



Bachelorarbeit

Benutzeroberfläche und Ansteuerung eines Batterie-Monitor-Systems auf Basis des One-Wire Bus

User interface and control of a battery monitor system based on
the One-Wire Bus

Studiengang Elektrotechnik - Elektrische Energietechnik
an der Fakultät für Informations-, Medien- und Elektrotechnik
der Technischen Hochschule Köln

Vorgelegt von: Mohammed Amine, Jazouli
Matrikel-Nr.: 11114370

Eingereicht bei: Prof. Dr. Eberhard Waffenschmidt
Zweitgutachter/in: Prof. Dr. Ingo Stadler

Köln, 30.08.2021

Danksagung

An dieser Stelle möchte ich die Möglichkeit nutzen, mich bei allen bedanken, die mich bei dieser Arbeit unterstützt haben. Ich bedanke mich bei allen, die mich von nah und fern geholfen, unterstützt und ermutigt haben, bei allen, die bei der Realisierung dieses Projekts an mich geglaubt haben. Hierzu zähle ich sowohl meine Familie, Arbeitskollegen, als auch meine Freunde, die mir stets zur Seite standen, mich motiviert und in belastenden Momenten wieder aufgebaut haben.

Da meine Arbeit fertiggestellt ist, möchten ich Herrn Prof. Dr. Eberhard Waffenschmidt für die Qualität seiner Betreuung danken, die auf einem hohen Maß an Kompetenz, Erfahrung und einer fortschrittlichen Arbeitsmethode basiert. Vielen Dank auch an Herr Prof. Dr. Ingo Stadler dass er meine Abschlussarbeit begutachtet.

Danke an Herr Sergej Baum für die Bestellungen und bereitstellen der Material für mein Abschlussprojekt.

Zum Abschluss möchte ich in besonderer Weise bei meinem Vater bedanken, den ich in diesen Schwierigen Zeiten verloren habe und ich bedanke mich bei meiner Mutter die mich im Laufe meines Studiums bis zur Fertigung Stellung meiner Arbeit unterstützt hat.

Kurzfassung/Abstract

Ziel dieser Arbeit war es, eine Benutzeroberfläche und Ansteuerung eines Battery-Monitor-Systems auf Basis des One-Wire Bus für die entwickelte 1-Wire-Battery-Monitor-Schaltung zu erstellen. Mithilfe eines Mikrocontrollers und des 1-Wire-Kommunikationsprotokolls wurde der 1-Wire-Battery-Monitor mit dem integrierten Smart-Battery-Monitor-Chip DS2438 für die Einzelüberwachung von Batterien, bisher im Praxisprojekt in Betrieb genommen, optimiert und für eine spätere Implementierung bereitgestellt. Ferner sollen die Funktionsweise des 1-Wire-Kommunikationsprotokolls und die methodische Vorgehensweise für die Inbetriebnahme und die Visualisierung erläutert werden.

Der 1-Wire-Battery-Monitor soll letztlich die Temperatur und Spannung messen können. Zudem soll er zunächst für jede einzelne Platine die ID-Nummer oder sogenannte ROM-Adresse von den Sensoren anzeigen, um einfacher nachvollziehen zu können, welche Batterie zurzeit gemessen wird. Anhand der im Theorieteil erworbenen Informationen sowie der im praktischen Teil gewonnenen Kenntnisse soll überdies eine Handlungsempfehlung für zukünftige weitere Entwicklungen des Battery Monitors ausgesprochen werden.

Abstract

The aim of this work was to create a user interface and control of a battery monitor system based on the one-wire bus for the developed 1-wire battery monitor circuit. With the help of a microcontroller and the 1-Wire communication protocol, the 1-Wire battery monitor with the integrated smart battery monitor chip DS2438 for the single monitoring of batteries, previously put into operation in the practical project, was optimized and made available for later implementation. Furthermore, the functionality of the 1-Wire communication protocol and the methodical approach for commissioning and visualization will be explained.

The 1-Wire battery monitor should ultimately be able to measure temperature and voltage. In addition, it should first display the ID number or so-called ROM address from the sensors for each individual board in order to more easily understand which battery is currently being measured. Based on the information acquired in the theoretical part and the knowledge gained in the practical part, a recommendation for future developments of the Battery Monitor will also be made.

Inhaltsverzeichnis

Danksagung	II
Kurzfassung/Abstract	III
Inhaltsverzeichnis	IV
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Methodisches Vorgehen	2
2 Technische Grundlagen	3
2.1 Hardware	3
2.1.1 Was ist 1-Wire?	3
2.1.2 Algorithmus des 1-Wires	4
2.1.3 1-Wire-Sensoren.....	6
2.1.4 Arduino	8
2.1.5 Ethernet Shield 2 W550	10
2.2 Software.....	12
2.2.1 Programmierung	12
2.2.2 LabVIEW	13
3 1-Wire Battery Monitor Schaltung	16
3.1 1-Wire-Battery-Monitor mit Potentialtrennung	16
3.2 Das neue Layout.....	17
3.3 Bestellung und Fertigung der Leiterplatten.....	18
4 Algorithmus und Methode für die Inbetriebnahme	20
4.1 1-Wire Master-Slave-Kommunikation	20
5 Messung und Auswertung	21
5.1 Auslesen der Temperatur und Spannung mit dem Arduino Uno.....	21
5.2 Auswertung der Daten und Diskussion	22
6 Versuchsaufbau	24
6.1 Platinen-Design für den Aufbau	24
6.2 Montage der 1-Wire-Battery-Monitor-Platinen	25
7 Benutzeroberfläche und Einrichten von Arduino	26
7.1 Eingesetzte Softwareanwendungen	26
7.2 Entwurf und Erstellung der Benutzeroberfläche	26
7.2.1 USB Schnittstelle:	26
7.2.2 TCP/IP-Schnittstelle.....	28
8 Visualisierung und Diskussion	31
8.1 Diskussion	32

9 Fazit und Handlungsempfehlung	33
Literaturverzeichnis	34
Abkürzungsverzeichnis	36
Tabellenverzeichnis	37
Abbildungsverzeichnis	38
Anhang	40
Eidesstattliche Erklärung	48

1 Einleitung

1.1 Motivation

Die Batterie ist bekannt für ihre Fähigkeit, elektrische Energie in Form von nutzbarer Energie zu speichern, die bei Bedarf verwendet werden kann. Sie ist ein unverzichtbarer Speicher von Energie für elektronische Komponenten, kleine elektronische Geräte und große Systeme wie erneuerbare Energien. Kleine elektronische Geräte wie Video-/Audiogeräte, medizinische Geräte, Elektrowerkzeuge, Messgeräte und Datenlogger sowie Fernsensoren sind typischerweise mit Batterien ausgestattet. Die in diesen Geräten installierten Batterien befreien die Benutzer von der Notwendigkeit einer externen Stromzufuhr und ermöglichen ihnen eine tragbare Anwendung. Mit der Zeit lässt die gespeicherte Energie der in diesen Systemen installierten Batterien nach, infolgedessen Letztere aufgeladen oder getauscht werden müssen, um die Kapazität wiederherzustellen. Dies hat dazu geführt, dass Batterien stets weiterentwickelt und verbessert wurden, um längere Laufzeiten zu ermöglichen. Offensichtlich werden Batterien nicht nur in kleinen elektronischen Geräten verwendet, vielmehr werden sie heute häufig als Energie-Backup, wie z. B. in Form von unterbrechungsfreien Stromversorgungen (USVs), genutzt und gelten als die beste und zuverlässigste Energiequelle bzw. Energieersatz, wenn es zu Stromausfällen kommt. [1]

Die am schwächsten belastete Batterie in einem Strang spielt eine wichtige Rolle für die Lebensdauer des gesamten Systems. Tatsächlich beträgt die Lebensdauer des Strangs oft nur einen Bruchteil der vom Hersteller erwarteten Lebensdauer. Die ständige Überwachung und der Ausgleich der einzelnen Ladespannungen helfen, die Verfügbarkeit des Batteriesystems sicherzustellen. Im Wege der Batterieüberwachung werden wichtige Daten geliefert, mithilfe deren festgestellt werden kann, ob die Reservestrombatterien die erwartete Leistung erbringen, wenn die kritische Stromversorgung unterbrochen wird. Ohne ordnungsgemäße Überwachung des Batteriesystemstatus besteht die Gefahr eines katastrophalen Ausfalls [2]. Aufgrund der schädlichen Auswirkungen von Über- und Unterspannungen ist es jedoch notwendig, die Batterie und die Verbraucher in instabilen Versorgungsnetzen durch regelmäßige Spannungs- sowie Temperaturmessungen vor Zerstörungen zu schützen.

In der vorliegenden Arbeit werden die Funktionsweise und die Methode vorgestellt, anhand welcher der Entwicklungsprozess des Überwachungssystems für die Batterie umgesetzt wurde. Mit Hilfe des 1-Wire-Bus-Kommunikationsprotokolls und der von Prof. Dr. Eberhard Waffenschmidt der Fachhochschule Köln entwickelten 1-Wire-Battery-Monitor-Schaltung wurde das Überwachungssystem realisiert und in Betrieb genommen. Im Weiteren soll eine Benutzeroberfläche mit LabVIEW programmiert werden, um eine Visualisierung zu ermöglichen.

1.2 Zielsetzung

Ziel dieser Arbeit war es, mithilfe eines Mikrocontrollers und des 1-Wire-Kommunikationsprotokolls den entwickelten 1-Wire-Battery-Monitor mit dem integrierten Smart-Battery-Monitor-Chip DS2438 für die Einzelüberwachung von Batterien in Betrieb zu nehmen, zu optimieren und eine spätere Implementierung zu ermöglichen. Der 1-Wire-Battery-Monitor soll letztlich die Temperatur und Spannung messen können. Ferner soll er zunächst für jede einzelne Platine die ID-Nummer oder sogenannte ROM-Adresse von den Sensoren anzeigen, damit einfacher zu erkennen ist, welche Batterie zurzeit gemessen wird.

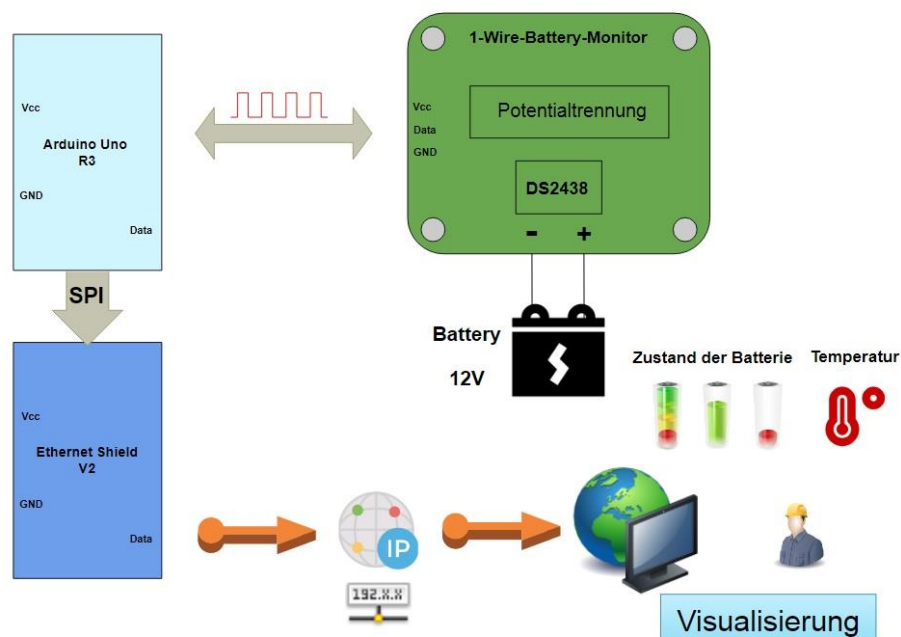


Abbildung 1 Battery Monitoring System – Eigene Darstellung[3]

1.3 Methodisches Vorgehen

Die vorliegende Arbeit gliedert sich in drei Teile. Der erste Abschnitt behandelt die Grundlagen des 1-Wire-Kommunikationsprotokolls, der 1-Wire-Sensoren sowie der prinzipiellen Mikrocontrollertechnik. Gegenstand des zweiten Abschnitts wiederum sind die Funktionsweise und die Optimierung der 1-Wire-Battery-Monitor-Schaltung die im Praxisprojekt vorgelegt ist. Zuletzt werden die Inbetriebnahme und die Kommunikation zwischen Master (Mikrocontroller) und Slave (1-W-Battery-Monitor) erklärt, die eingesetzten Softwareanwendungen für die Visualisierung präsentiert sowie Handlungsempfehlungen für eine weitere Forschungsentwicklung vorgeschlagen.

2 Technische Grundlagen

2.1 Hardware

2.1.1 Was ist 1-Wire?

Der 1-Wire Bus ist ein Kommunikations-Bussystem für die Bereitstellung von Daten, Signalen und Energie über ein einziges Signal mit niedrigen Datenraten, einer hohen Auflösung und einer großen Reichweite. Er wird typischerweise für die Kommunikation mit kleinen, preiswerten Geräten verwendet, wie Temperatursensoren, die als Slave mit dem Master (Mikroprozessor) arbeiten. Das 1-Wire-Bussystem liefert das ausreichende Steuer- und Betriebssignal, eine eindeutige ID-Seriennummer für jeden Sensor und unterstützt zugleich mehrere Temperatursensoren durch eine treibende Leistung (Parasite Power) auf einer einzigen Leitung. Die Temperatursensoren werden beim 1-Wire-Bussystem von zwei Arten von Stromversorgungen versorgt, externe Stromversorgung und Parasitic Power [4]. Es wird lediglich ein einziger Pull-up-Widerstand im Wert von $4,7\text{ k}\Omega$ benötigt, unabhängig davon, wie viele Geräte sich auf dem Bus befinden. Abbildung 2 zeigt, wie drei Sensoren auf dem gleichen Bus parallel mit einem Mikroprozessor angeschlossen sind. [5]

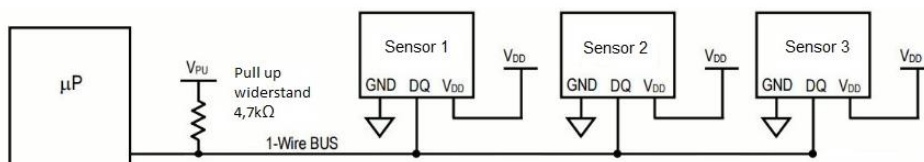


Abbildung 2 Parallelschaltung mit einem Mikrocontroller von drei 1-Wire Sensoren – Eigene Darstellung [6]

Das 1-Wire-Protokoll ermöglicht die Interaktion mit vielen Maxim-/Dallas-Semiconductor-Bauteilen, einschließlich Batterie und Wärmemanagement, Geräte, Speicher, iButtons®, etc. 1-Wire-Geräte bieten Lösungen für Identifikation, Speicher, Zeitmessung, Messung und Steuerung. Die 1-Wire-Datenschnittstelle ist auf das absolute Minimum reduziert (eine Datenleitung mit Massebezug). Da die meisten 1-Wire-Geräte eine relativ geringe Datenmenge zur Verfügung stellen, ist die typische Datenrate von 16 kbit/s hinreichend für die gesamte Arbeit. Oft ist es günstig, wenn ein GPIO Pin eines 8-Bit- oder 16-Bit-Mikrocontrollers im Rahmen einer „Bit-Banging“-Methode als Bus-Master fungiert. 1-Wire-Geräte kommunizieren über eine einzige Datenleitung und gut definierte, erprobte Protokolle. [7]

Weiterhin verfügt der 1-Wire-Bus über einen sehr hochentwickelten Adressierungsmechanismus, mithilfe dessen ermittelt werden kann, welche Geräte an einer einzelnen 1-Wire-Implementierung angeschlossen sind. Dies ist ein Algorithmus, dessen Design äußerst wissenswert ist. [8]

Die Vorteile des 1-Wire-Bussystems: [9]

- Ein einziger Kontakt genügt zur Steuerung und Bedienung
- Eindeutige ID werkseitig in jedem Gerät integriert
- Spannungsversorgung vom Datenbus abgeleitet („Parasitically Powered“)
- Multi-Drops-fähig: Unterstützt mehrere Geräte an einer Leitung

Einige Anwendungen des 1-Wire-Busses:[9]

- Druckerpatronen-ID
- ID medizinischen Verbrauchsmaterials
- Identifikation und Steuerung von Zubehör/Peripheriegeräten
- IP-Schutz, sichere Funktionskontrolle, Klon-Verhinderung
- Temperatur- und Spannungssensor

2.1.2 Algorithmus des 1-Wires

Die 1-Wire-Geräte von Maxim verfügen jeweils über eine eindeutige 64-Bit-Registrierungsnummer im Festwertspeicher (ROM) (Abbildung 1), über die sie von einem 1-Wire-Master in einem 1-Wire-Netzwerk individuell angesprochen werden können. Sofern die ROM-Nummern der Slave-Geräte im 1-Wire-Netzwerk nicht bekannt sind, können sie mit Hilfe eines Suchalgorithmus ermittelt werden. Dieser Algorithmus gilt für alle aktuellen und zukünftigen Geräte, die über eine 1-Wire-Schnittstelle verfügen. [10]

64-Bit 'Registration' ROM Number					
MSB				LSB	
8-Bit CRC		48-Bit Serial Number		8-Bit Family Code	
MSB	LSB	MSB	LSB	MSB	LSB

Abbildung 3 64-Bit eindeutige ROM-"Registrierungsnummer". [10]

Der Suchalgorithmus ist eine binäre Baumsuche, bei der Zweigen gefolgt wird, bis eine Geräte-ROM-Nummer oder ein Blatt gefunden wird. Nachfolgende Suchen nehmen daraufhin die restlichen Verzweigungspfade, bis alle vorhandenen Blätter entdeckt sind.

