

Staatlich geprüfter Techniker

Fachrichtung Elektrotechnik



Abschlussarbeit

2020

Laufschriftsteuerung mittels Bluetooth-Konsole

vorgelegt von:

Alexander Wiltz

Gutachter:

M.Sc. Thomas Bertel

Danksagung

Hiermit bedanke ich mich bei meiner Frau Julia-Annika Wiltz dafür, dass Sie einerseits an mich glaubt, und andererseits mir beim Lernen und Arbeiten den nötigen Freiraum gegeben hat und mich auch in den schwierigen Phasen wiederaufgerichtet hat.

Ein weiterer Dank geht an meinen Betreuer der Arbeit und Dozenten in Elektrotechnik und Elektronik, Herrn M.Sc. Thomas Bertel, für die Ratschläge und die Unterstützung bei der Entwicklung der Schaltung.

Zusätzlich bedanke ich mich bei meiner Klasse und den Zusammenhalt, mit der es wirklich Spaß gemacht die gesamte Zeit durchzuhalten.

Erklärung

Hiermit erkläre ich, Alexander Wiltz, dass ich die vorliegende Arbeit selbstständig und ausschließlich mit den zugelassenen und angegebenen Hilfsmitteln erstellte.

Wörtliche bzw. sinngemäße Übernahmen aus anderen Quellen sind als solche gekennzeichnet. Diese Arbeit wurde zuvor an keiner anderen Stelle zur Benotung oder Veröffentlichung abgegeben.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abkürzungsverzeichnis	1
1 Einführung	2
1.1 Beweggründe	2
1.2 Grundlagen.....	2
1.2.1 Gängige Lösungsmethoden.....	2
1.2.2 Problematik.....	3
1.2.3 Lösungsansatz.....	4
2 Elektronischer Aufbau.....	5
2.1 Blockschaltbild	5
2.2 Der µProzessor - Raspberry Pi	6
2.3 LED-Treiber - MAX7219CNG	9
2.4 LED – Matrix	11
2.5 Spannungsversorgung.....	13
2.5.1 Berechnung der maximalen Ströme.....	13
2.5.2 Reale Messwerte	15
2.5.3 Kühlkörperberechnung.....	15
2.6 Steuerung der LED-Matrix-Elemente	16
2.7 Bluetooth – Verbindung	16
3 Software	18
3.1 Übersicht der Software	18
3.2 Bibliotheken	20
3.3 Eigene Software	24
3.4 Systemsoftware und Applikationen	26
3.4.1 Raspbian.....	26
3.4.2 SSH und SCP Zugriff	27
3.4.3 SPI	27
3.4.4 Python.....	28
3.4.5 Remote Zugriff	29
3.4.6 MAX7219 Bibliothek.....	29
3.4.7 Autostart.....	30
3.4.8 Bluetooth.....	30
3.4.9 Software-Einrichtung.....	33
3.4.10 Syntax	35
4 Inbetriebnahme	37

4.1	Bedienung und Funktionen	37
5	Zusammenfassung	39
5.1	Zeitplanung	39
5.2	Resümee.....	39
5.3	Erweiterungsmöglichkeiten.....	41
	Quellenverzeichnis	42
	Abbildungsverzeichnis	44
	Tabellenverzeichnis.....	45
	Anhang	46
A.1	Zeichnungen	46
A.1.1	Schaltplan LED-Treiber.....	46
A.1.2	Layout LED-Treiber.....	47
A.2	Code.....	48
A.2.1	Programmablaufplan Clientsoftware	48
A.2.2	Programmablaufplan Raspberry	49
A.2.3	Bluetooth-Clientsoftware	51
A.2.4	Anwender-Software	54
A.3	Eingesetzte Software.....	64

