichtintensität, Geräuschpegel und Temperatur sowie einen externen Sensor für Feuchtigkeitsmessungen abfragt und diese Daten als *JSON*-String aufbereitet überträgt. Es handelt sich im Folgenden jeweils um die *Calliope*-spezifische *JavaScript*-Notation (Microsoft, 2025).

2.1 Calliope mini als Sender

Der Code von Listing 1 (Meinike, 2025a) ist in der Entwicklungsumgebung *MakeCode* verfügbar und kann für die jeweilige *Calliope*-Version (V1, V2, V3) kompiliert werden. Das gilt ebenfalls für die folgenden Listings. Die jeweils erzeugte *.hex*-Datei ist nach dem Kopieren auf das per USB angeschlossene Gerät direkt lauffähig. Mit dem A-Button wird die Datenübertragung gestartet und mit dem B-Button wieder beendet.

Der voreingestellte Sendezyklus beträgt drei Sekunden. Ein Datensatz hat die Form
{ "L" : 58, "S" : 93, "T" : 25, "F" : 81 }
mit den Komponenten Licht, Sound, Temperatur und Feuchtigkeit.

```
// Daten seriell für Browser-Verarbeitung
als JSON-String ausgeben

let send: boolean = false

input.onButtonEvent(Button.A,
input.buttonEventClick(), function() {
  let L: number = 0
  let S: number = 0
  let T: number = 0
  let F: number = 0
  let JSONstr: string = ""

  serial.redirectToUSB()
  serial.setBaudRate(9600)
  send = true

  basic.forever(function() {
    if(send) {
      L = input.lightLevel()
      // 0 (dark) to 255 (bright)
      S = input.soundLevel()
      // 0 (silent) to 255 (loud)
      T = input.temperature() // °C
      F = input_moisture()    // 0 to 100 %

      JSONstr = '{ "L" : ' + L + ', "S" : ' + S
      + ', "T" : ' + T + ', "F" : ' + F + ' }'

      serial.writeLine(JSONstr)
    }
    basic.pause(3000)
  })
})

input.onButtonEvent(Button.B,
input.buttonEventClick(), function() {
  send = false
})
```

Listing 1. Calliope-Code für die serielle Browser-Kommunikation

2.2 Web-Browser als Empfänger

Abbildung 2 zeigt die im Browser erscheinende Startansicht der Anwendung zum Empfangen und Auswerten der Daten (Meinike, 2025b). Sofern die Meldung „*Web Serial API nicht unterstützt.*“ ausbleibt, kann nun mit dem *Calliope mini* kommuniziert werden.

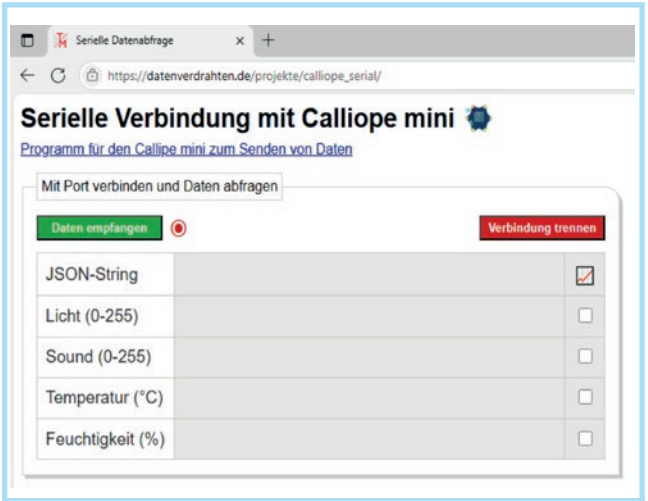

Abb. 2. Browser-Anwendung nach dem Laden

Die als *JSON*-Strings formulierten Datenpakete lassen sich im *JavaScript*-Teil Anwendung leicht mittels der Technik *JSON.parse(…)* auswerten, also die vier Einzelwerte isolieren und weiterverarbeiten. Die eingelesenen Daten werden tabellarisch dargestellt. Über die vier Checkboxen lassen sich zudem mit dem Vektorgrafik-Standard *SVG* realisierte Visualisierungen der innerhalb von sechs Minuten abgefragten Daten einblenden.