Der Suchalgorithmus beginnt damit, dass die Geräte am 1-Wire mit der Reset- und Präsenz-Impulsfolge zurückgesetzt werden. Sofern dies erfolgreich ist, wird der 1-Byte-

Suchbefehl gesendet. Letzterer bereitet die 1-Wire-Geräte darauf vor, die Suche zu beginnen. Diesbezüglich sind zwei Arten von Suchbefehlen zu differenzieren. Während der normale Suchbefehl (F0 hex) eine Suche mit allen teilnehmenden Geräten durchführt, nimmt der Alarmbefehl bzw. bedingte Suchbefehl (EC hex) eine Suche nur mit Geräten vor, die sich in einer Art Alarmzustand befinden. Aufgrund dessen wird der Such-Pool reduziert, um schnell auf Geräte zu reagieren, die Aufmerksamkeit benötigen.

Nach dem Suchalgorithmus muss der 1-Wire-Master dann ein Bit an die teilnehmenden Geräte zurücksenden. Wenn das teilnehmende Gerät diesen Bit-Wert hat, setzt es seine Teilnahme fort. Wenn es den Bit-Wert nicht hat, geht es in einen Wartezustand über, bis der nächste 1-Wire-Reset erkannt wird. Dieses Verfahren "read two bits" und "Write one bit" wird dann für die restlichen 63 Bits der ROM-Nummer wiederholt (siehe Tabelle 1).
[10]

Tabelle 1 1-Wire Master und Slave Suchablauf [10]

Master	Slave
1-Wire-Reset-Stimulus	Präsenz-Impuls erzeugen
Suchbefehl schreiben (normal oder Alarm)	Jeder Slave bereitet sich auf die Suche vor
Lesen 'AND' von Bit 1	Jeder Slave sendet Bit 1 seiner ROM-Nummer
Lesen 'AND' von Komplement bit 1	Jeder Slave sendet das Komplementbit 1 seiner ROM-Nummer.
Schreiben von Bit 1 Richtung (gemäß Algorithmus)	Jeder Slave empfängt das vom Master geschriebene Bit. Entspricht das Bit nicht gleich dem Bit 1 seiner ROM-Nummer, geht es in einen Wartezustand
Lesen 'AND' von Bit 64	Jeder Slave sendet Bit 64 seiner ROM-Nummer
Lesen 'AND' des Komplement-Bits 64	Jeder Slave sendet das Komplementbit 64 seiner ROM-Nummer
Schreiben von Bit 64 Richtung (gemäß Algorithmus)	Jeder Slave empfängt das vom Master geschriebene Bit. Sofern das gelesene Bit nicht mit dem Bit 64 seiner ROM-Nummer übereinstimmt, geht es in einen Wartezustand

2.1.3 1-Wire-Sensoren

1-Wire-Produkte bieten Kombinationen von Speicher-, Mixed-Signal und sicheren Authentifizierungs-Funktionen mit vollständigem Betrieb über eine einzige serielle Kontakt-schnittstelle. Aufgrund der Stromversorgung und der Kommunikation, die über das serielle Protokoll übertragen werden, sind die 1-Wire-Geräte in ihrer Fähigkeit unerreicht, die Schlüsselfunktionen für Anwendungen bereitzustellen, bei denen die I/O-Ressourcen des Mikrocontrollers begrenzt sind oder die Systemverbindung minimiert werden muss. [11]

Die 1-Wire-Slaves können in folgende drei Gruppen aufgeteilt werden:

A. Echte Chips:

Wie in den Abbildungen 4 und 5 erkennbar, sei als Beispiel der DS18B20 genommen, der über 3 Pins verfügt (VCC, Data, GND) und als Temperatursensor gilt. Rechts daneben befindlich ist der Smart-Battery-Monitor-Chip DS2438, der insgesamt 8 Pins aufweist und die Temperatur ebenso wie die Spannung messen kann.

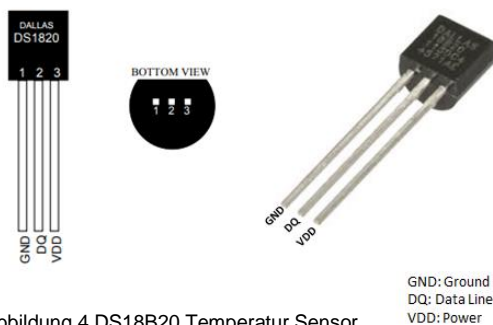


Abbildung 4 DS18B20 Temperatur Sensor

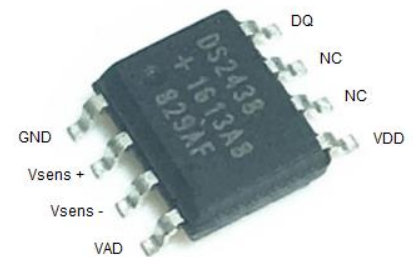


Abbildung 5 DS2438 Smart Battery Monitor Chip

Gegenwärtig sind ungefähr 44 einzelne Chips mit unterschiedlichen Funktionen verfügbar, wie die Abbildung 5 zeigt. [12]



Abbildung 6 1-Wire Produkte [12]

❖ Der DS18B20 Temperatursensor:

Der DS18B20 ist in zwei verschiedenen Bauformen erhältlich, als einfache sowie als wasserdichte digitale Variante. Deren Pin-Belegung ist in der folgenden Abbildung 6 dargestellt. Der DS18B20 kann die Temperatur von $-55\text{ }^{\circ}\text{C}$ bis $+125\text{ }^{\circ}\text{C}$ mit einer Genauigkeit von $\pm 5\%$ messen. [13]



Abbildung 7 DS18B20 Pin Darstellung [13]

❖ Der DS2438:

Der DS2438 wird Smart-Battery-Monitor genannt und verfügt über viele Funktionen. So kann er über die Temperaturmessung hinaus die Spannung an einer Batterie messen sowie die Auf- und Entladung durch den Stromeingang ($V_{\text{sens+}}/V_{\text{sens-}}$) kontrollieren. Er bietet eine eindeutige 64-Bit Seriennummer und ermöglicht die Überwachung der Batteriespannung durch den integrierten A/D-Wandler. Sein Betriebsbereich liegt zwischen $-40\text{ }^{\circ}\text{C}$ bis zu $85\text{ }^{\circ}\text{C}$ [14]



Abbildung 8 Smart Battery Monitor Chip DS2438

B. iButtons:



Abbildung 9 DS1923 Temperatur und Feuchtigkeit Sensor

Der DS1923 kann Temperatur und/oder Luftfeuchtigkeit erfassen und das Messergebnis in einem gesicherten Speichersystem registrieren. Die Registrierung wird mit einer vom Benutzer festgelegten Rate durchgeführt. Es werden bis zu 8192 Messungen mit 8 Bit oder 4096 Aufnahmen mit 16 Bit in einem Zeitraum von 1 Sekunde bis 273 Stunden abgespeichert. Diese werden später zu einem Interface über das 1-Wire-Bus übertragen [12]. Überdies wird der DS1923 selten benutzt, da er im Vergleich zu den anderen Sensoren zu kostenaufwendig ist. Laut der Website des Distributors Digikey liegt der Preis gegenwärtig bei 89€ pro Stück. [15]

C. Komplette 1-Wire-Module:

Diese werden aus vielen elektronischen Komponenten hergestellt, um sie als eigene Module/Sensoren bzw. Produkte zu benutzen. Die Abbildung 10 zeigt ein Beispiel eines DS18B20 Temperatursensors mit Edelstahl-Design. [16]



Abbildung 10 DS18B20 mit Edelstahl-Design

2.1.4 Arduino

Seit vielen Jahren ist Arduino das Gehirn von Tausenden von Projekten, Alltagsgegenständen bis hin zu komplexen wissenschaftlichen Instrumenten. Eine weltweite Gemeinschaft von Hobbyisten, Studenten, Künstlern, Programmierern und Profis hat sich um diese Open-Source-Plattform versammelt, deren Beiträge sich zu einer unglaublichen Menge an zugänglichem Wissen addiert haben, das Anfängern wie Experten gleichermaßen eine große Hilfe sein kann. [17]

Das Herzstück auf dem Arduino-Board (s. Abbildung 11) ist ein integrierter Mikrocontroller. Diesen kann man sich als kleinen Computer auf einem Chip vorstellen. [18]

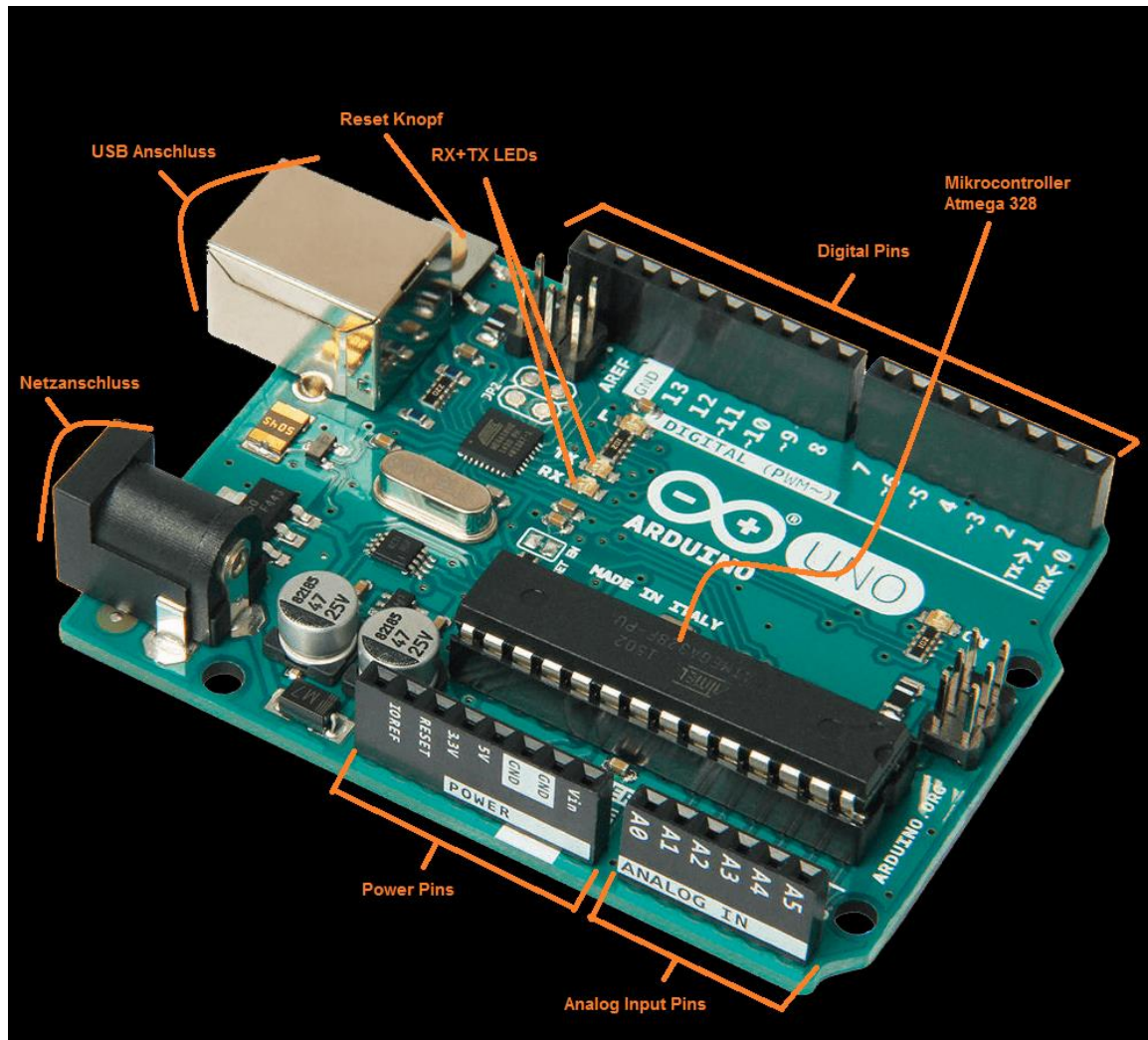



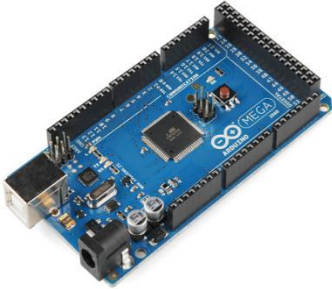
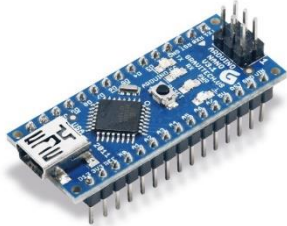
Abbildung 11 Arduino UNO R3 – Eigene Darstellung nach

Der Mikrocontroller stammt von der Firma Atmel (atmega328) und der Chip ist als AVR¹ bekannt. Er ist in modernen Begriffen langsam, läuft mit lediglich 16Mhz, mit einem 8-Bit-Kern und verfügt mit 32 Kilobytes Speicher und 2 Kilobytes Arbeitsspeicher über eine sehr begrenzte Menge an vorhandenem Speicher. Das Arduino Uno Interface-Board ist insbesondere für sein eigenartiges Design bekannt. [18]

Des Weiteren gibt es zwei weitere Arduino Boards, die ebenso berücksichtigt werden können. In der nachfolgenden Tabelle werden die drei Arduino Boards, die für das vorliegende Projekt in Frage kommen, einander gegenübergestellt.

¹ eine 8-Bit-Mikrocontroller-Familie des US-amerikanischen Herstellers Microchip [Wikipedia]

Tabelle 2 Vergleichen von Arduino Produkte

	Arduino Uno R3	Arduino Uno mega 2560	Arduino Nano
			
Preise	20 € - 22 €	36€ - 39€	19€-24€
Maße L x B	68,58 x 53,34 mm	101,6 x 53,34 mm	17,78 x 48,26 mm
Clock speed	16MHz	16MHz	16MHz
Speicherplatz	32KB	256KB	32KB
Arbeitsspeicher	2KB	8KB	2,5KB
Digital I/O mit PWM Pins	6	15	7
Kompatibilität mit Ethernet Shield	JA	JA	NEIN

Bei dem Arduino Uno R3 handelt es sich um die richtige Wahl für dieses Projekt, da für die Kommunikation mit dem 1-Wire-Bus nur ein einziger digitaler Pin notwendig ist und für die TCP/IP-Kommunikation ein Ethernet Shield hinzugefügt werden soll. Darüber hinaus stellt der Arduino Uno R3 die kostengünstigste Option dar. [19]

2.1.5 Ethernet Shield 2 W550

Der W5500-Chip ist ein Hardwired TCP/IP-Embedded-Ethernet-Controller, der eine einfachere Internetverbindung für Embedded-Systeme bereitstellt. Der W5500 befähigt Anwender, die Internet-Konnektivität in ihren Anwendungen zu nutzen, indem er nur einen einzigen Chip verwendet. [20]

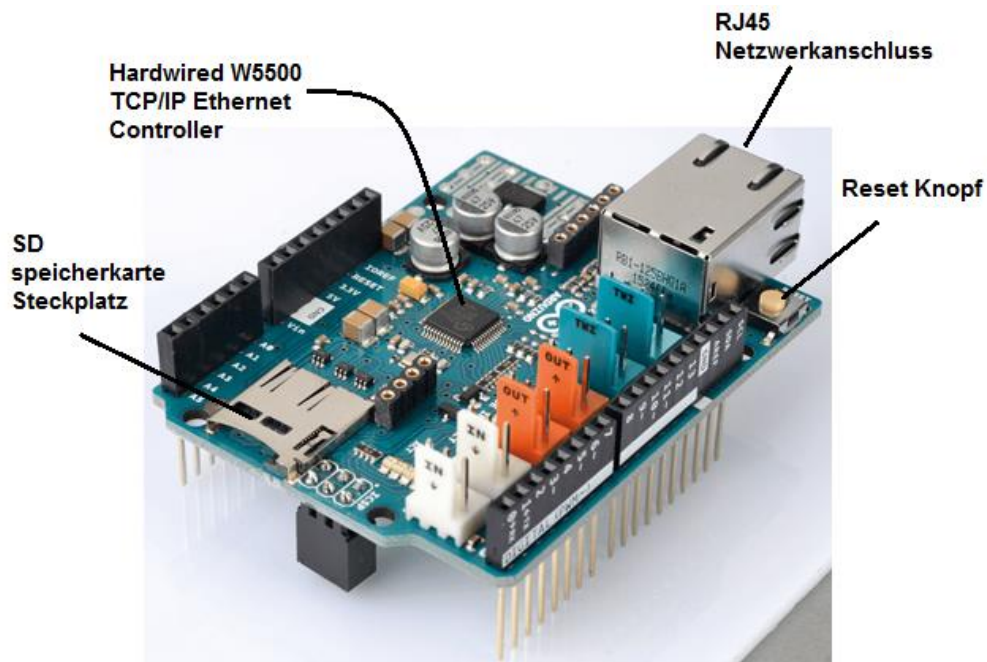


Abbildung 12 Ethernet Shield W5500 Eigene Darstellung nach

Das Ethernet Shield W5500 hat keinen Netzanschluss. Es wird unmittelbar auf das Arduino-Board montiert und durch den Arduino mit Spannung versorgt. Auf das Shield können wiederum andere Shields gestapelt werden, es muss lediglich mit dem Design aller Module kompatibel sein.

Ferner wird es durch ein Netzkabel (RJ45) mit einem Router oder mit einem PC verbunden. Auf diese Weise ermöglicht das Shield die Internetverbindung auf das Arduino und können ein Webserver erzeugt sowie die IP-Adressen auf den Servern eingestellt werden. Überdies verfügt das Shield über einen Steckplatz für eine Mikro-Speicher-SD-Karte für das Lesen und Schreiben von Daten.

Ein Beispiel für das Auslesen von zwei 1-Wire-Temperatursensoren DS18B20 wird in der Abbildung 13 dargestellt. Wie auf dem Bild erkennbar, ist das Ethernet Shield mit meinem Router von zuhause mit einem 10 m langen Netzkabel verbunden. Gleichzeitig sind die Sensoren über ein Datenkabel an das Arduino-Board angeschlossen (Digital Pin 2).

Ein Sensor ist mit der 1-Wire-Batterie-Monitor-Leiterplatte nach der Potentialtrennung-Schaltung angeschlossen. Die Ergebnisse können auf jedem Gerät, das über eine Internetverbindung verfügt, gezeigt werden. Im vorliegenden Beispiel habe ich mit dem Handy die erzeugte IP-Adresse des Ethernet Shield gegeben. Das Ergebnis ist der nachfolgenden Abbildung zu entnehmen.