Abkürzungsverzeichnis

GPIO	General Purpose Input/Output
BCM	Broadcom Pin Number = GPIO
CSI	Camera Serial Interface
DSI	Display Serial Interface
SD	Secure Digital
HDMI	High Definition Multimedia Interface
BLE	Bluetooth Low Energy
SSH	Secure Shell
WinSCP	Windows Secure Copy
AC	Alternating Current (dt. Wechselstrom)
DC	Direct Current (dt. Gleichstrom)
I ² C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
LED	Light Emitted Diode
IC	Integrated Circuit (Integrierter Schaltkreis)
CLK	Clock, hier das Taktsignal
SCLK	Serial Clock
DIG	Digital, dt: digital
SEG	Segment, dt. Segment
TTL	Transistor-Transistor-Logik
DIN	Digital-IN
DOUT	Digital-OUT
MOSI	MasterOut SlaveIn
MISO	MasterIn SlaveOut
CS0	Chip Select 0
LSB	Least Significant Bit, niedrigstwertigstes Bit
MSB	Most Significant Bit, höchstwertigstes Bit
RFCOMM	Radio Frequency Communication
L2CAP	Logical Link Control and Adaptation Protocol

1 Einführung

1.1 Beweggründe

Die Idee zu dieser Arbeit kam durch das große Interesse an Elektronik und Mikroprozessoren.

Ein kleiner Anstoß kam durch Herrn Bertel, der Elektronik an der Technikerschule unterrichtet. Die Aufgabe besteht darin, eine Laufschriftanzeige mittels Raspberry Pi zu lösen, der Befehle empfängt und diese entsprechend ausgibt. Als Übertragungsmedium wird Bluetooth mit einem Computer und Mobilfunkgerät genutzt, das mittels Konsole die anzugebenden Befehle zum Rechner übergibt. Ebenso war die Herausforderung, eine eigene Bibliothek in Python 3 zu entwickeln.

1.2 Grundlagen

Die Beschaffung der LEDs ist aufgrund des aktuell hohen Stellenwerts der Leuchtmittel als verbrauchsarme und effiziente Elektronikbauteile verhältnismäßig einfach.

Eine Entscheidung fiel zuerst auf grüne Leuchtdioden, die aus rein subjektiver Sicht ansprechend waren, mit relativ hoher Helligkeit und von einem speziellen LED-Versandhandel auf einer bekannten Internetbörse bezogen wurden.

Im weiteren Verlauf wurde der Einfachheit wegen auf die Matrixmodule aus den Versuchen gewechselt, die in Kapitel 1.2.3 beschrieben werden.

1.2.1 Gängige Lösungsmethoden

Handelsübliche Laufschriften werden in den meisten Fällen mit Schieberegistern gelöst, die von einem ATMEL-Mikroprozessor (oder vergleichbar) gesteuert werden.

Dabei wird vom Controller immer ein Bit in das Register des Controllers gesetzt und die anderen Bits um je eine Stelle verschoben.

Zur Steuerung eines 8x8-LED-Elements müssten zwei Schieberegister benutzt werden, jeweils für die Anoden und für die Kathoden.

Exkurs: Ein Schieberegister ist ein logisches Schaltwerk, in dem mehrere Flip-Flops in Reihe geschaltet sind und deren Inhalt bei jedem Takt um jeweils ein Bit verschoben werden. Die Register arbeiten nach dem First-In-First-Out-Prinzip, was bedeutet, dass das erste gespeicherte Bit das Register als erstes verlässt.

Diese Lösungen sind zwar verhältnismäßig günstig und leicht zu montieren, allerdings sind die Möglichkeiten der Erweiterung mit größerem Aufwand durch Verdrahtung und Softwareänderungen verbunden.

1.2.2 Problematik

Bei der Systemwahl zum Ansprechen der Treiber stehen SPI und I²C als gängigste Systeme zur Auswahl.