Zunächst wird die serielle Kommunikation über den in Abbildung 2 ersichtlichen Button *Daten empfangen* gestartet, wobei der verfügbare *COM*-Port gemäß Abbildung 3 je nach verwendetem System variiert.

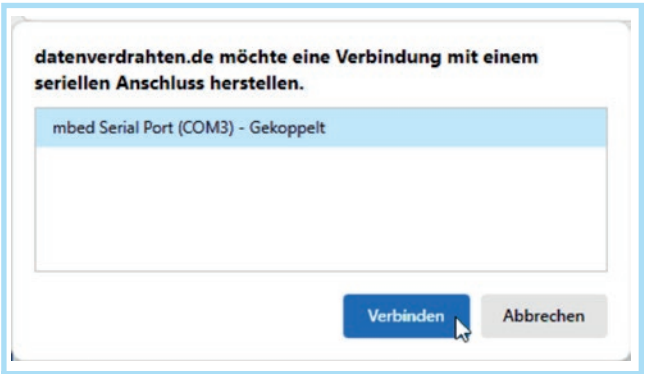

Abb. 3. Serielle Verbindung herstellen

Nach erfolgreicher Verbindung erscheinen die empfangenen Daten und die aktivierten grafischen Verläufe, wie in Abbildung 4 dargestellt.

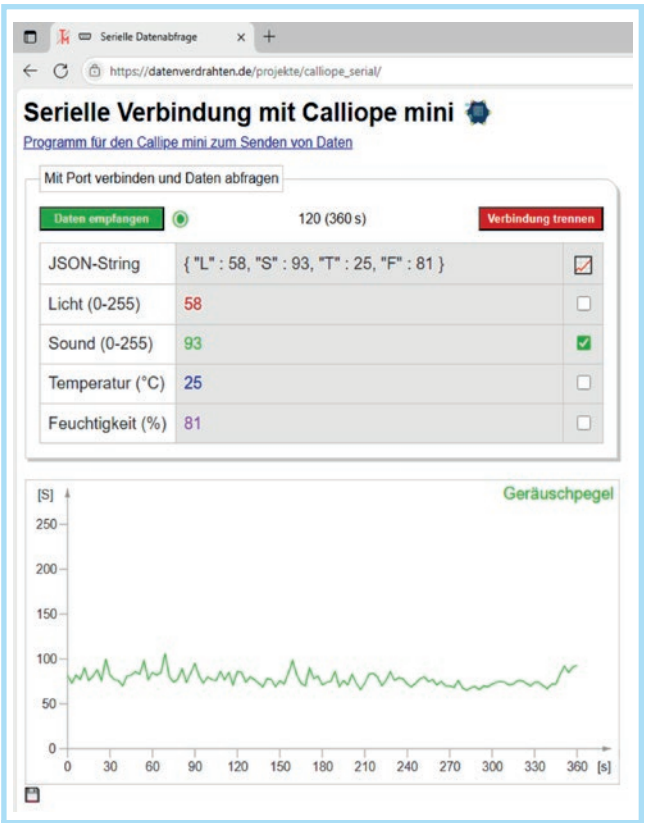

Abb. 4. Datenausgabe mit Geräuschpegel-Grafik

Unterhalb der Grafiken sind kleine Buttons für den Download der Ergebnisse als *SVG*-Dateien angeordnet. Im Download-Ordner des Systems liegen dann wahlweise die Dateien *Lgraph_dl.svg*, *Sgraph_dl.svg*, *Tgraph_dl.svg* und *Fgraph_dl.svg* vor.