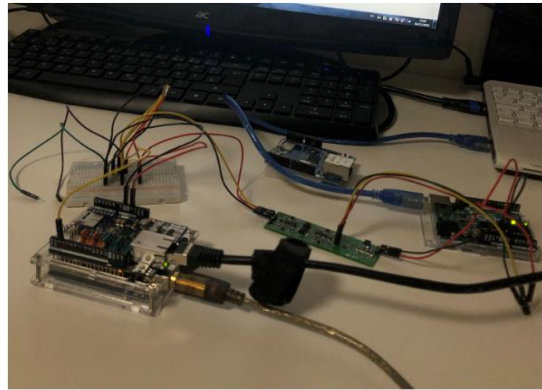
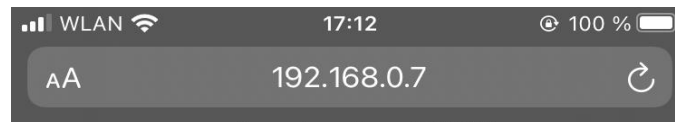


Abbildung 13 Auslesen von zwei DS18B20 Temperatur Sensoren durch mit einem Webserver durch Ethernet Shield W5500 – Eigenes Bild

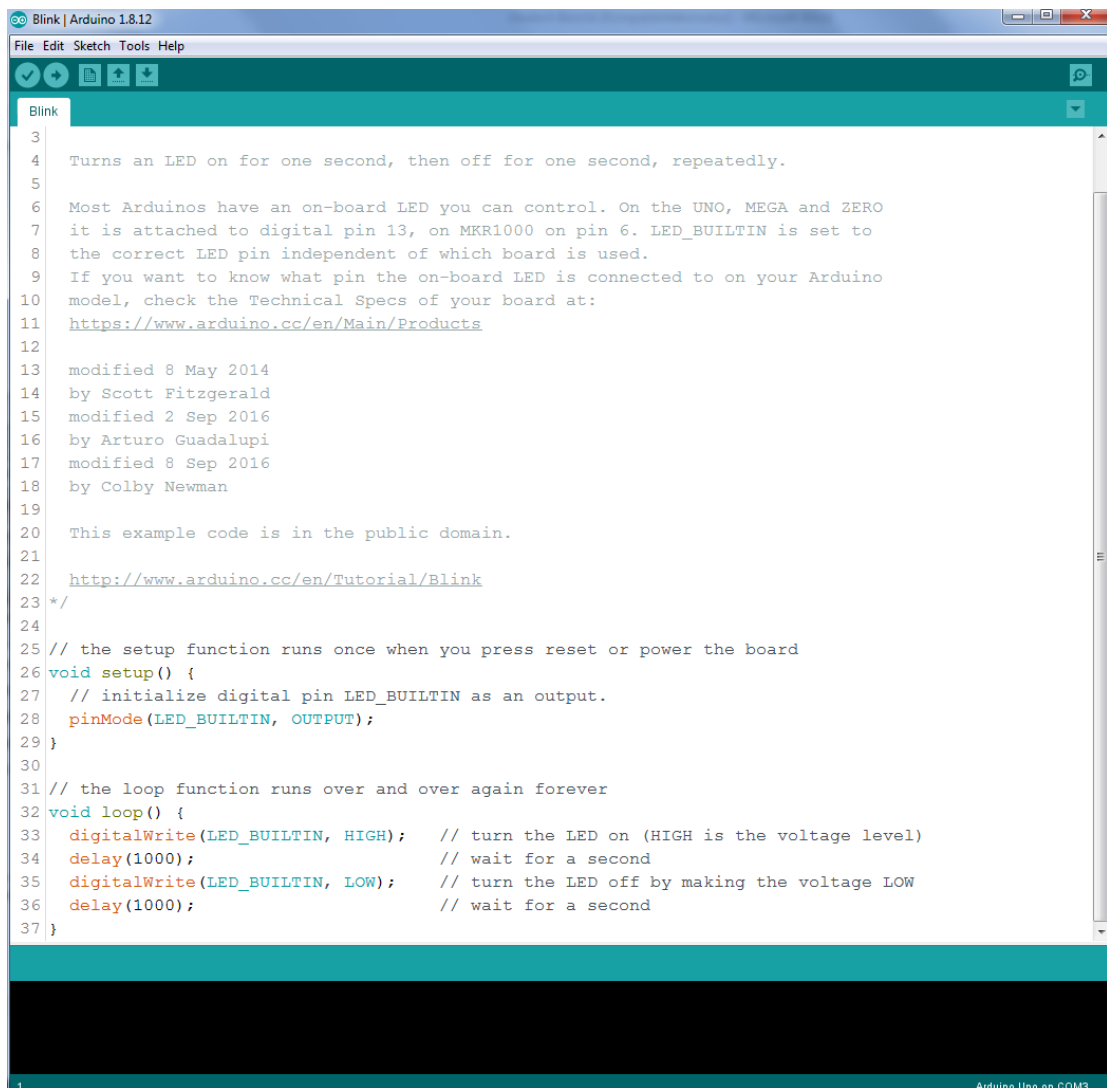
2.2 Software

2.2.1 Programmierung

Die verwendete Programmiersprache in der Entwicklungsumgebung des Arduino basiert grundsätzlich auf C und C++. Da die Arduino-Plattform eine „Open-Source“ ist, hat jeder die Möglichkeit, auf diese Plattform zugreifen zu können. Mit der Hilfe von Bibliotheken und Funktionen, die seitens zahlreicher Firmen oder Programmierexperten geschrieben sind, ist die Programmierung in der Arduino-Plattform einfacher nachvollziehbar. Darüber hinaus können die Benutzer in dieser Entwicklungsumgebung eigenständig den Code umschreiben und bearbeiten [21]. Ein weiterer Vorteil dieser Plattform besteht darin, dass sie eine Vielzahl vom Arduino-Team entwickelter Standardbibliotheken enthält, die einen einfachen und gezielten Aufbau von Funktionen zur Verfügung stellen, um die Programmierung in dieser Plattform so einfach wie möglich zu halten. Die Bibliotheken sowie Beispiele (Sketchs) können jederzeit aufgerufen bzw. auf der Plattform neu instal-

liert werden. Hierfür bedarf es lediglich der Open-Source Arduino Software (IDE). In dieser Software können beliebige Standard-Codes bzw. Beispiele aus Arduino-Sketchs selektiert sowie kompiliert werden, bevor der Code anschließend unmittelbar auf das Arduino-Board hochgeladen wird.

Die Arduino-Programmiersprache lässt sich in drei Hauptteile unterteilen: Funktionen, Werte (Variablen und Konstanten) und Struktur. Ein einfaches Beispiel für ein LED, das in einem Delay von 1 Sekunde ein und aus leuchtet, ist in der Abbildung 14 dargestellt.



```

3
4 Turns an LED on for one second, then off for one second, repeatedly.
5
6 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8 the correct LED pin independent of which board is used.
9 If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }

```

Abbildung 14 Beispiel eines Programms in Arduino IDE – Eigenes Bild

2.2.2 LabVIEW

LabVIEW hat das Arbeitsleben von Tausenden von Wissenschaftlern, Ingenieuren und Technikern vereinfacht sowie deren Produktivität gesteigert. Die Automatisierung hat die Kosten gesenkt und den Produktionsausstoß von Fabriken weltweit erhöht. Die Zykluszeiten für die Produktentwicklung wurden verkürzt und die Qualität vieler Produkte hat sich stetig verbessert. Wenngleich LabVIEW nicht für alle diese Verbesserungen verantwortlich gemacht wird, hat es ohne Frage in vielen Unternehmen eine wertvolle Rolle bei der Erreichung dieser Ziele gespielt. [22]

LabVIEW-Programme werden als Virtual Instruments (VIs) bezeichnet, da ihr Erscheinungsbild und ihre Arbeitsweise eine Simulation der Eigenschaften technischer Geräte, wie z. B. Oszilloskope und Multimeter, darstellen. Jedes VI enthält zahlreiche Funktionalitäten, um Eingaben von der Bedienoberfläche zu behandeln oder andere Datenquellen zu beeinflussen, diese darzustellen oder sie in andere Dateien oder auf einen anderen Rechner zu laden. [23]

Ein VI hat die folgenden zwei wichtigen Elemente:

- **Frontpanel:**

Stellt die Bedienoberfläche dar: Wenn ein neues oder ein existierendes VI-Projekt aufgerufen wird, wird das Frontpanel-Fenster des VIs als grafische Benutzeroberfläche (GUI) eines VIs eingeblendet. Der Quellcode, mit dem das Frontpanel ausgeführt wird, ist im Blockdiagramm zu finden. Das Frontpanel-Fenster enthält eine Symbolleiste am oberen Rand und eine Steuerelementpalette, auf die zugegriffen werden kann, indem mit der rechten Maustaste auf eine beliebige Stelle des Frontpanels geklickt wird. [23]

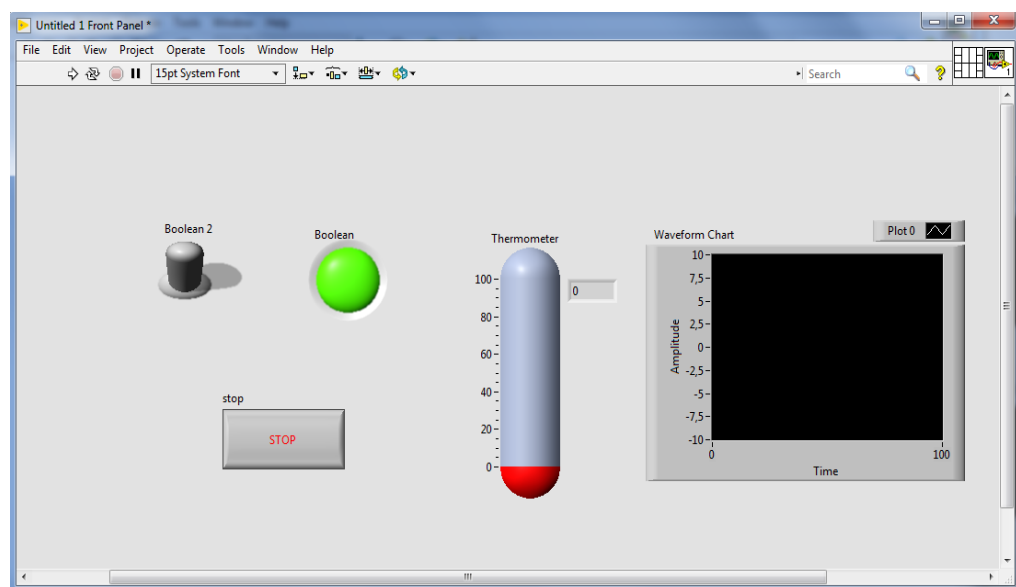


Abbildung 15 Front Panel – Eigene Darstellung

- **Blockdiagramm:**

Das Blockdiagramm besteht aus dem Quellprogramm, das die einzelnen Elemente des VIs definiert. Das Blockdiagramm beschreibt den grafischen Programmcode eines LabVIEW-Programms. Das Prinzip des Blockdiagramms liegt darin, den grafischen Quellcode auf eine logische und simple Art und Weise von der Bedienoberfläche zu

scheiden. Bedienfeldobjekte treten im Blockdiagramm als Anschlüsse auf[23].

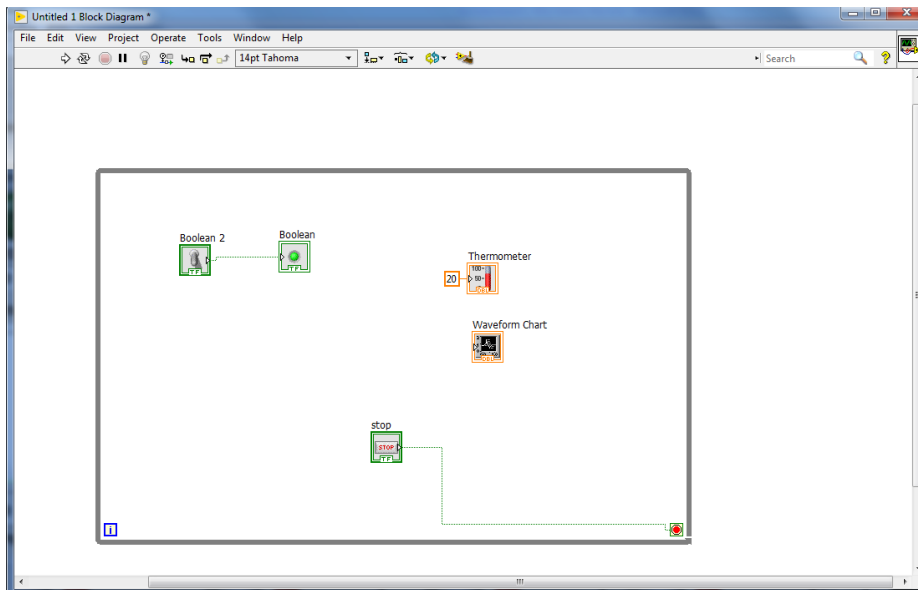


Abbildung 16 Block Diagramm Eigene Darstellung

Außerdem stehen Terminals bzw. die Klemmen für den Datentyp des Steuerelements oder des Indikators. Man kann Steuerelemente oder Indikatoren auf der Frontplatte so konfigurieren, dass sie im Blockdiagramm als Symbol- oder Datentyp-Terminals erscheinen. Standardmäßig werden Objekte auf der Frontplatte als Icon-Terminals angezeigt. Klemmen sind Eingabe- und Ausgabe-Ports, die Daten zwischen der Frontplatte und dem Blockdiagramm übertragen. Über die Steueranschlüsse werden die Daten, die man über die Schaltflächen an der Vorderseite des Diagramms einstellt, an das Blockdiagramm übermittelt. [23]

Nachdem man ein VI-Frontpanel und ein Blockdiagramm erstellt hat, erstellt man das Icon oder das Symbol und den Anschlussbereich, so dass man das VI als Sub-VI betrachten darf. Jedes VI besitzt ein Symbol in der obersten rechten Ecke des Frontpanel- und des Blockdiagrammfensters. Ein Icon ist eine zeichnerische Darstellungsform eines VIs. Es können Texte, Bilder oder eine Mischung aus beidem vorkommen. Wenn man ein VI als Sub-VI benutzt, weist das Symbol das Sub-VI im VI-Blockdiagramm aus. Man kann auf das Symbol doppelklicken, um es anzupassen oder zu bearbeiten.

3 1-Wire Battery Monitor Schaltung

3.1 1-Wire-Battery-Monitor mit Potentialtrennung

Die Schaltung zum gesamten Battery-Monitor (Abbildung 17) wurde am Ende 2018 vom Prof. Dr. Eberhardt Waffenschmidt entwickelt und mit der Software Kicad gezeichnet. Sie hat zwei Seiten:

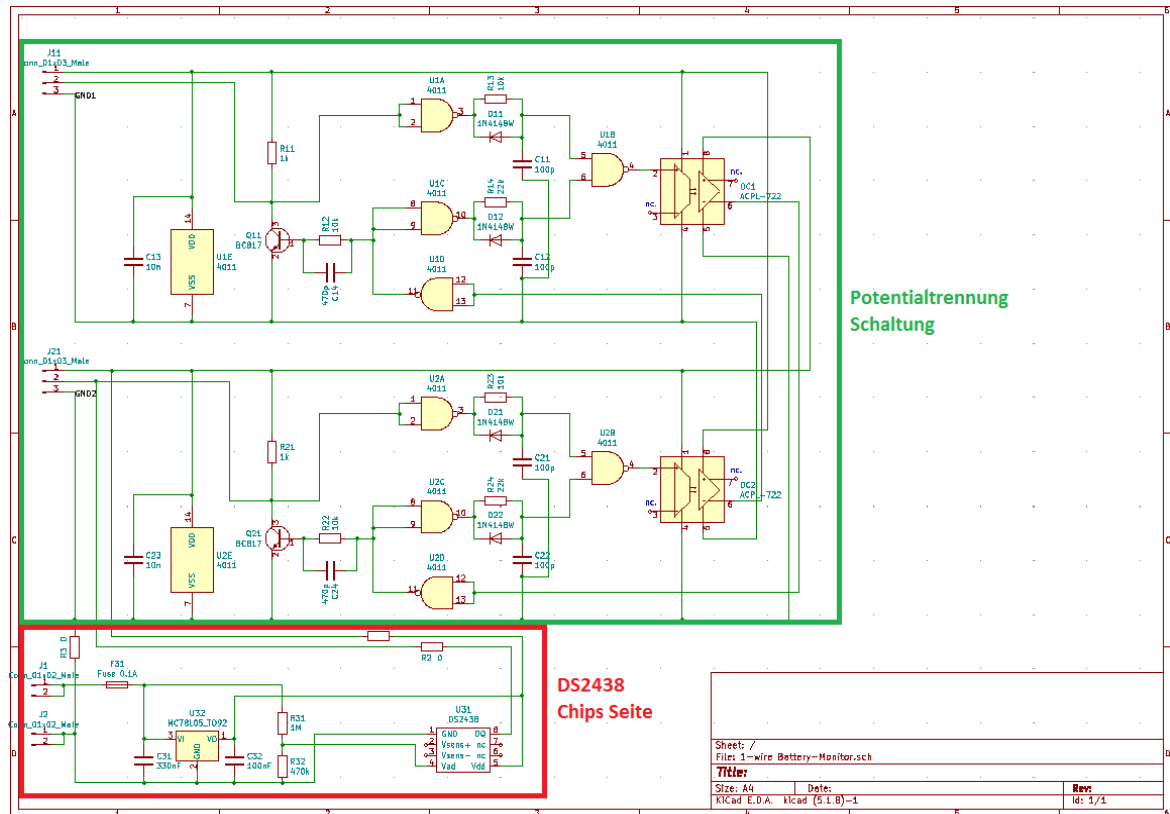


Abbildung 17 1-Wire Battery Monitor Schaltung- Kicad Bezeichnung [24]

- Die obere Seite ist die Potentialtrennung mit 2 Optokopplern. So wird die Interface-Seite mit den 12V Batterien galvanisch getrennt.
- Auf der unteren Seite wurde der Smart-Battery-Monitor-Chip DS2438 integriert und die 5V Versorgungsspannung kommt von der Batterie durch einen Spannungsregler.