Eine Gegenüberstellung:

Funktion	SPI	I ² C
Geschwindigkeit	Bis 80MHz	100-400kHz, max. 5MHz
Standard	nein	ja
Entwickler	Motorola	Phillips Semiconductors
Art	„unechtes“ 2-Draht-System (besitzt im Grunde 4 Leitungen)	Zweidrahtbussystem
Weitere Nennenswerte Besonderheiten	Einfach gestaltet, sehr geeignet für Kommunikationslösungen in kleinen Systemen, einfache Implementierung	Bustakt wird vom Master generiert, Adressierung der Busteilnehmer

Tabelle 1.1: Gegenüberstellung SPI-I²C-Bus

Bei der Auswahl des Treiberbausteins ist die Wahl der Kommunikationsschnittstelle auf den SPI-Bus gefallen, um den I²C des Raspberry Pis für eventuelle Schnittstellenaufgaben mit anderen Mikrocomputern freizuhalten.

Bei diesem Bussystem lassen sich einfache digitale Schaltungen mittels Master-Slave-Prinzip verbinden.

Der wohl gängigste Treiber mit der größten Informationsvielfalt ist der MAXIM MAX7219.

Mit diesen Bausteinen gibt es viele bereits vorgefertigte Module aus Fernost, beispielsweise von Geekcreit®, mit denen einfache Experimente aufgebaut werden können, um das Betriebsverhalten zu untersuchen.

Die Reichweite der Informationen und das Bestehen etlicher Bibliotheken für den MAX7219 bieten die besten Möglichkeiten, sich mit genau diesem Typ zu beschäftigen und die Aufgabenstellung zu lösen.

Zum Betrieb der Anzeige eignet sich ein Raspberry Pi, da der kleine Computer vielseitig ist, etliche Schnittstellen liefert und die Programmiersprache Python unterstützt wird, welche im Rahmen der Technikerausbildung vermittelt wurde.

1.2.3 Lösungsansatz

Zum Testen der grundsätzlichen Funktionen werden acht fertige Module seriell an einen Raspberry Pi angeschlossen und mit 5 V DC versorgt. Im Vorfeld bietet sich an, die Software sowohl auf Funktion als auch auf den benötigten Umfang zu testen und anzupassen.

Auf den Modulen stecken DOT-Matrix-Elemente. Der Anspruch, die Elemente selbst herzustellen, indem die benötigten LEDs zu einem Element verlötet werden, ist zwar eine Option, bietet sich aber aus Kostengründen nicht an.

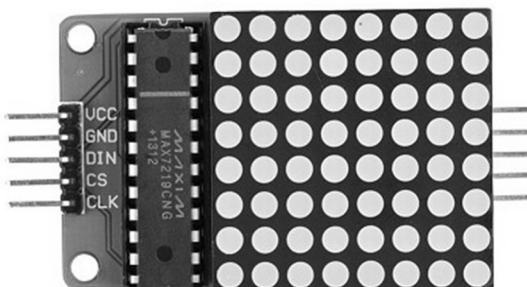


Abbildung 1.1: Matrix-Fertigmodul

Nach ausgiebigen Tests der bestehenden Bibliotheken, die im Internet zu finden sind, um deren Aufbau und Struktur zu verstehen, wächst der Anspruch, eine eigene Software zu entwickeln, die dem angedachten Funktionsumfang gerecht wird.

2 Elektronischer Aufbau

Im nachfolgenden Teil werden der Einsatz und der Aufbau der elektronischen Elemente beschrieben.

2.1 Blockschaltbild

Das nachfolgende Blockschaltbild zeigt den geplanten schematischen Aufbau der Steuerung mit sechs Displays: der Mikrocontroller, auf dem eine autarke Software zur Steuerung läuft, und die Bluetooth-Schnittstelle, die die anzuzeigenden Daten von einem Mobilgerät empfängt und verarbeitet.

Verknüpft mit dem Controller via SPI sind die LED-Treiberbausteine, die jeweils ein Display ansprechen.