Nach Ablauf der sechs Minuten werden die Ausgaben nicht mehr verändert, wobei die eigentliche Datenkommunikation weiterläuft, siehe den erreichten Endstatus in Abbildung 4. Aus Sicht des Browsers kann der Empfang über den Button *Verbindung trennen* beendet werden. Das Senden vom *Calliope mini* aus lässt sich durch Betätigen des B-Buttons stoppen.

Details der im Browser laufenden *JavaScript*-Implementierung erschließen sich über das mitgelieferte Skript *calliope_serial.js*, welches Datenverarbeitung und Ergebnisausgabe koordiniert.

2.3 Anwendungsszenario

In Abbildung 5 ist das praktische Prinzip der Kopplung eines *Calliope mini* an den mittels USB-Kabel angeschlossenen Computer und zusätzlich mit dem in einer Topfpflanze platzierten Feuchtigkeitssensor realisiert. Im Browserfenster sind Zahlenwerte der abgelaufenen Messung und speziell der zu den Feuchtigkeitswerten aufgezeichnete *SVG*-Graph ersichtlich.

Abb. 5. Messplatz mit Topfpflanze und Calliope mini V1

2.4 Details zum Feuchtigkeitssensor

Verwendet wurde der *Moisture Sensor V1.4* von *Grove* (Seeed Studio, 2025), vgl. Abbildung 6. Dieser eignet sich z.B. für Messungen der Feuchtigkeit in Bodensubstraten wie Blumenerde und ist auch aus einschlägigen Schulexperimenten zu trockenen bzw. gegossenen Topfpflanzen bekannt.

Der Sensor wird am Port A1 oberhalb des B-Buttons angeschlossen. Gemessen wird dort über den Analog-Pin C16. Brauchbare Ergebnisse erfordern eine Kalibrierung. Dazu werden die Daten des Sensors in Luft sowie in Wasser bestimmt. Im Test ergaben sich dafür an einem *Calliope mini V1* die Basiswerte 255 und 640. Darüber lassen sich Messwerte zwischen 0 und 100 % ableiten.

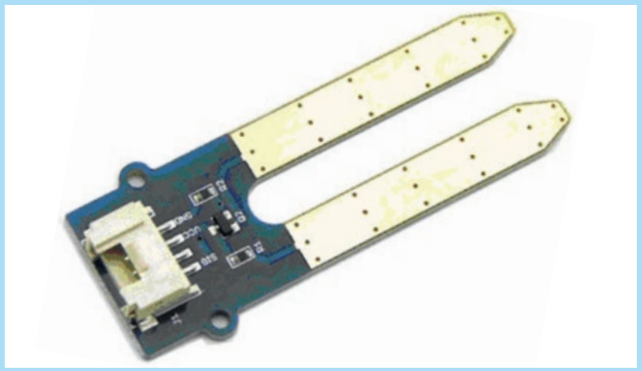

Abb. 6. Moisture Sensor V1.4

Bei kontinuierlichen Messungen auftretende Schwankungen lassen sich durch mehrere Einzelmessungen ausgleichen. Als praktikabel erwies sich die Bestimmung von zehn Einzelwerten in kurzer Abfolge (hier 100 ms) und Verwendung des größten Wertes. Der Mittelwert taugt hingegen nicht, da das Signal durchaus bis auf Nullwerte abfällt. Listing 2 enthält die zusätzlich im Code von (MEINIKE, 2025a) enthaltene Funktion *input_moisture()*.

```
// Funktion input_moisture()

function input_moisture(): number {
  // Kalibrierung
  const dry_value: number = 255
  const wet_value: number = 640
  const min_value: number = 0
  const max_value: number = 100

  // Maximalwert aus 10 Messungen
  const moiarr: number[] = []
  for(let i: number = 0; i < 10; i++) {
    moiarr[i] = pins.analogReadPin(AnalogPin.C16)
    basic.pause(100)
  }

  moiarr.sort(function(a: number, b: number):
  number { return b - a })
  const raw_value: number = moiarr[0]
  const cal_value: number = Math.round
  ((raw_value - dry_value) /
   (wet_value - dry_value) * max_value)

  if(cal_value < min_value) {
    return min_value
  }
  else if(cal_value > max_value) {
    return max_value
  }
  else {
    return cal_value
  }
}
```