Die Signale werden durch den 1-Wire Bus übertragen. Da der Bus potentialgetrennt sein sollte, erfolgt die Schaltung mit 2 Bussen, „Bus A“ und „Bus B“, die über 2 Optokopplern des Typs ACPL 072 gekoppelt sind. Das Übertragen von Signalen gestaltet sich wie folgt: Wird ein Bus auf Low gezogen, wird der Gegner-Bus High und umgekehrt funktioniert es, wenn der erste Bus wieder High wird. [24]

Die NAND-Gatter gelten als Verriegelungs-Regelung mit vierfachen NAND-Gattern in einem konkreten Chip HCF401 und mit jeweils zwei Eingängen. Die RCD-Glieder C11, R13, D11, C12, R14 und D12 sind als gezielte Verzögerungsglieder gebaut und das RC-Glied C14 sowie R12 als Beschleunigungsglied. Als Pull-up-Widerstand wurden die R11

und R21 mit einem Wert von 1 kΩ zwischen der Versorgungsspannung und den Datenbussen addiert.

Die rot markierte Seite, dort, wo der Smart-Battery-Monitor DS2438 integriert wurde, erhält die 5V Versorgungsspannung (VDD) durch den Spannungsregler U32 (MC78L05) von den 12V Bleibatterien. Die zwei Widerstände R31 und R32 gelten als Spannungsteiler für die Spannungsmessungen an den Pins VAD vom DS2438. Die Spannung an den Batterien wird mit der Spannungsteiler-Regel wie folgt berechnet:

$$U_{Batt} = \left(\frac{R_{31}}{R_{32}} + 1 \right) \times U_{VAD}$$

Anschließend sendet der Sensor mit dem Pin 8 DQ die Daten über die potentialfreie Schaltung zu dem Mikrocontroller zurück.

Laut der ersten Dokumentation des 1-Wire Battery-Monitor wurden verschiedene Versionen der Schaltung getestet. Die Version 1.0 galt als finale Version und 3 Stück wurden zu Beginn 2019 als Prototypen fertig gebaut. Das alte Platine-Layout wurde mit der Software Kicad gezeichnet (s. Abbildung 18). [24]

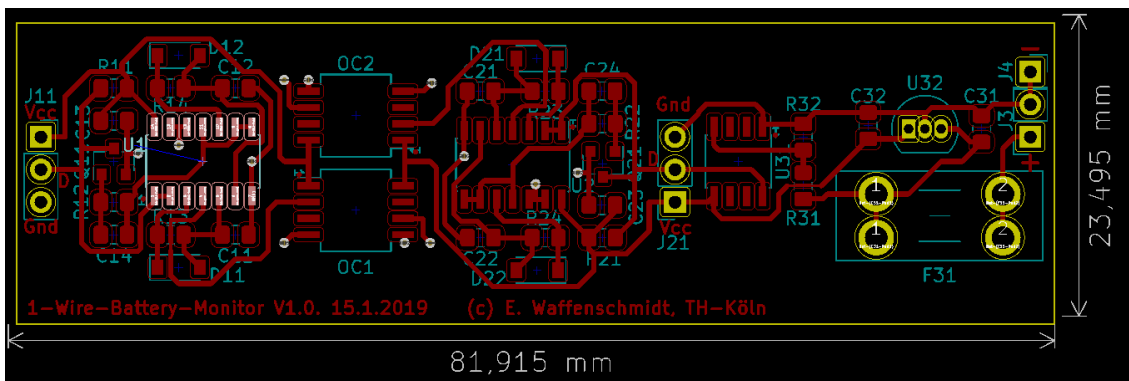


Abbildung 18 Das alte Layout von 1-Wire Battery Monitor Schaltung [24]

Als Nächstes gilt es, das Layout zu editieren, damit die Platine richtig auf die Batterie aufsetzen kann. Um die Temperatur messen zu können, muss der Battery-Monitor-Chip sich auf der Rückseite der Leiterplatte befinden.

3.2 Das neue Layout

Ein neues Layout für den 1-Wire-Battery-Monitor wurde mit der Software Kicad am 12.04 entworfen, wie auf der Abbildung 19 zu sehen ist. Die neue Version der Platine V2.0 wurde im Praxisprojekt neu optimiert und bearbeitet, so dass die Platine gut an die 12V Bleibatterie angepasst ist.

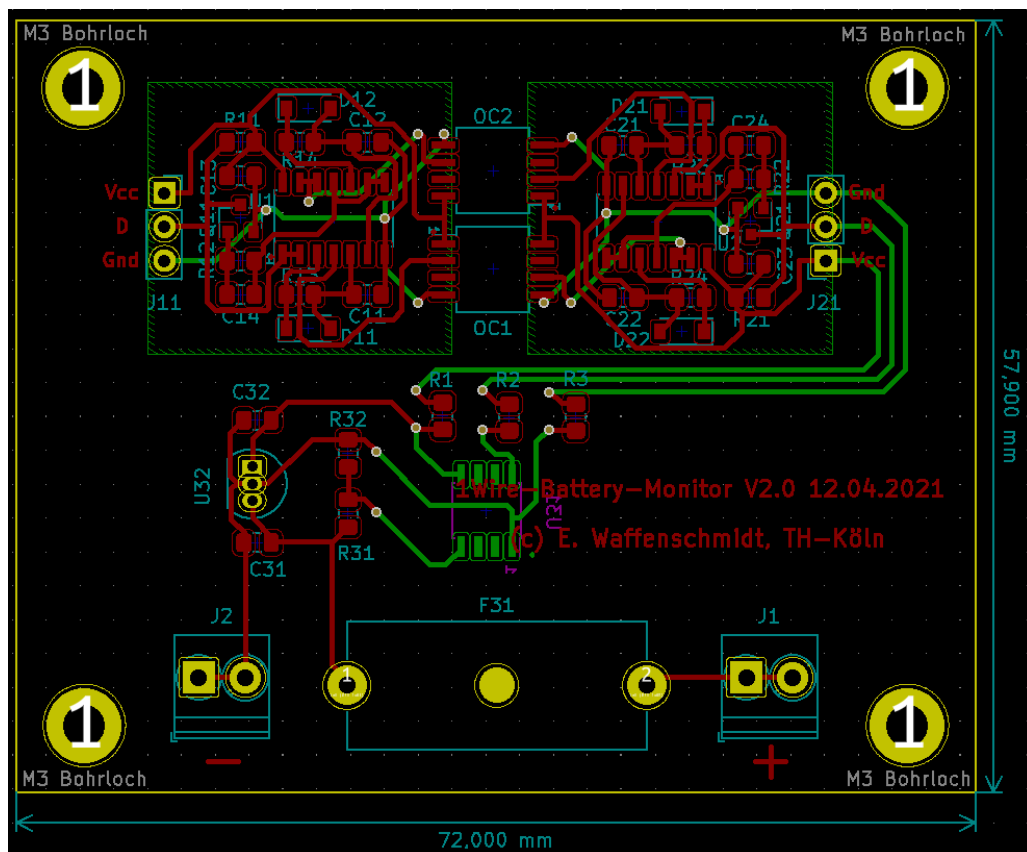


Abbildung 19 Das neues bearbeitete Layout von 1-Wire Battery Monitor – Eigene Darstellung

3.3 Bestellung und Fertigung der Leiterplatten

Am 23.04.21 wurden 25 Leiterplatten mit allen elektronischen Bauteilen zur Verfügung gestellt. Die Bauteile wurden separat bei der Firma RS-Components bestellt, im Anhang ist diesbezüglich eine Excel-Tabelle mit Bestellungsnummern und Beschreibungen befindlich. Da SMD-Widerstände, Kondensatoren und Halbleiter bereits vorhanden waren, wurden diese erst einmal nicht bestellt.

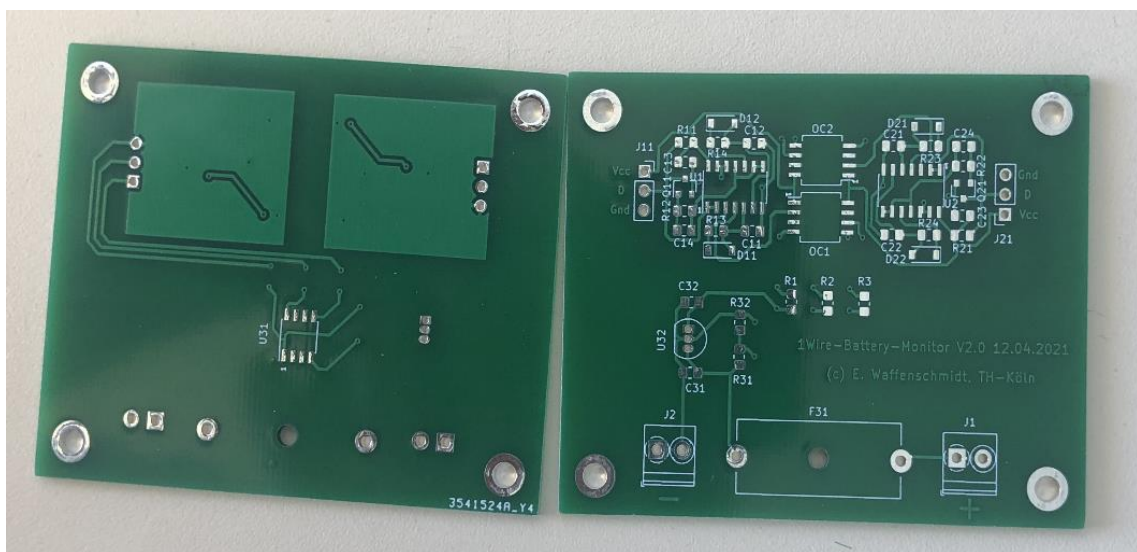


Abbildung 20: Vorderseite und Rückseite der Platine ohne Bauteile – Eigene Bilder

Nach der Bestellung mussten alle Bauteile auf die Platinen gelötet werden. Sieben Stück wurden schon fertig bestückt und in Betrieb genommen. Wie in der Abbildung 21 zu sehen, ist der Battery-Monitor-Chip auf der Rückseite platziert, sodass er die Temperatur direkt von der Oberfläche der Bleibatterie fühlen kann.

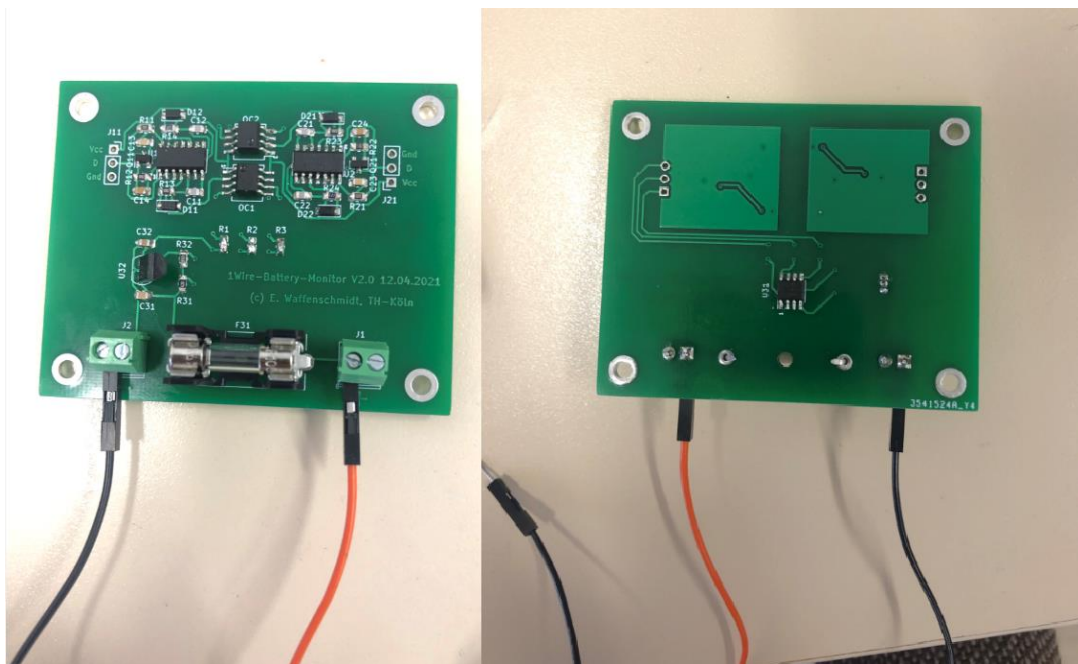


Abbildung 21 gefertigte 1-Wire Battery Monitor Platine

In dem kommenden Kapitel werden der Algorithmus und die Methode für die Inbetriebnahme des 1-Wire-Battery-Monitors, das Auslesen von Temperatur und Spannung an den Batterien sowie das Aufbau und eine Benutzeroberfläche und Ansteuerung für die Visualisierung dargelegt.

4 Algorithmus und Methode für die Inbetriebnahme

4.1 1-Wire Master-Slave-Kommunikation

Für eine Master-Slave-Kommunikation hat die Firma MAXIM/DALLAS ein Übertragungsschema ermöglicht:

- Ein **Reset-Impuls** wird vom Master via Datenbus geschickt, woraufhin alle angeschlossenen 1-Wire Geräte (Slaves), die sich in der Regel im Stand-by-Modus aufhalten, reagieren mit einem Anwesenheitsimpuls über den Datenbus.
- Der Master spricht den gewünschten Slave an "**Device-Selection-Command**", indem jeder Slave seine eigenständige Fabrikationsnummer oder sogenannte "ROM-Adresse" trägt. Ist der gewünschte Slave auf diese Art selektiert worden, kann der Master das aktuelle Kommando an den Slave schicken "**Device-Function-Command**".
- Der Slave antwortet, er setzt den übermittelten Befehl um und liefert die vom Master verlangten Daten, wobei der Master auch die Datenübertragung beherrscht.[12]

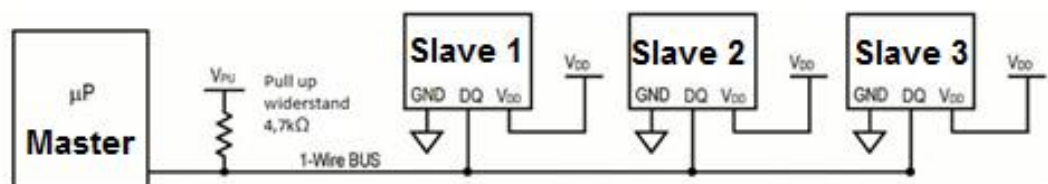


Abbildung 22 Drei Slaves im parallel geschaltet und mit einem Mikrocontroller durch One Wire Bus verbunden – Eigene Darstellung

5 Messung und Auswertung

Im Vorfeld wurden mehrere Tests mit dem Temperatursensor DS18B20 und dem Smart-Battery-Monitor DS2438 durchgeführt und die Daten mit dem Arduino Uno ausgelesen. Mithilfe der 1-Wire-Bibliothek der Firma DALLAS SEMICONDUCTOR wurde der Code in der Programmiersprache C geschrieben und die Werte können auf dem Arduino IDE Serial Monitor angezeigt werden.

Nach der Fertigung der neuen Leiterplatten wurde jede Platine einzeln getestet und ausgelesen. Die Schaltung muss ein sauberes Signal von Bus A zu Bus B übertragen. Dies muss jedes Mal geprüft werden, bevor mit der Messung begonnen wird. Wie in der Abbildung 23 zu sehen ist, wurde ein Rechtecksignal vom Arduino zu dem 1-Wire-Battery-Monitor geschickt und von der DS2438-Seite das Signal im Oszilloskop gezeigt.



Abbildung 23 Prüftest von der 1-WBM mit einem Rechtecksignal – Eigene Bilder

Der Code für den Rechteckgenerator ist im Anhang befindlich.

5.1 Auslesen der Temperatur und Spannung mit dem Arduino Uno

Das Auslesen von 1-Wire-Sensoren kann anhand verschiedener Interfaces durchgeführt werden. Im vorliegenden Beispiel wurde der 1-Wire-Battery-Monitor mit dem Arduino Uno R3 ausgelesen und die Werte von Temperatur und Spannung können im Serial Monitor angezeigt werden. Überdies ist es wichtig, dass das Programm ebenfalls die ID-Nummer bzw. die ROM-Adressen von den Sensoren sucht und zeigt, denn es ist im Rahmen der Durchführung der Messungen an den Batterien notwendig, dass jede Batterie mit einer ID-Nummer definiert werden kann.

Der 1-Wire-Battery-Monitor wurde zunächst mit einer 9V Batterie (gemessen 7~8V) getestet. Der Arduino ist direkt mit dem Eingang von der Platine verbunden und liefert 5V Versorgungsspannung. Der Battery-Monitor-Chip DS2438 nimmt die 5V Versorgungsspannung von der Batterie durch den Spannungsregler, welcher die Spannung wiederum auf 5V reduziert. Der 1-Wire Bus ist an den digitalen Pin 2 von Arduino angeschlossen, infolgedessen die Daten dem Mikrocontroller geliefert werden.

Abbildung 24 und Abbildung 25 zeigen die Verbindung des 1-Wire-Battery-Monitors mit dem Arduino sowie die angezeigten Werte im Serial Monitor vom Arduino IDE. Ein Schaltplan für die Verbindung ist dem Anhang zu entnehmen.