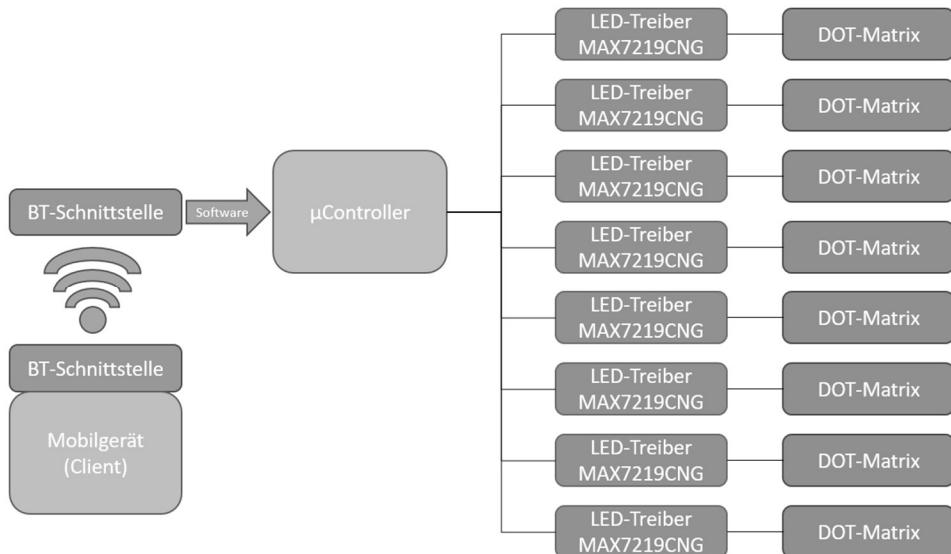


Abbildung 2.1: Blockschaltbild

Die Anzahl der Displays ergibt sich aus der Vorstellung, eventuell eine Zeitanzeige herzustellen, bei dem auf jeweils einem Element eine Zahl steht und die Uhrzeit im Format hh:mm:ss anzeigt.

2.2 Der µProzessor - Raspberry Pi

Zur Steuerung wird ein Raspberry Pi 3 B eingesetzt.

Spezifikationen:

CPU:	ARM Cortex-A53
Kerne:	4
Takt:	1200 MHz
Architektur:	ARMv8 (64-bit)
GPU:	Broadcom Dual Core VideoCore IV, Full HD 1080p30
GPU-Takt:	300 MHz
Videoausgabe:	Composite Video, HDMI (Typ A)
Ton:	HDMI (digital), 3,5 mm Klinke (analog)
Arbeitsspeicher:	1024 MB LPDDR2-SDRAM
HDD:	microSD
USB:	4x USB 2.0
Netzwerk:	10/100 Mbit
WLAN:	Broadcom BCM43143 2,4 GHz, b/g/n
Bluetooth:	4.1LE
GPIO-Pins:	26
Schnittstellen:	1xCSI, 1xDSI, 1xI ² C
Leistung:	4 W (800 mA)
Spannung:	5 V über Micro-USB-B

Der Grund für den Einsatz eines Raspberry Pi 3 liegt in erster Linie an der Menge der zur Verfügung gestellten Schnittstellen und der Kombination aus vollwertigem Computer und steuerbaren Pins ohne Erweiterungen.

Es besteht der direkte Zugriff aus der Linux Distribution auf die GPIOs mittels Python.

Des Weiteren ist er standardmäßig webfähig und hat sowohl eine Ethernet- als auch eine WLAN-Schnittstelle, die den Fernzugriff erleichtern. Zusätzlich ist eine Bluetooth-Schnittstelle verbaut.

Relevant zur Steuerung sind die GPIOs:

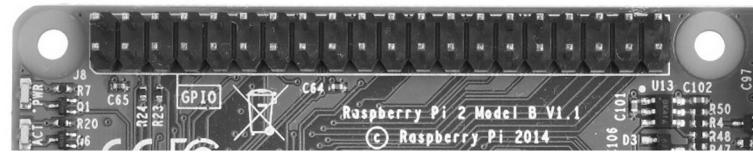


Abbildung 2.2: Pin-Header Raspberry

Die Belegung ist wie folgt:

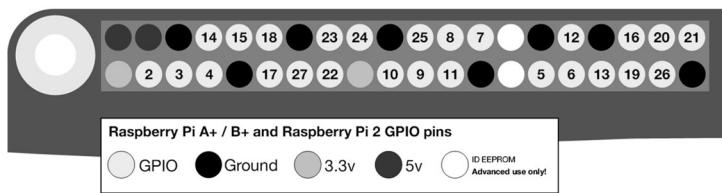


Abbildung 2.3: Pin-Belegung Raspberry mit GPIO-Bezeichnung

Die Pins können sowohl als Eingang wie auch als Ausgang genutzt werden. Werden sie als Ausgang benutzt, geben sie eine Spannung von 3,3 V als HIGH-Signal ab.

Wichtig:

Beim Benutzen als Eingang darf die angelegte Spannung keinesfalls über 3,3 V liegen!

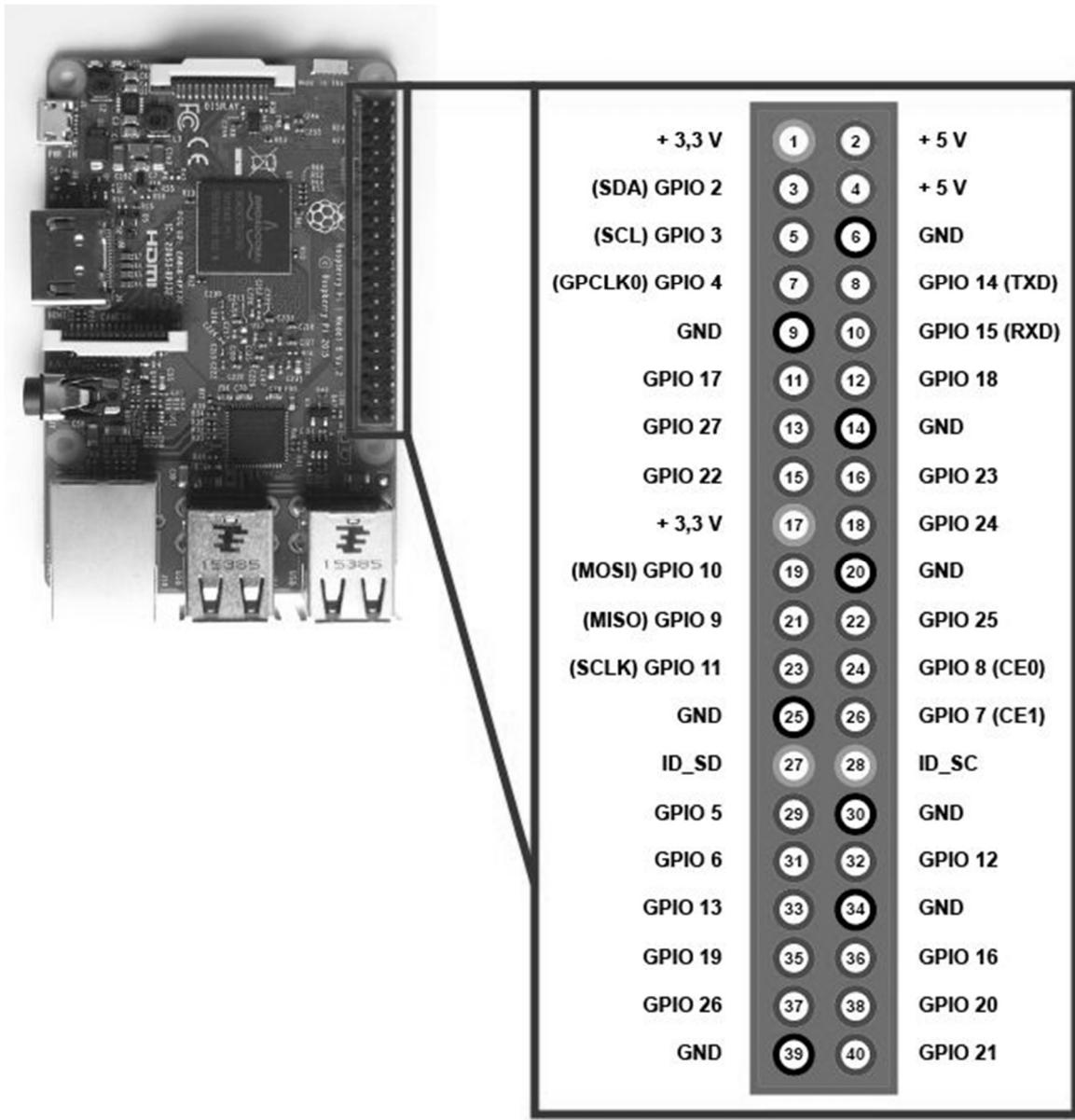


Abbildung 2.4: Pin-Belegung RPi 3B

Zum Betreiben der Treiberbausteine am Raspberry Pi werden folgende Pins belegt:

RPi-Pin	Funktion	MAX7219
2	5 V	VCC
6	GND	GND
24	CE0	CS
23	SCLK	CLK
19	MOSI	DIN

Tabelle 2.1: Belegung zw. RPi und MAX7219

2.3 LED-Treiber - MAX7219CNG

Dieser Treiber kann im Prinzip mit drei Anschlüssen alle Informationen, die er zur Verarbeitung benötigt, von einem Mikrocontroller empfangen und verarbeiten.

Entwickelt wurde der Baustein für den Betrieb von Sieben-Segment-Anzeigen und Leuchtdioden.

Angeschlossen wird der Baustein an den SPI-Bus und kann bis zu 64 LEDs (8x8) steuern. Durch Kaskadieren von weiteren Treiberbausteinen können Anzeigen innerhalb der technischen Grenzen nahezu unbegrenzt verlängert bzw. vergrößert werden.