Listing 2. Funktion *input_moisture()* zur Feuchtigkeitsmessung

Zum Testen der Sensordaten auf Reproduzierbarkeit wurde eine weitere Anwendung erstellt, welche die Werte auf dem 5x5-LED-Display veranschaulicht (MEINIKE, 2025c). Eine LED repräsentiert 4 % anteilig zu 100 %. Befindet sich der Sensor in Wasser, leuchten alle 25 LEDs, dagegen sind in Luft alle inaktiv. Abbildung 7 demonstriert das Prinzip anhand einer Messung im Fruchtfleisch einer Melone. Die 17 aktiven LEDs gehören entsprechend zum Messwert von 68 %.

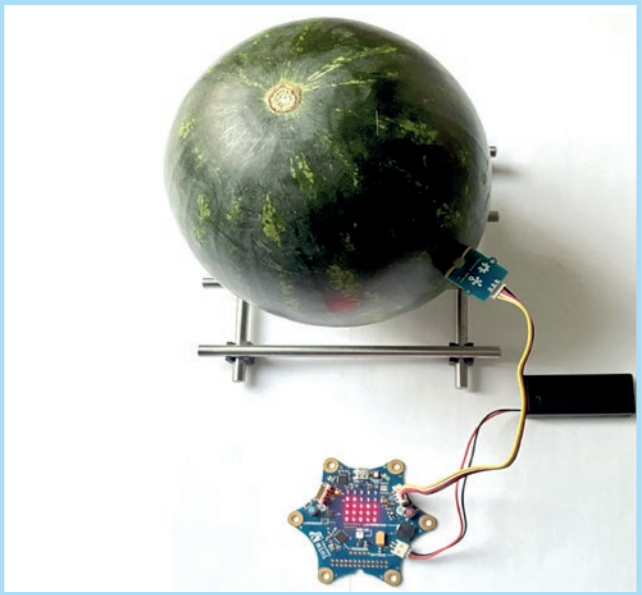

Abb. 7. Feuchtigkeitsmessung mit Calliope mini V1 in einer Melone

Listing 3 enthält den relevanten Code aus (MEINIKE, 2025c) für die Schaltung der LEDs.

```
// Daten des Feuchtigkeits-Sensors auf LED-Matrix
ausgeben

// ...
F = input_moisture()  // 0 to 100 %
N = Math.round(F / 4) // Anzahl LEDs
setLEDs(N)
// ...

function setLEDs(n: number): void {
  basic.clearScreen()
  let i: number = 0

  for(let row: number = 4; row >= 0; row--) {
    for(let col: number = 0; col <= 4; col++) {
      if(i < n) {
        led.plot(col, row)
        i++
      }
      else return
    }
  }
}
```