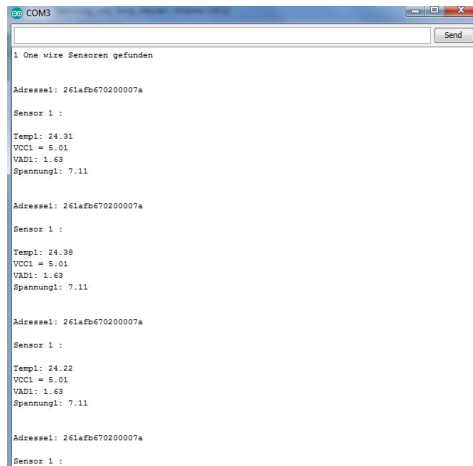


Abbildung 25 Serial Monitor von Arduino

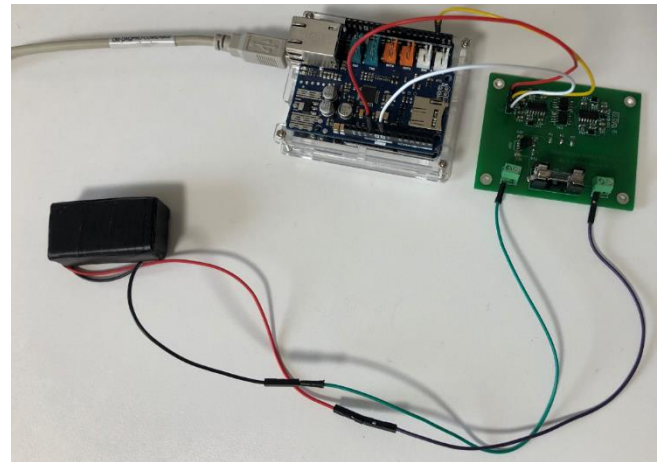


Abbildung 24 Auslesen von einer 1WBM Platine

Die Kommunikation zwischen Mikrocontroller und dem Battery-Monitor-Chip DS2438 ist bidirektional. Der Arduino sendet einen Impuls durch den 1-Wire Bus und der DS2438 als Slave antwortet mit den Daten zurück.

Die Abbildung 26 illustriert die Verbindung zwischen dem Arduino und dem 1-Wire-Battery-Monitor.

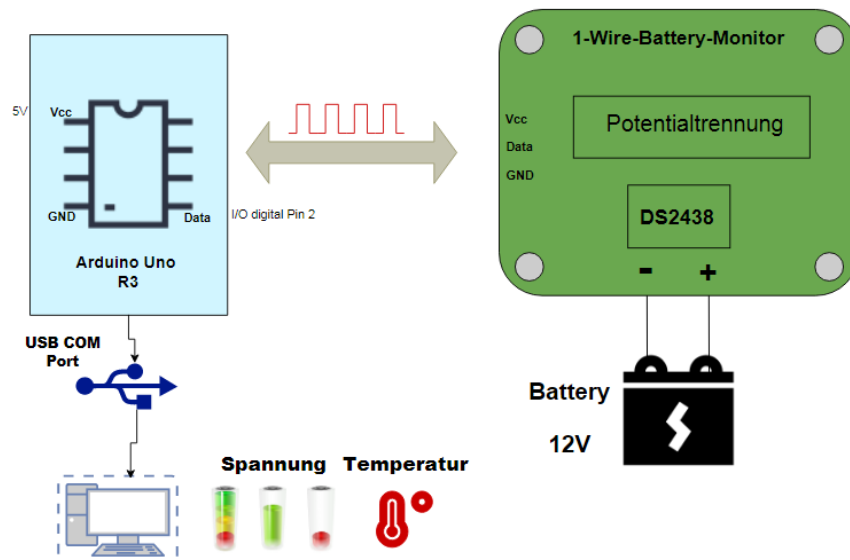


Abbildung 26 Verbindung zwischen Arduino und dem 1-WBM – Eigene Darstellung

5.2 Auswertung der Daten und Diskussion

Wie im vorausgehenden Abschnitt beschrieben wurde, wurde der 1-Wire-Battery-Monitor in Betrieb genommen und mithilfe der Arduino können die Werte von Temperatur und

Spannung an einer Batterie ausgelesen werden. Nach der Messung bedarf es der Prüfung der Messwerte, da es um die Sicherheit der Batterien geht und insoweit darauf zu achten ist, dass die Abweichung nicht so groß ist. In den vorgenommenen Messungen lag die Abweichung ungefähr zwischen -0,8 V und -1,6V. Nach einigen Messungen auch mit einem Netzgerät, das 12V liefert, wurde es im Praxisprojekt abschließend die Werte der Widerstände R31 und R32 verringert. Da der Sensor selbst einen Innenwiderstand von 500 kΩ aufweist (s. Abbildung 27 unten), könnte es sein, dass der Wert des Widerstands den Spannungsteiler verhindert.

DC ELECTRICAL CHARACTERISTICS

(-40°C to +85°C; 2.4V ≤ VDD ≤ 10.0V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Input Logic High	V _{IH}		2.0			V	1
Input Logic Low	V _{IL}		-0.3		0.5	V	1
Shutdown Current	I _{DD1}	DQ=0, RTC Active		25		μA	
Active Current	I _{DD}	DQ=1, ICA Active or Temperature or Voltage Conversions or EEPROM write in progress		50	100	μA	
Input Resistance	R _I	DQ		500		kΩ	2

Abbildung 27 Wert von den Innenwiderstand von DS2438 [14]

Die Widerstände R31 und R32 waren vorher mit den Werten 1MΩ und 470kΩ gebaut. Daraufhin wurden die Werte um den Faktor 100k kleiner gestellt, bis auf die Werte von 330kΩ für R31 und 100kΩ für R32 gelangt wurde. Die Abweichung liegt nun zwischen 0,2 und 0,3 V.

Die nachstehende Excel-Tabelle führt die Schritte auf, denen es gefolgt ist, bis es eine kleine Messabweichung erhalten wird.

Tabelle 3 Spannung Messungen mit verschiedenen Widerstände – Eigene Excel-Tabelle

	R31 in kΩ	1000	680	330	220
	R32 in kΩ	470	330	100	100
	Spannung VAD in V	3,3	3,47	2,73	3,6
Sollwert: 12V	Spannung an der Batterie in V	10,32	10,62	11,74	11,52

6 Versuchsaufbau

Im Folgenden werden das methodische Vorgehen für den Aufbau der Platinen auf den Batterien beschrieben und die mögliche Implementierung vorgestellt.

6.1 Platinen-Design für den Aufbau

Wie in dem vorherigen Abschnitt beschrieben, wurde ein neues Platinen-Layout mit der Software Kicad erstellt, um einen einfachen Aufbau und eine simple Anwendung der Batterien zu ermöglichen. Im Layout wurden vier M3 Bohrlöcher gezeichnet, dort werden Schrauben samt Gewinde fest eingebaut.

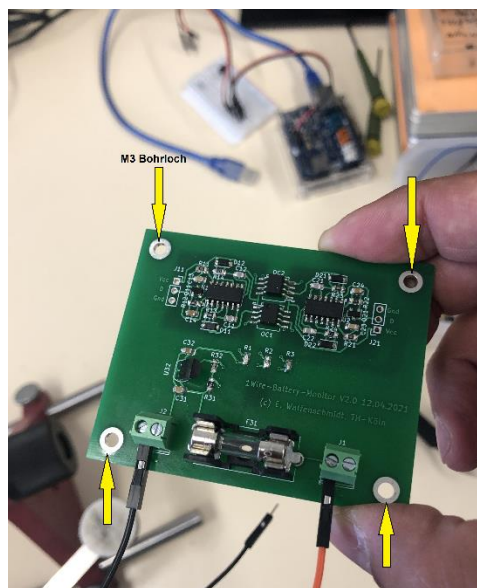


Abbildung 28 Das fertige Produkt von 1-WBM mit M3 Bohrlöcher – Eigenes Bilder

Danach werden alle Platinen, wie in der Abbildung 29 zu sehen, vorbereitet, um diese festzukleben. Eine M3-Schraube mit flachem Kopf wurde benutzt, damit diese einfach auf die Oberfläche der Batterie geklebt werden kann. Des Weiteren wäre es stets optimal, die ID-Nummer der Sensoren mit einem Schreibgerät auszudrucken und auf die Platinen zu kleben.

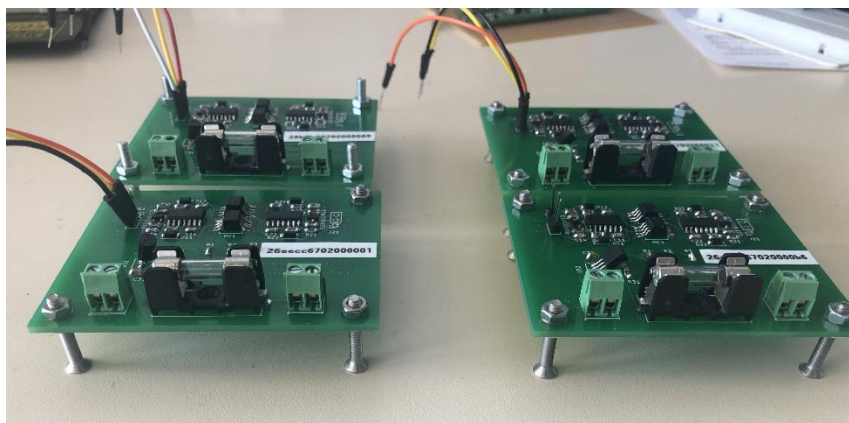


Abbildung 29 Vier Platinen mit einer M3-Schraube und ID-Nummern – Eigenes Bild

6.2 Montage der 1-Wire-Battery-Monitor-Platinen

Nach Vorbereitung der 1-Wire-Battery-Monitor-Platinen müssen diese zunächst auf den Batterien fixiert werden. Mit einem Zwei-Komponentenkleber, bestehend aus unterschiedlichen Komponenten, können die Schrauben mit jeweils flachem Kopf durch die Mischung der beiden Komponenten auf die Oberfläche der Bleibatterien fest montiert werden. Nach der Verwendung von Epoxidharz-Kleber müssen die Bauteile 24 Stunden still gelassen werden, bis die Teile fest kleben, bevor sie zum Durchführen bereitgestellt werden.

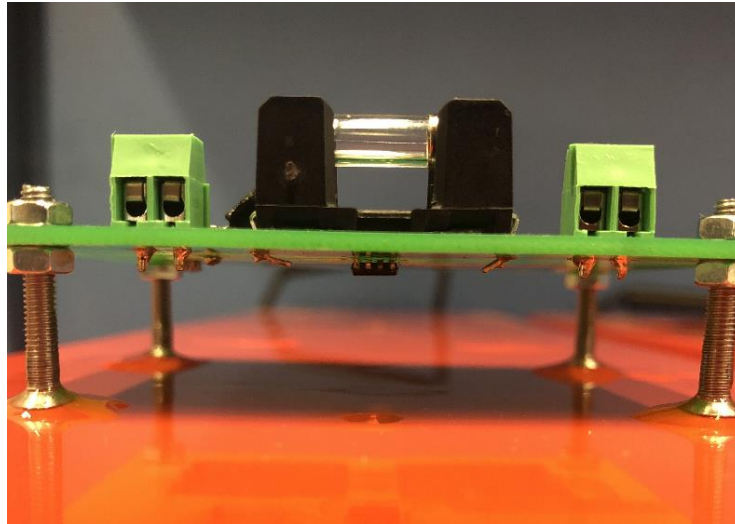


Abbildung 30 Aufbau von der Platine mit zwei Komponentenkleber – Eigenes Bild

Die Abbildung 31 zeigt das abschließende Ergebnis: Die 1-Wire-Battery-Monitor-Platinen wurden auf den Batterien aufgebaut, mit dem Arduino parallel verbunden. Diese sind bereit zur Inbetriebnahme. Ferner muss ein Kupferteil, etwa 2*2cm, mit einer Wärmeleitpaste unter die Sensoren gestellt werden, um eine sichere Wärmeübertragung für die Temperaturmessungen zu gewährleisten.



Abbildung 31 Aufbau von vier 1-WBM – Eigenes Bild

7 Benutzeroberfläche und Einrichten von Arduino

7.1 Eingesetzte Softwareanwendungen

Eine Benutzeroberfläche wurde mit der Software LabVIEW programmiert. Die Methodik des Projekts kann in zwei Teile gegliedert werden: Softwareentwicklung und Hardwareimplementierung. Die Tätigkeiten in der Softwareentwicklung umfassen die Messungen und die Visualisierung der Daten von dem Mikrocontroller durch USB-Schnittstelle oder TCP/IP-Schnittstelle. Überdies unterstützt LabVIEW die Erstellung einer grafischen Benutzeroberfläche, verwendet die automatische Speicherverwaltung. Das Interesse an visueller Programmierung resultiert in der Selektierung von LabVIEW als Software. Ein GUI-basiertes System ermöglicht es dem Benutzer, Überwachungsparameter einzustellen, Werte von Sensoren zu lesen, Daten zu erfassen sowie wichtige Parameter anzuzeigen. Darüber hinaus eignet sich LabVIEW für ein automatisches Bewässerungssystem, da es über mehrere Funktionen verfügt.

7.2 Entwurf und Erstellung der Benutzeroberfläche

Der Entwurf des Blockdiagramms und die Frontplatte werden mit der Software LabVIEW entwickelt. Das Design ist so aufgebaut, dass es die Temperatur und Spannung von den Batterien mithilfe der 1-Wire-Battery-Monitor-Platine durch Arduino ausliest, die maximalen und minimalen Werte kontrolliert sowie die Daten in einer Kurve darstellt.

Es werden zwei Versionen für das Monitoring erstellt, einmal mit der USB-Schnittstelle und einmal mit der TCP/IP-Schnittstelle.

7.2.1 USB Schnittstelle:

Um Daten seriell in LabVIEW zu erhalten, wird ein weiterer Treiber benötigt. Dieser Treiber wird für die serielle Kommunikation mit Arduino und LabVIEW VI verwendet. Ohne die Installation des Treibers können keine Daten in LabVIEW abgerufen werden.

Einrichten :

- Zunächst ist der NI VISA-Treiber herunterzuladen.
- Nach dem Herunterladen gilt es, den Treiber zu installieren.
- Nach der Installation erstellt der NISA-Treiber automatisch ein Tool mit dem Namen des COM-Ports.
- Nach der Installation wird dieses Tool nach erstmaligem Starten von LabVIEW automatisch im LabVIEW-Komponenten-Fenster angezeigt.

Abbildung 32 präsentiert die beiden Fenster Front Panel (Links) und Block Diagram (Rechts) von LabVIEW und es erfolgt die Bearbeitung des vorliegenden Monitoring-Systems für eine Einzelüberwachung. Auf das Front Panel werden dem Benutzer zwei Kurvendiagramme und Datendarstellungen angezeigt, die den Messwert der Spannung und

Temperatur angeben. So kann der Benutzer die Zustände einer Batterie bequem überwachen. Auf der rechten Seite enthält das Blockdiagramm den graphischen Quellcode des Programms. Das Ziel ist es, den Grafik-Quellcode von der Bedienoberfläche auf eine sinnvolle und unkomplizierte Art zu trennen.

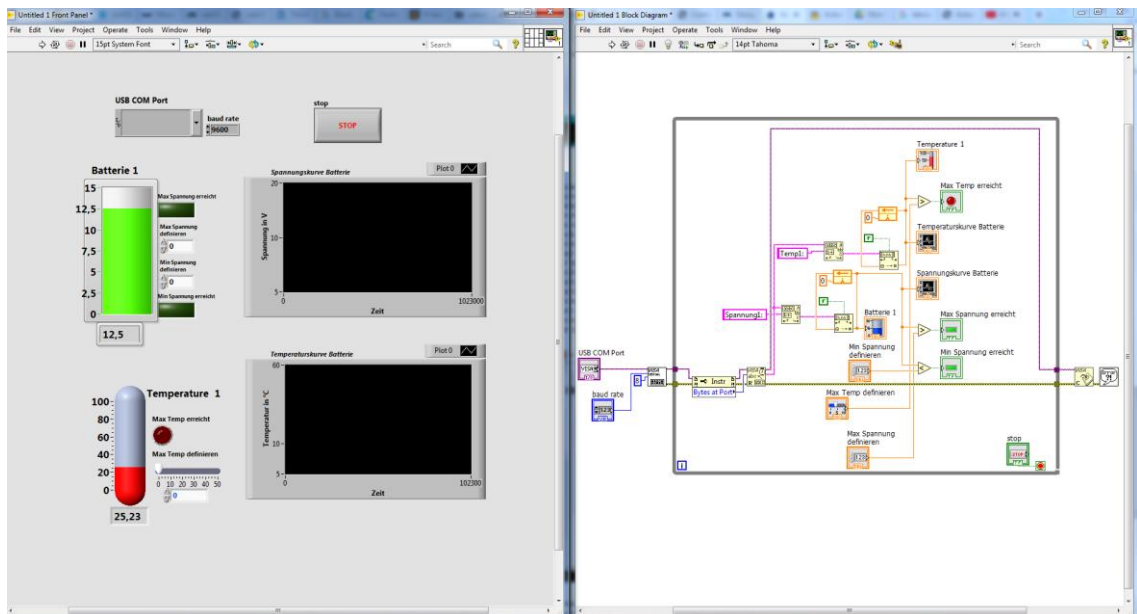


Abbildung 32 Visualisierung einer Einzelüberwachung von 1-WBM mit der USB Schnittstelle

Das Blockdiagramm setzt sich aus verschiedenen Blöcken zusammen, welche die Programmierung der Benutzeroberfläche vereinfachen.

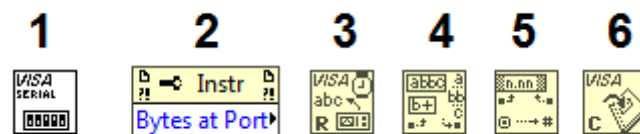


Abbildung 33 Benutzte Blöcke für die Visualisierung

1-> VISA Serial Port: Richtet die serielle Schnittstelle ein.

2-> Bytes at Port: Prüft, ob die Daten verfügbar sind.

3-> VISA Read: Liest die Daten, die an der seriellen Schnittstelle verfügbar sind, von dem verbundenen Gerät.

4-> Match Pattern: Subtrahiert oder addiert ein Zeichen.

5-> Fract/Exp String To Number: Konvertiert eine Zeichenkette in Zahlen.

6-> VISA Close: Schließt die hergestellte Verbindung.