Die maximale Strombelastung beträgt 500 mA, daher muss über den Pin I_{SET} ein entsprechender Widerstand zur Strom- bzw. Helligkeitsregulierung aus dem Diagramm im Datenblatt gewählt werden.

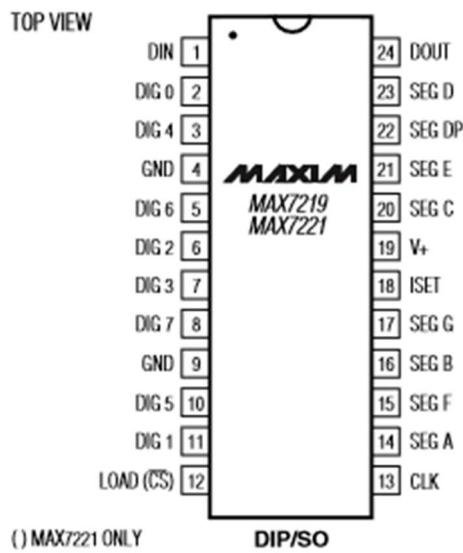


Abbildung 2.5: Pin Belegung MAX7219
(Auszug aus dem Datenblatt)

An die Anschlüsse DIG 0-DIG 7 werden jeweils die Kathoden der LEDs angeschlossen. Die Anoden werden mit den Pins SEG A – SEG G, SEG DP verbunden.

An den Anschluss des I_{SET} wird ein 10 kΩ Widerstand zur Vorbelegung der Beleuchtungsintensität verdrahtet. Die Pins 19 (V+) wie auch 4 und 9 (GND) werden mit der TTL-Spannung von 5 V DC versorgt.

An den Mikrocontroller kommen zur Kommunikation die Anschlüsse LOAD an den CS0 und das Serial-Clock-Signal (CLK) an den SCLK Anschluss.

Weiterhin wird Serial-Data-Input (DIN) mit dem MOSI-Pin des Raspberry verbunden.

Anmerkung: die Pins DIN, CLK und LOAD arbeiten laut Datenblatt in einem Bereich von -0.3 V bis 6 V.