Listing 3. Funktion *setLEDs(n)* für die Schaltung der LED-Matrix

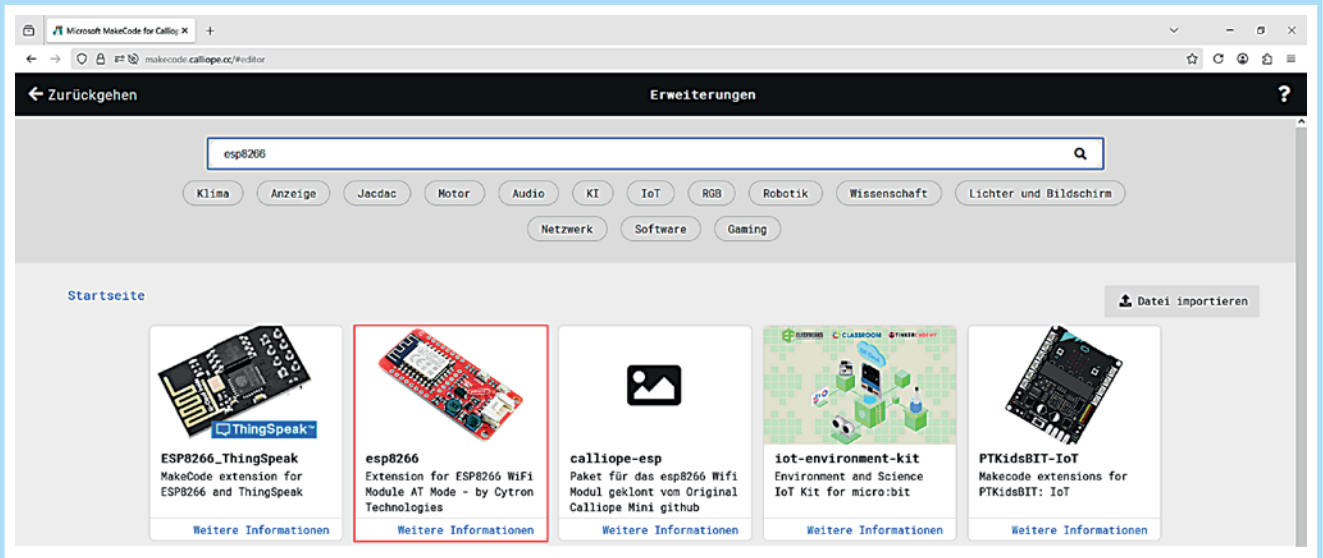

Abb. 8. ESP8266-Erweiterungen für MakeCode

3 Datenkommunikation über WLAN

3.1 WLAN-Modul

Für den *Calliope mini* gibt es neben den bereits integrierten Sensoren weitere zukaufbare Komponenten, etwa zur Messung des CO_2-Anteils in der Raumluft und eine Reihe an Zusätzen für Steuerungsaufgaben. Für die allgemeine Verarbeitung von Daten ist das WLAN-Modul mit der Typenbezeichnung *ESP8266* hilfreich. Dafür existieren einige Software-Erweiterungen für die Entwicklungsumgebung *MakeCode*.

Mit dem in Abbildung 8 rot umrandeten Paket von *Cytron Technologies* (Cytron Technologies, 2024) ließ sich das am Port A1 angeschlossene WLAN-Modul praktikabel einsetzen. Es bietet Unterstützung für die Anwendungen *Blynk*, *Internet Time*, *Telegram* und *ThingSpeak*.

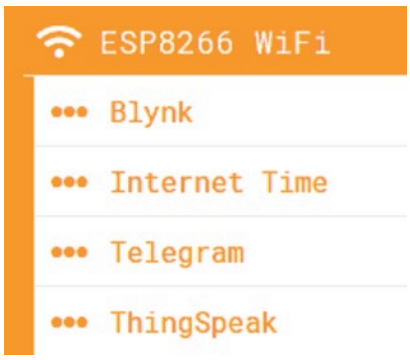

Abb. 9. Vom Paket ESP8266 WiFi unterstütze Zielanwendungen

Für die in Abbildung 9 ersichtlichen Ziele sind vorgefertigte Funktionsaufrufe zur Verbindung sowie zum Senden bzw. Empfangen von Daten verfügbar. Zunächst muss jeweils die Adressierung der Anschlusspins am *Calliope mini* gesetzt werden (konkret *Tx* = *P17*, *Rx* = *P16*). Für die WLAN-Nutzung werden *SSID* und *Passwort* benötigt. Als besonders komfortabel erwies sich die Verwendung eines auf dem Smartphone aktivierten mobilen Hotspots.

3.2 Verbindung mit ThingSpeak

Als Gegenstelle wurde die IoT-Plattform *ThingSpeak* (MathWorks, 2025) eingesetzt. Dort kann man sich nach dem Anlegen eines Accounts diverse Diagramme und Widgets zur Visualisierung der übertragenen Daten einrichten. Interessant ist, dass *ThingSpeak* von der Firma *MathWorks* betrieben wird, welche auch für die an der Hochschule eingesetzte Software *MATLAB* bekannt ist. Ein dafür bereits existierender Account erlaubte den Zugriff.