Arduino sendet zwei Datenstrings, die mit einem „Temp1“ und „Spannung1“ verkettet sind. Beim Eintritt in die While-Schleife kommt es zu einer Wartezeit von 5 Sekunden, um die Messwerte zu kontrollieren. Nach der Lese-Funktion werden die Strings mit dem Match-Pattern verglichen, um jeden Daten-String zu „filtern“. Daraufhin werden sie in

Zahlen (double) umgewandelt und mit den Thermometern für die Temperaturmessung sowie mit dem Tank für die Spannungsmessung verbunden.

7.2.2 TCP/IP-Schnittstelle

Um zwei Kommunikationsnetze zwischen Teilnehmern einzurichten, muss eine Kommunikationsschnittstelle zwischen ihnen eingerichtet werden, die sowohl Hardware- als auch Softwareteile umfasst. Der Softwareteil implementiert ein Kommunikationsprotokoll, wohingegen der Hardware-Teil der Schnittstelle für die physische Verbindung und den Informationstransfer zwischen den Teilnehmern sorgt. Im TCP/IP-Protokoll ist der Port eine Struktur, die aus abstrakten Datenstrukturen und Puffern besteht.

Die Port-Funktion ist die gleiche wie die Datei I/O-Operationen, der Port kann sowohl gelesen als auch geschrieben werden. Um verschiedene Arten der Kommunikation zwischen dem Host-Prozess zu ermöglichen, ist ein konkurrierender Prozess in der Netzwerkumgebung erforderlich, der die Portnummer und die IP-Adresse bestimmen kann.[25]

Um eine TCP/IP-Kommunikation zu ermöglichen, muss der Arduino als Server und mit dem Internet verbunden sein. Aus diesem Grund wird das Ethernet Shield 2 W5500 benutzt, welches die Internetverbindung auf das Arduino erlaubt und infolgedessen ein Webserver erzeugt werden sowie IP-Adressen auf dem Server eingestellt werden können. LabVIEW wird als Client arbeiten und sorgt dafür, die Messdaten nachzufragen und diese in einer graphischen Darstellung visualisieren zu lassen.

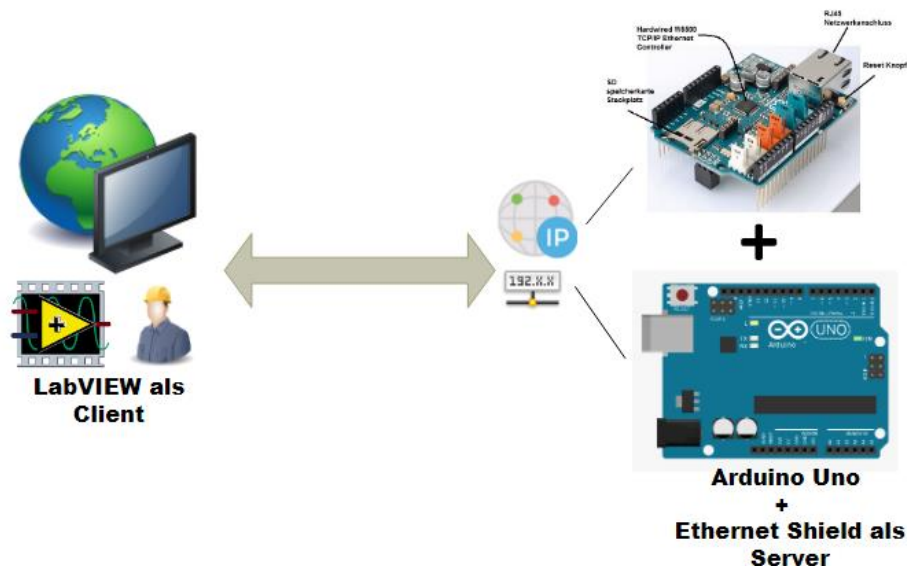


Abbildung 34 Bilddarstellung für die Kommunikation zwischen LabVIEW und Arduino Server – Eigene Darstellung

Einrichten der Hardware:

Zunächst muss eine IP-Adresse erstellt werden, sodass im ersten Schritt eine Verbindung von Arduino und Ethernet Shield hergestellt werden muss. Hierzu wird das Shield einfach auf das Board von Arduino eingesteckt und mit einem Netzkabel mit einem lokalen Netzwerk oder auch einem Router verbunden. Daraufhin ist es betriebsbereit.

Als Erstes gilt es, die IP-Adresse auszulesen und zwar mit dem Befehl *ipconfig/all* in MS-DOS. Die Adresse lautet wie in der Abbildung 35 angezeigt und kann unmittelbar im Arduino Code eingesetzt werden.

```
Windows IP Configuration

Host Name . . . . . :
Primary Dns Suffix . . . . . :
Node Type . . . . . :
IP Routing Enabled. . . . . :
WINS Proxy Enabled. . . . . :
DNS Suffix Search List. . . . . :

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . . :
Description . . . . . :
Physical Address. . . . . : 64-00-6A-02-2F-0E
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::257e:9cae:15eb:1922%7 (Preferred)
IPv4 Address. . . . . : 192.168.1.233 (Preferred)
Subnet Mask . . . . . : 255.255.255.0
```

Abbildung 35 Konfiguration der IP Adresse mit MS-DOS

Nach Konfiguration der IP-Adresse muss der Code auf Arduino hochgeladen werden. Anschließend erscheint die Adresse in dem Serial Monitor von Arduino IDE. Der Code kann mithilfe der Ethernet-Bibliothek geschrieben werden. Zudem ist es notwendig, eine Switch Case zu verwenden, um eine bidirektionale Kommunikation zwischen LabVIEW und Arduino zu ermöglichen.

Einrichten der Software:

Wie es im Zusammenhang mit der seriellen Schnittstelle präsentiert worden ist, wird das Programm in LabVIEW erstellt, um die Daten zugleich durch TCP/IP-Kommunikation auszulesen. Ferner werden neue Blöcke wie TCP Write und TCP Read und eine Switch Case benutzt, indem eine einfache Kommunikation zwischen LabVIEW und Arduino hergestellt wird. LabVIEW muss den Mikrocontroller ansprechen und nach den Daten, deren es für die Visualisierung bedarf, nachfragen.

Abbildung 36 zeigt eine einfache Visualisierung für eine einzelne Überwachung mit der TCP/IP-Schnittstelle des 1-Wire-Battery-Monitors.

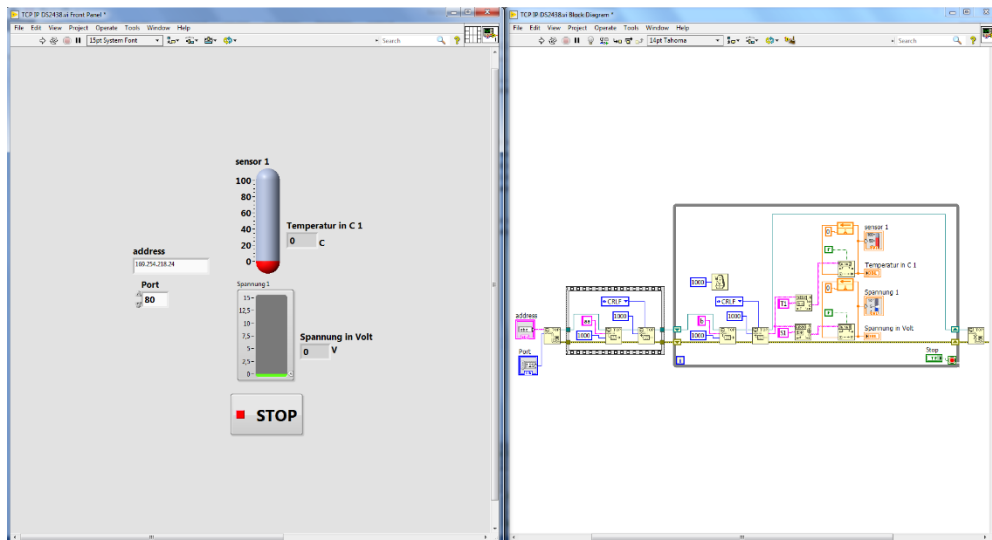


Abbildung 36 Visualisierung einer Einzelüberwachung von 1-WBM mit der TCP/IP Schnittstelle

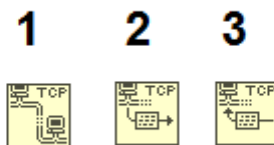


Abbildung 37 Blöcke für die Visualisierung einer 1-WBM

1-> TCP Open Connection: Öffnet eine TCP-Netzwerkverbindung mit der Adresse und dem Remote-Port oder dem Servicenamen.

2-> TCP Write: Schreibt bzw. schickt Daten in einer TCP-Netzwerkverbindung.

3-> TCP Read: Liest die gelieferten Daten von dem Server.

Die IP-Adresse und die Port-Nummer können im Code von Arduino beliebig eingestellt und daraufhin im Front Panel von LabVIEW eingegeben werden. Nach dem Ausführen entsteht eine Verbindung zwischen LabVIEW und dem Server, LabVIEW sendet ein Zeichen zum Arduino, welcher im Wege einer Switch-Case-Funktion mit den entsprechenden Daten von Temperatur und Spannung antwortet.

Fernerhin muss das Programm weiterentwickelt und erweitert werden. Da das Monitoring-System für insgesamt 24 Batterien aufgebaut werden soll, wird für jede der vier Batterien ein Mikrocontroller installiert. Folglich wird das Programm in LabVIEW erweitert, um eine Visualisierung der gesamten vier Batterien zu ermöglichen.

In dem nächsten Kapitel werden die Ergebnisse sowohl mit der USB-Schnittstelle als auch mit der TCP/IP-Schnittstelle dargestellt und diskutiert.

8 Visualisierung und Diskussion

Die Visualisierung mit der USB-Schnittstelle und TCP/IP-Schnittstelle wurde bereits erfolgreich durchgeführt. Die Abbildungen 38 und 39 illustrieren die Ergebnisse der vier 1-Wire-Battery-Monitors, die durch den 1-Wire Bus parallel geschaltet sind.

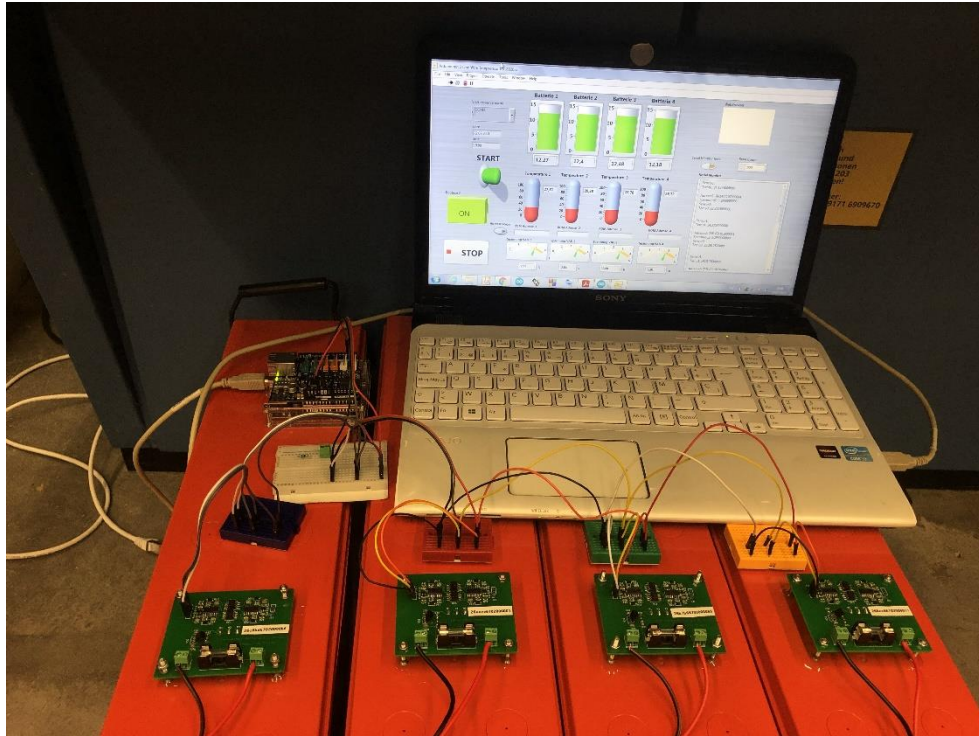


Abbildung 38 Versuchsdurchführung mit vier 1-WBM durch USB Schnittstelle

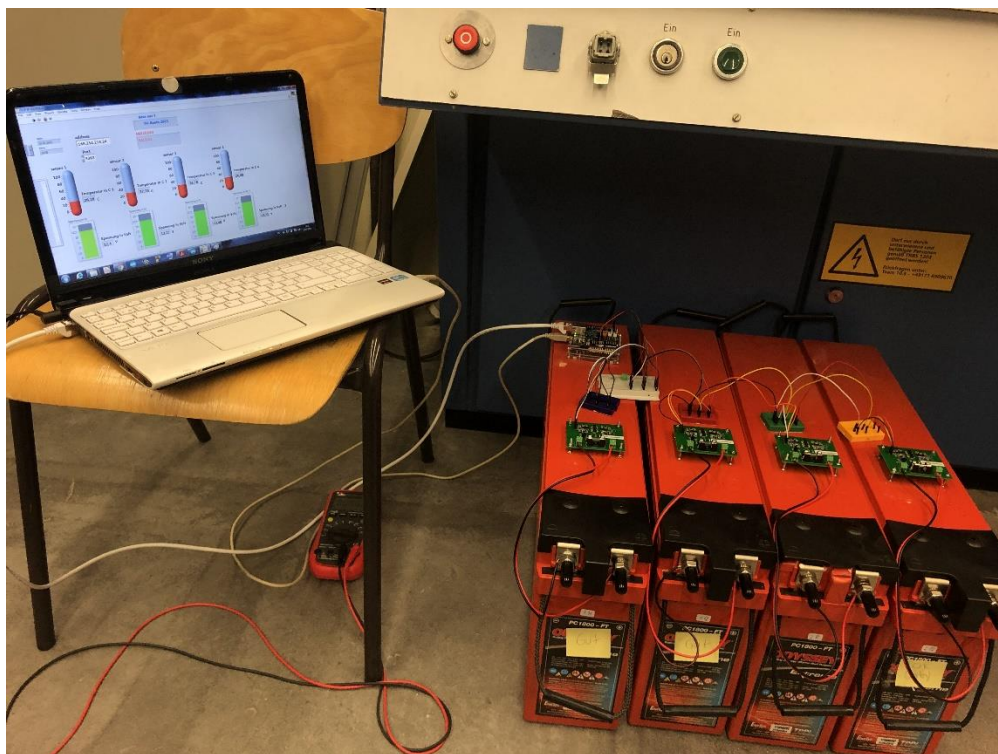


Abbildung 39 Versuchsdurchführung mit vier 1-WBM durch TCP/IP Schnittstelle

8.1 Diskussion

Bei einer TCP/IP-Kommunikation kommen die Daten nicht nacheinander in Reihe und wurden verzögert angezeigt. Dies sollte ein Programmierfehler sein, da der Puffer nicht genügend Speicherplatz erlangt. Des Weiteren kommt es nach ca. 20 min zu einer Unterbrechung der Verbindung zwischen LabVIEW und Arduino Server, infolgedessen keine Daten mehr geliefert werden. Dementsprechend ist eine Visualisierung mit der TCP/IP-Schnittstelle nicht zuverlässig, sodass es einer Erweiterung bedarf.

Im Gegensatz dazu lief die Kommunikation mit der USB-Schnittstelle gut und funktioniert einwandfrei. Die Daten werden in der Reihe und in einem Delay von 5 Sekunden schnell geliefert. Darüber hinaus können die ROM-Adressen von den Sensoren ebenso wie weitere Werte wie Spannung am Eingang des 1-Wire-Battery-Monitor-Sensors zugleich auf der Benutzeroberfläche angezeigt werden lassen. Mit der USB-Kommunikation ist eine anfängliche Einstellung überflüssig, so muss lediglich der Code auf das Arduino hochgeladen und das LabVIEW ausgeführt werden.

Die Abbildung 40 zeigt die letzte erweiterte Version von der Visualisierung, so wird das Monitoring-System am Ende implementiert. Es wurde erst mal nur bei der USB-Kommunikation weiterentwickelt, theoretisch kann dies ebenso bei einer TCP/IP-Kommunikation umgesetzt werden.

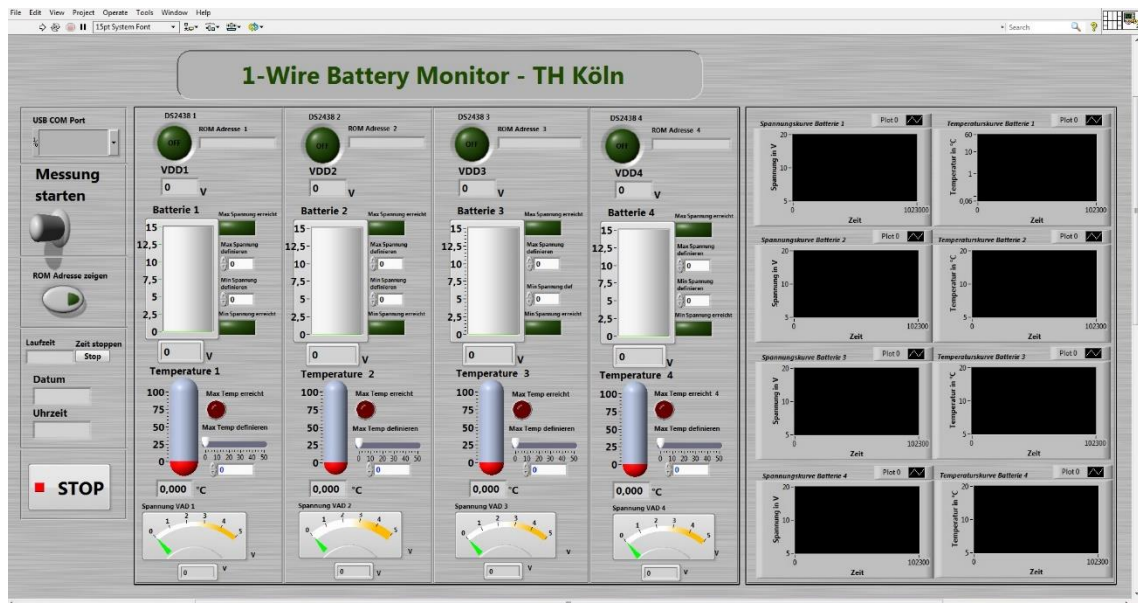


Abbildung 40 Die finale Version von der Visualisierung eines Battery Monitors

9 Fazit und Handlungsempfehlung

Die Batterieüberwachung ist eine einfache, moderne Fernüberwachung, die in Elektroautos oder auf einem Batteriespeicher für PV-Anlagen zum Einsatz kommen kann. So kann der Batteriestatus übersichtlich angezeigt und aktiv überwacht werden.