Damit der Treiberbaustein die gewünschten Operationen durchführt, müssen Befehle im Hexadezimalformat gesendet werden. Die Befehle bestehen aus einem Register und einem Adressbereich. Nachfolgend ein Auszug der Register und Adressierungstabelle:

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xA
Scan Limit	X	1	0	1	1	0XB
Shutdown	X	1	1	0	0	0XC
Display Test	X	1	1	1	1	0XF

Abbildung 2.6: Register Address Map MAX7219
(Auszug aus dem Datenblatt)

Beim Starten wird der MAX7219 initialisiert. Das bedeutet, alle Register werden resettet und der Baustein wechselt in den „Shutdown-Mode“. Diese Abläufe sollten eingehalten werden, da sonst auf der Matrix keine (korrekte) Ausgabe erscheint.

Die genauen Abläufe zum Betreiben bzw. Initialisieren der Laufschrift ist dem Programmablaufplan aus Kapitel 3.1. zu entnehmen.

Die Beleuchtungsintensität wird in 16 Halbschritten festgelegt. Das bedeutet, das Minimum beträgt 1/32 bei Register 0xX0 und Adresse 0000_{BIN}. Durch Aufaddieren von jeweils 1/16 ist das Maximum bei 31/32 erreicht (Register: 0XF, Adresse 1111_{BIN}).

Weiterhin müssen „Display-Test“ und der „Decode-Mode“ berücksichtigt werden. Der „Decode-Mode“ legt fest, ob eine LED-Matrix oder eine Sieben-Segment-Anzeige betrieben wird. Dazu wird im Adressbereich die „0“ gesendet. Der Display-Test bringt lediglich die LEDs alle zum Leuchten, wenn dieser aktiviert wurde. Da diese Betriebsart im Normalbetrieb nicht benötigt wird, sendet man auch hier eine „0“.

Zur Vermeidung von Stromspitzen wird die Versorgungsspannung mit einem 10 µF Elektrolytkondensator und parallel zwischen V+ und GND einem 0,1 µF Keramikkondensator gepuffert.

Weiterhin wird allgemein empfohlen, die Entfernung zwischen den Chips und der LEDs so kurz wie möglich zu halten, um Störeinflüsse wie Induktivitäten und elektromagnetische Störungen zu vermeiden.

Das Schalten der Anzeigen erfolgt in etwa mit 800 Hz, was das „Flackern“ bzw. den Signalwechsel für das menschliche Auge unsichtbar macht (zum Vergleich: moderne Fernsehgeräte haben 100-200 Hz).

2.4 LED – Matrix

Die 8x8 Matrix ist nach untenstehendem Schema verbunden. Pro Zeile sind alle Kathoden und pro Spalte alle Anoden miteinander verbunden. So werden 64 LEDs durch einen Baustein betrieben.

Zum Beschalten der unteren rechten LED muss also der Anschluss SEG G und DIG 7 durch den Treiber bestromt werden. Nach diesem Prinzip werden alle Leuchtdioden angesteuert.

Der Vorteil besteht darin, dass mit 16 Kontakten insgesamt 64 LEDs angesteuert werden können. Der Verdrahtungsaufwand reduziert sich also immens.