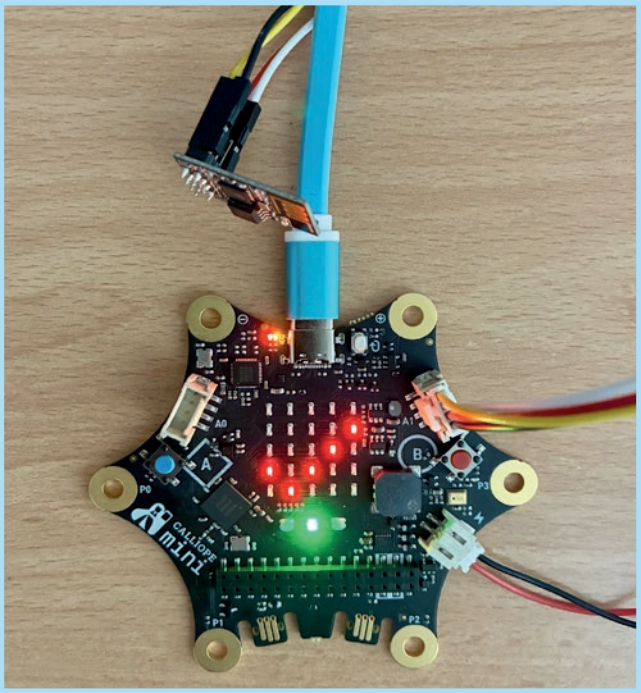

Abb. 10. Calliope mini V3 mit WLAN-Modul in Aktion

Zur Verbindung werden für das Profil zwei Schlüssel generiert: *Read API Key* und *Write API Key*. Mit diesen kann dann über den