Im Rahmen dieser Arbeit wurde der 1-Wire-Battery-Monitor realisiert und mit dem 1-Wire-Kommunikationsprotokoll in Betrieb genommen. Ferner wurde der Smart-Battery-Monitor DS2438 verwendet, um die Temperatur und die Spannung an der Batterie zu erfassen und der Arduino UNO als Mikrocontroller eingesetzt, um die Daten vom Sensor auszulesen. Des Weiteren ist die Kommunikation zwischen dem Arduino und dem DS2438 bidirektional und verwendet das Master-Slave-Kommunikationsprotokoll. Das System kann für eine spätere Anwendung an den Batterien implementiert werden.

Im Praxisprojekt habe ich mit der Hardware beschäftigt, die Schaltung wurde optimiert und mit der Software Kicad editiert. Daraufhin wurden die Platinen mit den Batterien verbunden. Es wurden ein neues Layout entworfen und zusätzlich 25 Stück Leiterplatten bestellt. Acht Stück Leiterplatten habe ich danach selber bestückt und getestet, vier Stück funktionierten bereits im Parallelbetrieb und wurden mit einem Netzgerät zum Testen angeschlossen. Des Weiteren musste in der Bachelorarbeit eine Benutzeroberfläche mit LabVIEW programmiert werden, um das Monitoring-System zu visualisieren. Darüber hinaus mussten vier Platinen in Parallelbetrieb in Kombination mit den Batterien installiert werden, um die Spannung und Temperatur messen zu können.

Probleme ergaben sich bei der TCP/IP-Kommunikation und bei der Parallelschaltung von vier Platinen. Folglich wird eine weitere Forschung an dem Battery-Monitor-System empfohlen, sodass die Daten von Arduino durch eine SPI-Kommunikation, in einem Raspberry Pi, ausgelesen und mit Python die Visualisierung anhand einer TCP/IP-Schnittstelle weiter programmiert werden können. Zudem ist zu empfehlen, die Schaltung mit einem Wert von $5\text{k}\Omega$ als Pull-up-Widerstand umzubauen, da bei der Parallelschaltung von vier Platinen lediglich drei angezeigt werden. Hier wurde der Wert des Pull-up-Widerstands von $1\text{k}\Omega$ auf $2,2\text{k}\Omega$ erhöht, damit die Platinen in Parallelbetrieb gebracht werden. Idealweise wäre der Wert vom Datenblatt der DS2438 ($5\text{k}\Omega$) optimal für eine Parallelschaltung.

Für eine weitere Entwicklung sind im Anhang die benutzten Programme und weitere Ergebnisse mit einer Visualisierung im LabVIEW befindlich.

Literaturverzeichnis

- [1] Y. Xing, E. W. M. Ma, K. L. Tsui, and M. Pecht, "Battery Management Systems in Electric And Hybrid Vehicles" *Energies*, vol. 4, no. 11, pp. 1840–1857, 2011.
- [2] S. Yonghua, Y. Yuexi, H. Zechun, "Present Status and Development Trend of Batteries for Electric Vehicles," *Power System Technology*, Vol. 35, No. 4, pp. 1-7, 201.
- [3] Mohammed Amine Jazouli - Eigene Darstellung 2021
- [4] Solariumautonomie- 1-Wire Temperaturmessung abgerufen am August 2021 <https://www.solarautonomie.de/1-wire-temperaturmessung/>
- [5] 1-Wire Implementierung für den AVR abgerufen am August 2021 <https://www.kampis-elektroecke.de/2019/11/1-wire-fuer-den-avr>
- [6] Mohammed Amine Jazouli Eigene Darstellung 2021
- [7] Sashavalli Maniyar Microchip Technology Inc. 1-Wire® Communication with PIC® Microcontroller – MICROCHIP AN1199 pdf
- [8] 1-Wire Algorithmus - abgerufen im August 2021 <https://www.iot-programmer.com/index.php/books/22-raspberry-pi-and-the-iot-in-c/chapters-raspberry-pi-and-the-iot-in-c/39-the-multidrop-1-wire-bus>
- [9] Master of Science Thesis in Medical Engineering Stockholm 2021 SALWAN NAJAR – Simulation of 1-Wire Sensors KTH Technology and Health PDF
- [10] 1-WIRE SEARCH ALGORITHM - APPLICATION NOTE 187 abgerufen im August 2021 <https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/187.html>
- [11] 1-Wire Devices & Products – abgerufen im August 2021 <https://www.maximintegrated.com/en/products/ibutton-one-wire/one-wire.html>
- [12] Bernd vom Berg Peter Groppe Projekt Nr. 275: „Am langen Draht der 1-Wire-Bus“ Grundlagen und Anwendungen des 1-Wire-Busses (Stand: 21.11.2010, V1.3) PDF
- [13] Introduction to DS18B20 A complete step by step tutorial on Introduction to DS18B20 – Stand 2019 abgerufen im August 2021 <https://www.theengineeringprojects.com/2019/01/introduction-to-ds18b20.html>
- [14] DS2438 Smart Battery Monitor – Datenblatt Maximintegrated
- [15] DS1923 Temperatur und Feuchtigkeit Sensor abgerufen im August 2021 <https://www.digikey.de/product-detail/de/maxim-integrated/DS1923-F5/DS1923-F5-ND/2045359>
- [16] Bernd vom Berg Peter Groppe Projekt Nr. 275: „Am langen Draht der 1-Wire-Bus“ Grundlagen und Anwendungen des 1-Wire-Busses (Stand: 21.11.2010, V1.3) PDF Seite 59

- [17] Arduino Introduction – abgerufen im August 2021 <https://www.arduino.cc/en/guide/introduction>
- [18] Buch PDF : Brian Evans - Beginning Arduino Programming Writing Code for the Most Popular Microcontroller Board in the World [Seite 8]
- [19] Artikel: Miguel Gudino abgerufen im Juli 2021 <https://www.arrow.com/en/research-and-events/articles/arduino-uno-vs-mega-vs-micro>
- [20] Datenblatt W5500 Ethernet Shield v1.0 abgerufen im Juli 2021 https://cdn-reichelt.de/documents/datenblatt/A300/103030021_01.pdf
- [21] Bachelorarbeit Daniel Faust RT-BA 2018-14 „Entwurf und Implementierung eines einheitlichen Steuerkonzeptes für einen Sonnen- und Erdalbedosimulator im RACOON-Labor“
- [22] Buch von Rick Bitter, Taqi Mohiuddin und Matt Nawrocki – „LabVIEW Advanced Programming Techniques SECOND EDITION“
- [23] LabVIEW User Manual: abgerufen im August 2021 <https://www.ni.com/pdf/manuals/320999e.pdf>
- [24] Prof. Dr. Eberhardt Waffenschmidt – Doku „1-Wire Sensoren mit Potentialtrennung“
- [25] Artikel: „TCP/IP Remote Communication for Arduino based Motion Control using Virtual Instrumentation „, Department of Computer Science & Engineering, Department of Electronics Engineering G. H. Rasoni College of Engineering, Nagpur, India.

Abkürzungsverzeichnis

1-WBM: 1-Wire Battery Monitor

USV: unterbrechungsfreien Stromversorgungen

ROM: Read Only Memory

VCC: Voltage at the common collector

GND: Ground

VI: Virtual Instrument

GUI: Graphical User Interface

IDE: Integrated Development Environment

USB: Universal Serial Bus

TCP/IP: Transmission Control Protocol/Internet Protocol

Tabellenverzeichnis

Tabelle 1 1-Wire Master und Slave Suchablauf [10].....	5
Tabelle 2 Vergleichen von Arduino Produkte.....	10
Tabelle 3 Spannung Messungen mit verschiedenen Widerstände – Eigene Excel- Tabelle.....	23

Abbildungsverzeichnis

Abbildung 1 Battery Monitoring System – Eigene Darstellung[3]	2
Abbildung 2 Parallelschaltung mit einem Mikrocontroller von drei 1-Wire Sensoren – Eigene Darstellung [6].....	3
Abbildung 3 64-Bit eindeutige ROM-"Registrierungsnummer". [10]	4
Abbildung 4 DS18B20 Temperatur Sensor.....	6
Abbildung 5 DS2438 Smart Battery Monitor Chip	6
Abbildung 6 1-Wire Produkte [12].....	6
Abbildung 7 DS18B20 Pin Darstellung [13]	7
Abbildung 8 Smart Battery Monitor Chip DS2438.....	7
Abbildung 9 DS1923 Temperatur und Feuchtigkeit Sensor	8
Abbildung 10 DS18B20 mit Edelstahl-Design.....	8
Abbildung 11 Arduino UNO R3 – Eigene Darstellung nach	9
Abbildung 12 Ethernet Shield W5500 Eigene Darstellung nach	11
Abbildung 13 Auslesen von zwei DS18B20 Temperatur Sensoren durch mit einem Webserver durch Ethernet Shield W5500 – Eigenes Bild.....	12
Abbildung 14 Beispiel eines Programms in Arduino IDE – Eigenes Bild.....	13
Abbildung 15 Front Panel – Eigene Darstellung	14
Abbildung 16 Block Diagramm Eigene Darstellung.....	15
Abbildung 17 1-Wire Battery Monitor Schaltung- Kicad Bezeichnung [24].....	16
Abbildung 18 Das alte Layout von 1-Wire Battery Monitor Schaltung [24]	17
Abbildung 19 Das neues bearbeitete Layout von 1-Wire Battery Monitor – Eigene Darstellung.....	18
Abbildung 20: Vorderseite und Rückseite der Platine ohne Bauteile – Eigene Bilder	18
Abbildung 21 gefertigte 1-Wire Battery Monitor Platine	19
Abbildung 22 Drei Slaves im parallel geschaltet und mit einem Mikrocontroller durch One Wire Bus verbunden – Eigene Darstellung	20
Abbildung 23 Prüftest von der 1-WBM mit einem Rechtecksignal – Eigene Bilder..	21
Abbildung 24 Auslesen von einer 1WBM Platine.....	22
Abbildung 25 Serial Monitor von Arduino.....	22
Abbildung 26 Verbindung zwishen Arduino und dem 1-WBM – Eigene Darstellung	22
Abbildung 27 Wert von den Innenwiderstand von DS2438 [14]	23

Abbildung 28 Das fertige Produkt von 1-WBM mit M3 Bohrlöcher – Eigenes Bilder	24
Abbildung 29 Vier Platinen mit einer M3-Schraube und ID-Nummern – Eigenes Bild.....	24
Abbildung 30 Aufbau von der Platine mit zwei Komponentenkleber – Eigenes Bild..	25
Abbildung 31Aufbau von vier 1-WBM – Eigenes Bild	25
Abbildung 32 Visualisierung einer Einzelüberwachung von 1-WBM mit der USB Schnittstelle	27
Abbildung 33 Benutzte Blöcke für die Visualisierung	27
Abbildung 34 Bilddarstellung für die Kommunikation zwischen LabVIEW und Arduino Server – Eigene Darstellung	28
Abbildung 35 Konfiguration der IP Adresse mit MS-DOS	29
Abbildung 36 Visualisierung einer Einzelüberwachung von 1-WBM mit der TCP/IP Schnittstelle	30
Abbildung 37 Blöcke für die Visualisierung einer 1-WBM	30
Abbildung 38 Versuchsdurchführung mit vier 1-WBM durch USB Schnittstelle.....	31
Abbildung 39 Versuchsdurchführung mit vier 1-WBM durch TCP/IP Schnittstelle	31
Abbildung 40 Die finale Version von der Visualisierung eines Battery Monitors.....	32

Anhang

Der Code für das Rechtecksignal zum Prüfen.

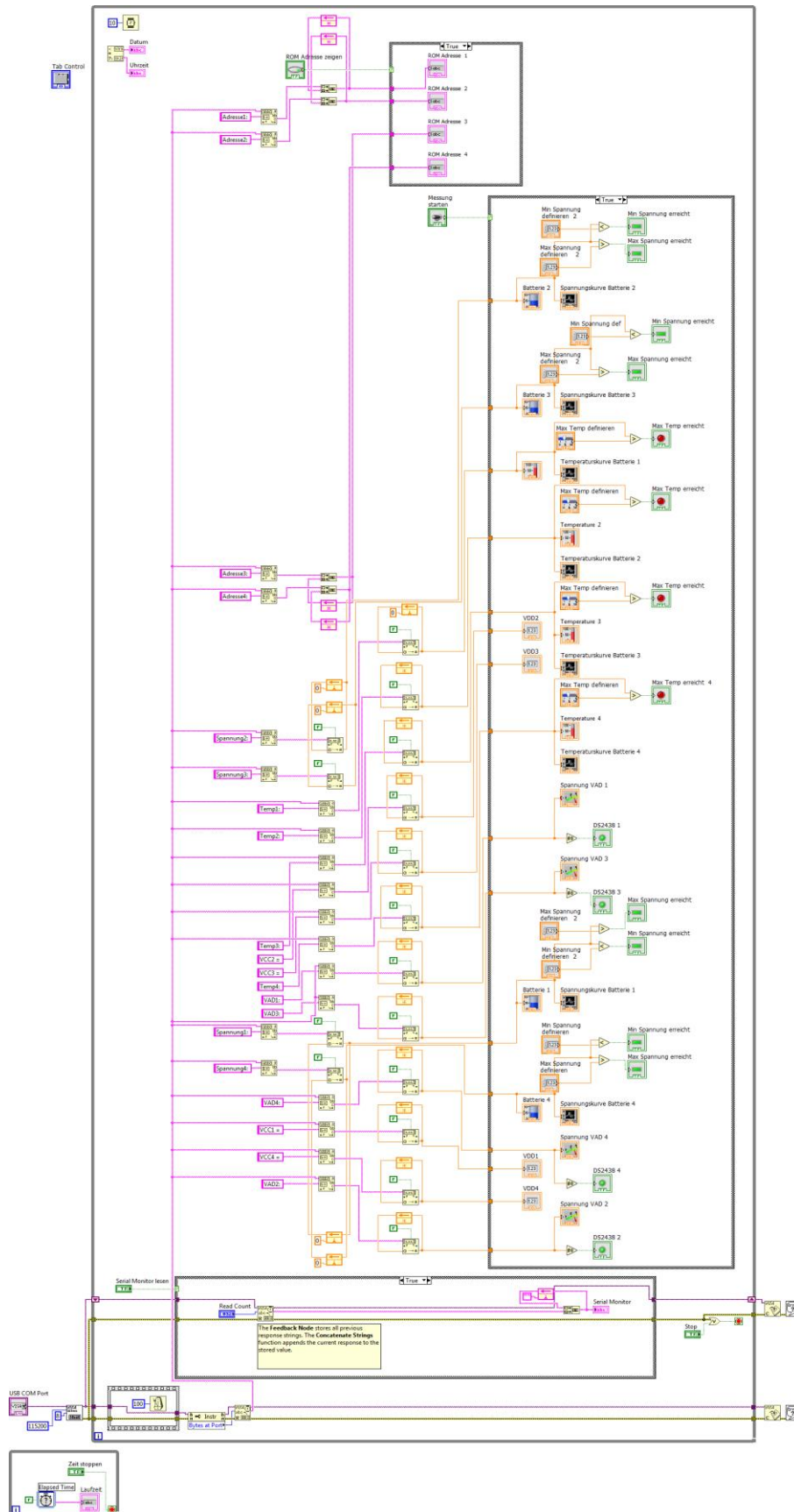


The image shows a screenshot of the Arduino IDE interface. The window title is "Rechteckgenerator_Arduino | Arduino 1.8.12". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for saving, undo, redo, and uploading. The main editor area shows the following code:

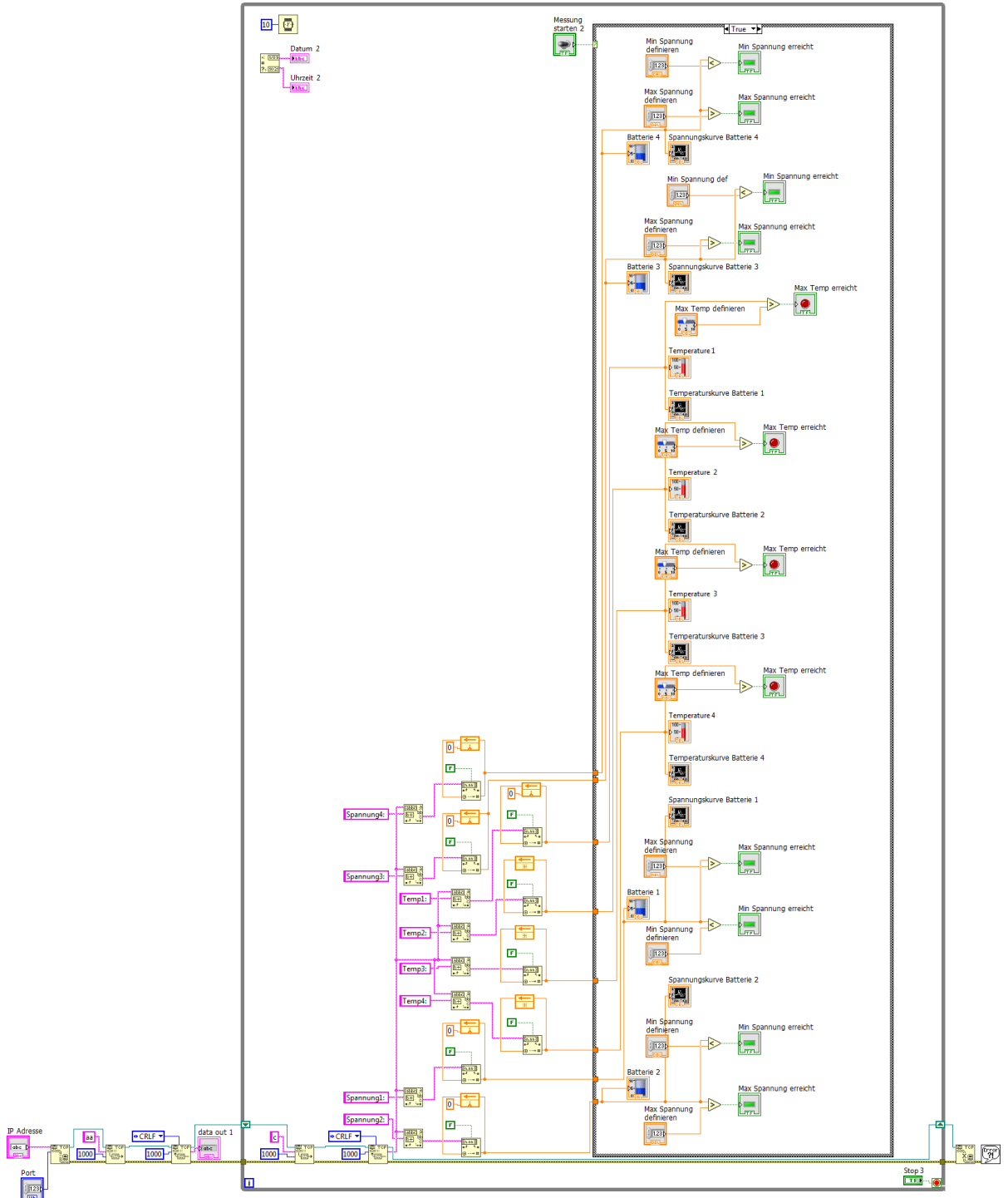
```
1 /* Rechteckgenerator für den Test des 1-Wire Battery Monitors:
2     das Code hochladen dann die Data Pin mit der GND Pin am Arduino
3     anschliessen, danach soll Das Ergebniss auf dem Oszilloskop gezeigt werden.
4 */
5 //Pin vom Arduino definieren
6 const int outPin = 2;
7
8 void setup() {
9
10     pinMode(outPin, OUTPUT);
11
12 }
13
14 void loop() {
15
16     digitalWrite(outPin, !digitalRead(outPin));
17
18     delayMicroseconds(200); //Delay in Mikrosekunde
19
20 }
```

At the bottom of the IDE, the status bar shows "3" on the left and "Arduino Uno on COM3" on the right.