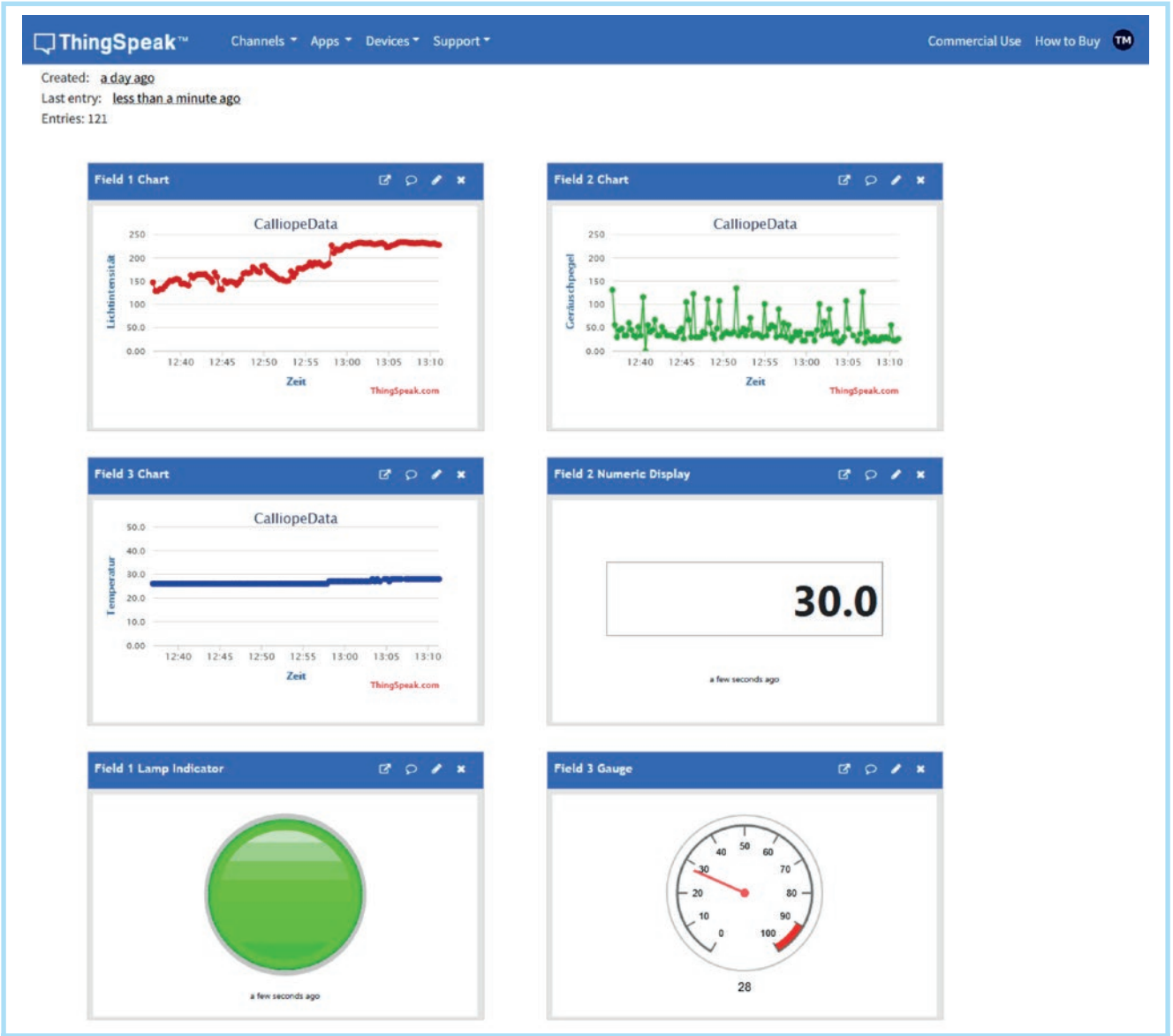

Abb. 11. Visualisierungen von an ThingSpeak gesendeten Daten

auf dem *Calliope mini* laufenden Code kommuniziert werden. Insgesamt können acht Datenfelder belegt werden. Im Beispiel wurden periodisch die drei Datenpunkte für Licht, Sound und Temperatur übermittelt. Als minimaler Abstand zwischen den Übertragungen sind 15 Sekunden vorgesehen und so auch im Code verankert (15000 ms). Listing 4 umfasst das zum Senden von Daten entwickelte Programm (MEINIKE, 2025d).

```
// Daten via WLAN-Modul an ThingSpeak-Server
übertragen

let send: boolean = false

esp8266.init(SerialPin.P17, SerialPin.P16, BaudRate.
BaudRate115200)

if(esp8266.isESP8266Initialized()) {
  esp8266.connectWiFi("ssid", "password")
}
```

```
input.onButtonEvent(Button.A,
input.buttonEventClick(), function() {
  let L: number = 0
  let S: number = 0
  let T: number = 0
  send = true

  basic.forever(function() {
    if(send) {
      basic.showLeds(`
        . . . . .
        . . . . .
        . . # . .
        . . . . .
        . . . . .
      `)

      L = input.lightLevel()
      // 0 (dark) to 255 (bright)
```

```
      S = input.soundLevel()
      // 0 (silent) to 255 (loud)
      T = input.temperature()
      // °C

      esp8266.uploadThingspeak("writeApiKey",
      L, S, T, 0, 0, 0, 0, 0)

      if(esp8266.isThingspeakUploaded()) {
        basic.showLeds(`
          . . . . .
          . . . . #
          . . . # .
          # . # . .
          . # . . .
        `)
      }
    }
    basic.pause(15000)
  })
})

input.onButtonEvent(Button.B,
input.buttonEventClick(), function() {
  send = false
  basic.clearScreen()
})

basic.forever(function() {
  if(esp8266.isWifiConnected()) {
   basic.setLedColors(0x000000, 0x00ff00, 0x000000)
  }
  else {
   basic.setLedColors(0x000000, 0xff0000, 0x000000)
  }
})
```

Listing 4. Calliope-Code für die ThingSpeak-Kommunikation

3.3 Umsetzung und Ergebnis

Abbildung 10 zeigt die Ansicht des auf dem Gerät laufenden Projektes. Oberhalb befindet sich das in Nachbarschaft des A-Buttons angeschlossene *ESP8266*-Modul. Die grüne LED ist aktiv, sofern eine WLAN-Verbindung besteht. Der symbolische rote Haken erscheint jeweils nach einem erfolgreichen Daten-Upload.