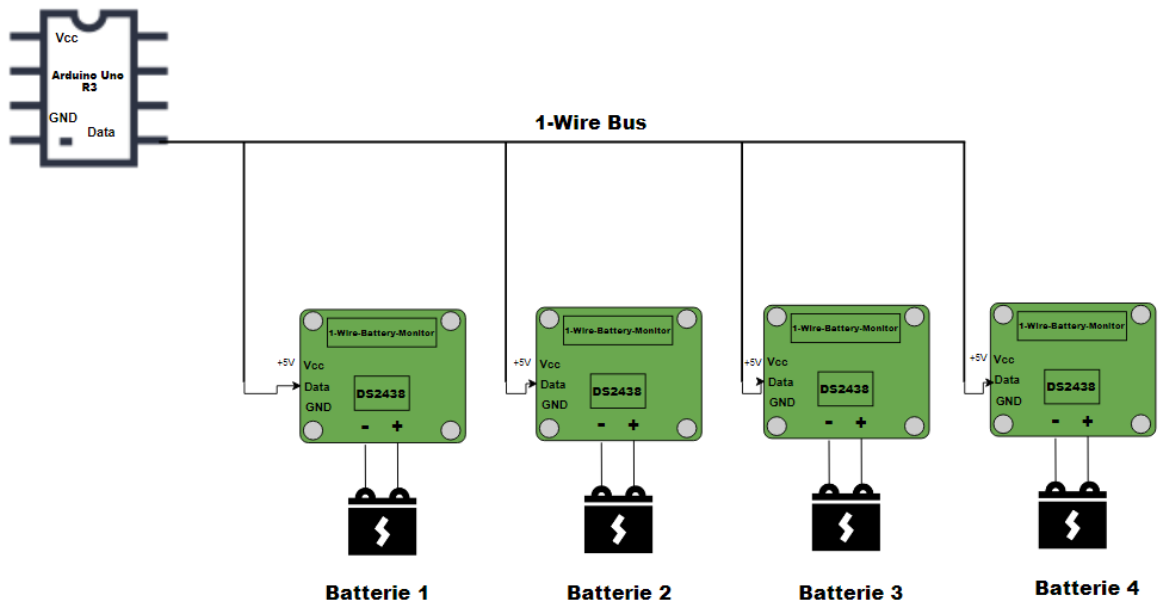
LabVIEW Diagramm für eine USB Schnittstelle mit 4 angeschlossenen Battery Monitor



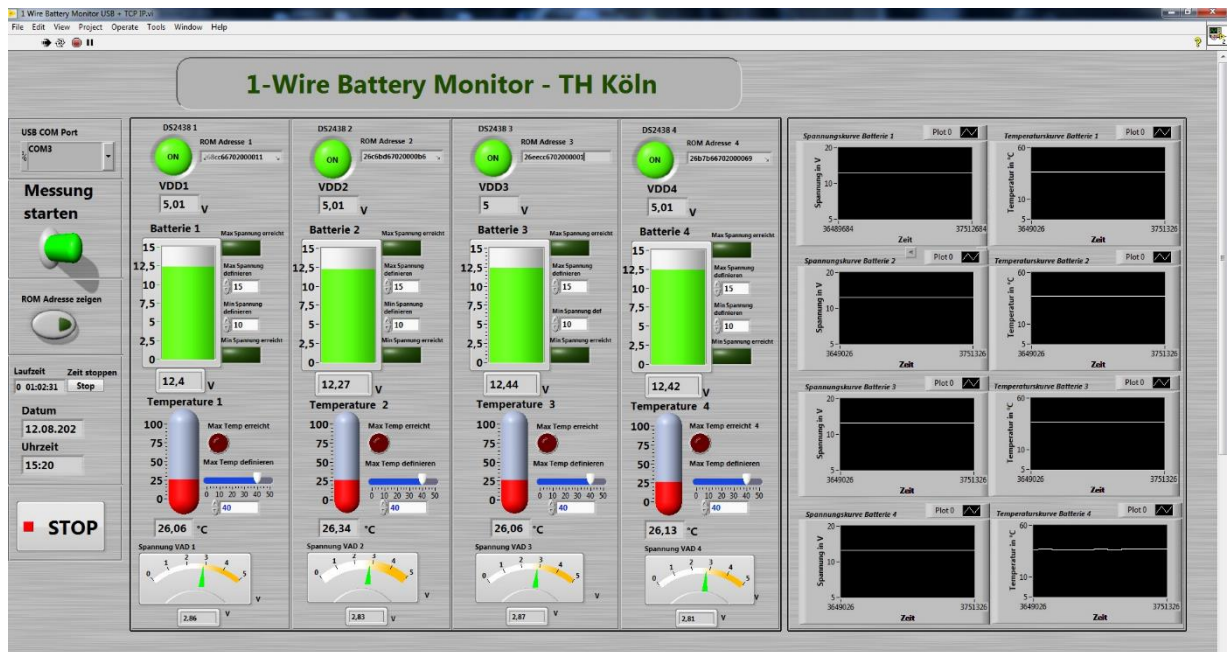
LabVIEW Diagramm für eine TCP/IP Schnittstelle mit vier angeschlossenen Battery Monitor



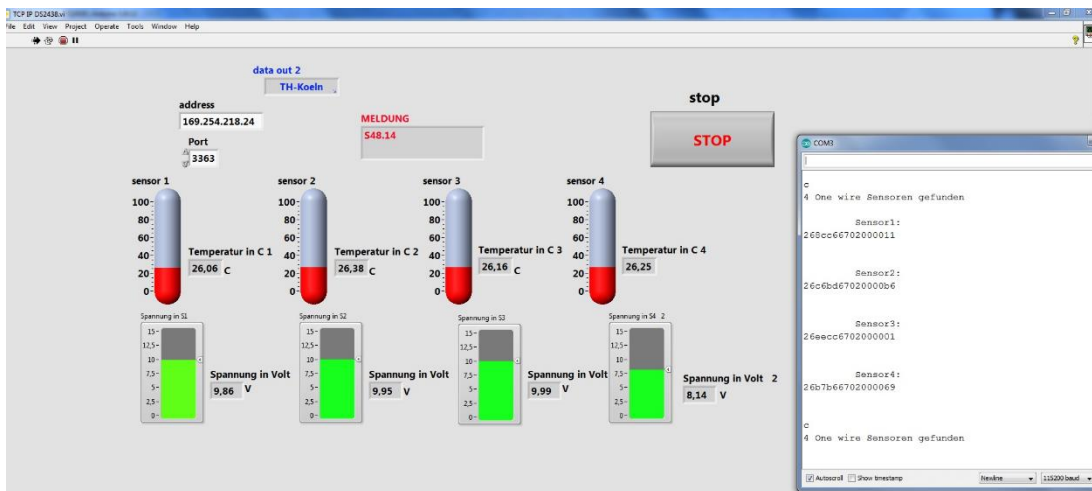
Schaltplan für eine Parallel Schaltung von 1-Wire Battery Monitor:



Visualisierung im LabVIEW nach einer Stunde Laufzeit:



Die Visualisierung mit einer TCP/IP Schnittstelle:



Das Arduino Code für das Auslesen von 1-Wire Battery Monitor

```

/*
    DS2438 Temperatur und Spannung Messung

    Dieser Code stellt die Verwendung der DS2438-Bibliothek zum Auslesen von Temperatur und Spannung von einem DS2438-Batteriemonitor der Firma Dallas Semiconductor unter Verwendung der Arduino One Wire-Bibliothek dar.
*/

//Bibliotheken

#include <Arduino.h> //Arduino Bibliothek

#include <OneWire.h> //OneWire Bibliothek ist von der Firma Dallas Semiconductor und gilt für Zugriff auf 1-wire Sensoren sowie Speicher.

#include <dynaBus.h> //Diese Bibliothek ist zuständig für die Anzahl der Sensoren sowie um die ROM Adressen zu zeigen

#include "DS2438.h" //Die Bibliothek ist zuständig für die Messwerte von Temperatur und Spannung

//Definieren Sie hier den digitalen I/O-Pin des Arduinos, der für das 1-Wire-Bus verwendet werden soll

const uint8_t ONE_WIRE_PIN = 2; //Pin Nummer des Arduino definieren

dynaBus ds(ONE_WIRE_PIN);

// legen Sie hier die 1-Wire Adresse des DS2438 Batteriemonitors fest (lsb zu erst)

uint8_t DS2438_address1[] = { 0x26, 0xb4, 0xea, 0x67, 0x02, 0x00, 0x00, 0x84 };

uint8_t DS2438_address2[] = { 0x26, 0x8c, 0xc6, 0x67, 0x02, 0x00, 0x00, 0x11 };

uint8_t DS2438_address3[] = { 0x26, 0xc6, 0xbd, 0x67, 0x02, 0x00, 0x00, 0xb6 };

uint8_t DS2438_address4[] = { 0x26, 0xee, 0xcc, 0x67, 0x02, 0x00, 0x00, 0x01 };

OneWire ow(ONE_WIRE_PIN); //Digital Pin von Arduino

DS2438 sensor1(&ow, DS2438_address1);
DS2438 sensor2(&ow, DS2438_address2);
DS2438 sensor3(&ow, DS2438_address3);
DS2438 sensor4(&ow, DS2438_address4);

void setup() {
    Serial.begin(115200); //Bau rate definieren
    //Anzahl der 1-Wire Sensoren suchen und finden
    ds.begin();
    ds.find();
    Serial.print(ds.nb()); //Anzahl der Sensoren
    Serial.println(" One wire Sensoren gefunden ");
    Serial.println();
    /*Sensoren initialisieren*/
    sensor1.begin();
    sensor2.begin();
    sensor3.begin();
    sensor4.begin();
    /*ROM Adresse suchen und zeigen*/
    for (int j = 0; j < ds.nb(); j++) {
        Serial.println("");
        Serial.print("Adresse" + String(j + 1) + ": " );
        ds.ROMtochar(j);
        Serial.println();
        Serial.println();
    }
}

void loop() {
    /*Reset Funktion um den Sensoren zu reseten*/
    sensor1.update();
    sensor2.update();
    sensor3.update();
    sensor4.update();

    /* Falls der Sensor 1 nicht da ist dann sind die Werte alle auf Null*/
    if (sensor1.isError()) {
        Serial.println("Sensor 1 nicht ansprechbar ");
        Serial.println("Temp1: 0.00 " );
        Serial.println("Spannung1: 0.00 ");
        Serial.println("VAD1: 0.00");
        Serial.println(" " );
    }
    else {
        Serial.println("Sensor 1 : ");
    }
}

```

```

Serial.println(" ");

String temperature1 = String(sensor1.getTemperature(), 2); //Temperatur Werte aufrufen

Serial.println("Temp1: " + temperature1);

Serial.print("VCC1 = ");

Serial.print(sensor1.getVoltage(DS2438_CHB), 2);
//der Wert von der Eingang Spannung von den Sensor auslesen

Serial.println("");

String vad1 = String(sensor1.getVoltage(DS2438_CHA), 2); //der Wert von der Spannung am Widerstand auslesen

Serial.print("VAD1: ");

Serial.print(vad1); //Serial.print(" v.");

Serial.println();

float Spannung1 = float((vad1.toFloat() * 4.3) + 0.1);
//Spannung mit der Spannungsteiler Regel berechnen und zeigen

Serial.print("Spannung1: ");

Serial.print(Spannung1); //Serial.print(" v.");

Serial.println();

Serial.println();

}

/* Falls der Sensor 1 nicht da ist dann sind die Werte alle auf Null*/

if (sensor2.isError()) {

Serial.println("Sensor 2 nicht ansprechbar");

Serial.println("Temp2: 0 ");

Serial.println("Spannung2: 0 ");

Serial.println("VAD2: 0");

Serial.println(" ");

Serial.println(" ");

}

else {

Serial.println("Sensor 2 : ");

Serial.println(" ");

String temperature2 = String(sensor2.getTemperature(), 2); //Temperatur Werte aufrufen

Serial.println("Temp2: " + temperature2);

Serial.print("VCC2 = ");

Serial.println(sensor2.getVoltage(DS2438_CHB), 2); //der Wert von der Eingang Spannung von den Sensor auslesen

Serial.println("");

String vad2 = String(sensor2.getVoltage(DS2438_CHA), 3); //der Wert von der Spannung am Widerstand auslesen

Serial.print("VAD2: ");

Serial.print(vad2);

Serial.println();

float Spannung2 = float((vad2.toFloat() * 4.3) + 0.1); //Spannung mit der Spannungsteiler Regel berechnen und zeigen

Serial.print("Spannung2: ");

Serial.println(Spannung2);

Serial.println();

}

/* Falls der Sensor 1 nicht da ist dann sind die Werte alle auf Null*/

if (sensor3.isError()) {

Serial.println("Sensor 3 nicht ansprechbar");

Serial.println("Temp3: 0.00 ");

Serial.println("Spannung3: 0.00 ");

Serial.println("VAD3: 0.00");

Serial.println("VCC3 = 0.00");

Serial.println(" ");

}

else {

Serial.println("Sensor 3 : ");

Serial.println(" ");

String temperature3 = String(sensor3.getTemperature(), 2); //Temperatur Werte aufrufen

Serial.println("Temp3: " + temperature3);

Serial.print("VCC3 = ");

Serial.print(sensor3.getVoltage(DS2438_CHB), 2); //der Wert von der Eingang Spannung von den Sensor auslesen

Serial.println("");

String vad3 = String(sensor3.getVoltage(DS2438_CHA), 3); //der Wert von der Spannung am Widerstand auslesen

Serial.print("VAD3: ");

Serial.print(vad3);

Serial.println();

```

```

    float Spannung3 = float((vad3.toFloat() * 4.3) +
0.1); //Spannung mit der Spannungsteiler Regel berech-
nen und zeigen

    Serial.print("Spannung3: ");

    Serial.println(Spannung3);

    Serial.println();

}

/* Falls der Sensor 1 nicht da ist dann sind die Werte
alle auf Null*/

if (sensor4.isError()) {

    Serial.println("Sensor 4 nicht ansprechbar");

    Serial.println("Temp4: 0.00 ");

    Serial.println("Spannung4: 0.00 ");

    Serial.println("VAD4: 0.00");

    Serial.println("VCC4 = 0.00");

    Serial.println(" ");

}

else {

    Serial.println("Sensor 4: ");

    Serial.println(" ");

    String temperature4 = String(sensor4.getTempera-
ture(), 2); //Temperatur Werte aufrufen

    Serial.println("Temp4: " + temperature4);

    Serial.print("VCC4 = ");

    Serial.print(sensor4.getVoltage(DS2438_CHB),
2); //der Wert von der Eingang Spannung von den Sen-
sor auslesen

    Serial.println("");

    String vad4 = String(sensor4.getVol-
tage(DS2438_CHA), 3) ; //der Wert von der Spannung
am Widerstand auslesen

    Serial.print("VAD4: ");

    Serial.print(vad4);

    Serial.println();

    float Spannung4 = float((vad4.toFloat() * 4.3) +
0.1); //Spannung mit der Spannungsteiler Regel berech-
nen und zeigen

    Serial.print("Spannung4: ");

    Serial.println(Spannung4);

    Serial.println();

}

delay(5000); //Anfrage nach 5 Sekunden wiederholen

}

```

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 30. 08.2021

Unterschrift

Mohammed Amine, Jazouli

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 30. 08.2021

Unterschrift

Mohammed Amine, Jazouli

A handwritten signature in blue ink, consisting of stylized initials and a long horizontal stroke.

TH Köln
Gustav-Heinemann-Ufer 54
50968 Köln
www.th-koeln.de