Einen Eindruck vom visuellen Ergebnis der Datenkommunikation mit dem *ThingSpeak*-Server vermittelt Abbildung 11.

4 Fazit und Ausblick

Der *Calliope mini* bietet ein breites Einsatzspektrum insbesondere für Lehrzwecke im MINT-Bereich. Die hier aufgezeigten Ideen und Umsetzungen können in diesem Kontext ohne größere Aufwände nachvollzogen werden. Es werden nur die *Calliope*-Basisgeräte und zwei zusätzliche externe Module im Preisbereich von jeweils ca. zehn Euro benötigt. Der erstellte Code und die vorkompilierten Anwendungen stehen unter (Meinike, 2025e) zum Download zur Verfügung.

mit online www.mnu.de ergänzung

Literatur

Beaufort, F. (2020). Aus einem seriellen Port lesen und darauf schreiben. Chrome für Entwickler.
https://developer.chrome.com/docs/capabilities/serial?hl=de

Calliope gGmbH (2025). Calliope mini – der kleine Computer für große Ideen!
https://calliope.cc/

Can I use (2025). Web Serial API.
https://caniuse.com/web-serial

Cytron Technologies (2024). ESP8266 AT Mode Extension for Microsoft MakeCode.
https://makecode.calliope.cc/pkg/cytrontechnologies/pxt-esp8266

Seeed Studio (2025). Grove – Moisture Sensor V1.4.
https://wiki.seeedstudio.com/Grove-Moisture_Sensor/

MathWorks (2025). ThingSpeak for IoT Projects.
https://thingspeak.mathworks.com/

MDN Web Docs (2025). Web Serial API.
https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API

Meinike, T. (2018). Von den Kleinen lernen – Calliope mini als Türöffner zur Programmierung. Vortrag auf der tekom-Jahrestagung, Stuttgart 2018.
https://tinyurl.com/2ehzu63t

Meinike, T. (2025a). MakeCode-Anwendung: Daten seriell für Browser-Verarbeitung als JSON-String ausgeben.
https://makecode.calliope.cc/_UzXdCpXYVPKj

Meinike, T. (2025b). Projekt-Website: Serielle Verbindung mit Calliope mini.
https://datenverdrahten.de/projekte/calliope_serial/

Meinike, T. (2025c). MakeCode-Anwendung: Daten des Feuchtigkeits-Sensors auf LED-Matrix ausgeben.
https://makecode.calliope.cc/_iq72AkA3cDPb

Meinike, T. (2025d). MakeCode-Anwendung: WLAN-Kommunikation mit ThingSpeak über ESP8266-Modul.
https://makecode.calliope.cc/_ggvEmyVPRWEs

Meinike, T. (2025e). Download-Paket zu den behandelten Projekten.
https://datenverdrahten.de/projekte/calliope_serial/calliope_download.zip

Microsoft (2025). MakeCode Docs: JavaScript.
https://makecode.calliope.cc/javascript

W3C (2025). Web Serial API Draft – Draft Community Group Report.
https://wicg.github.io/serial/

Dr. Thomas Meinike, thomas.meinike@hs-merseburg.de, ist als Lehrkraft für besondere Aufgaben an der Hochschule Merseburg tätig. Seine Arbeitsschwerpunkte in Lehre und Forschung sind XML-Anwendungen in der Technischen Kommunikation, Onlinehilfen und Webentwicklung.