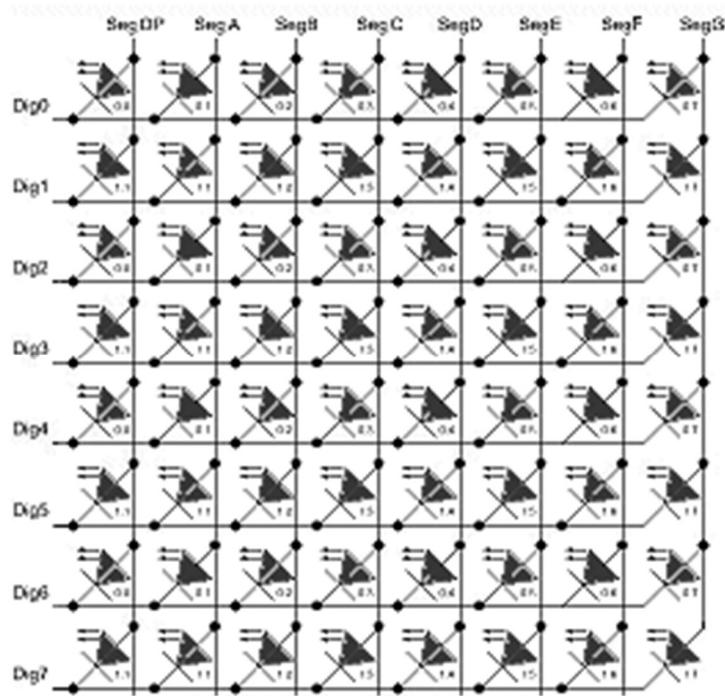


Abbildung 2.7: Verbindungs-/Anschlusschema LEDs

2.5 Spannungsversorgung

Im folgenden Abschnitt wird erläutert, weshalb ein Raspberry Pi Netzteil ausreichend ist.

2.5.1 Berechnung der maximalen Ströme

Der Strom für die Matrixelemente wird durch den Widerstand an I_{SET} festgelegt. Nach dem Diagramm aus dem Datenblatt wird ein $40\text{ k}\Omega$ Widerstand eingesetzt. Demnach ergibt sich bei einer Segmentspannung von 2,1 V eine Strombelastung von etwa 17 mA pro Segment.

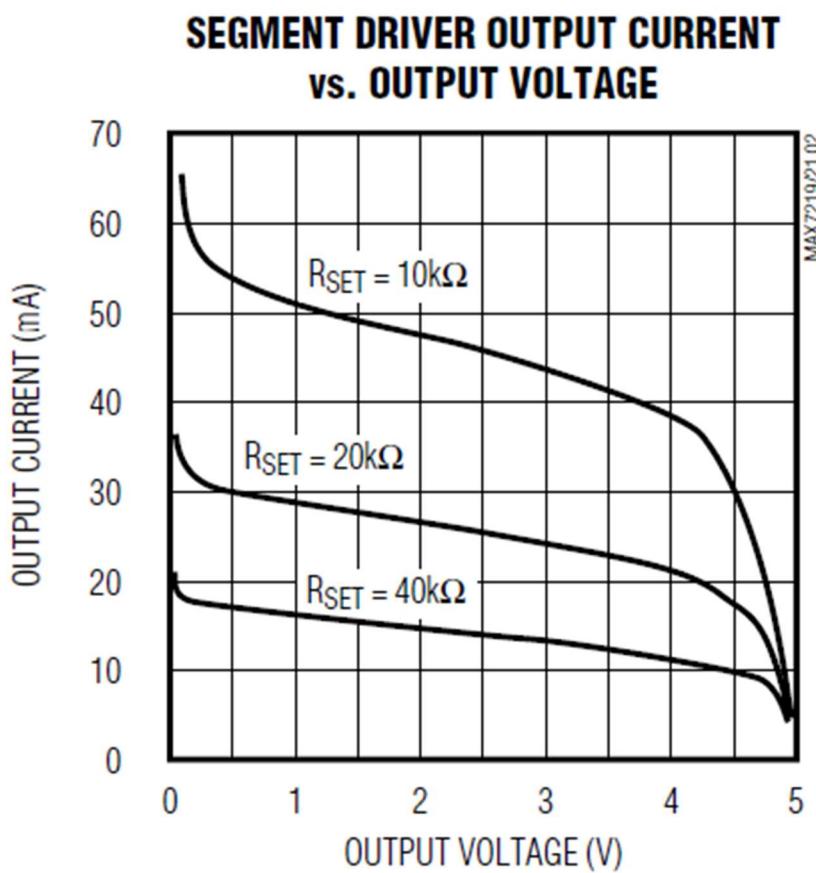


Abbildung 2.8: Ausgangsstrom am Segment
(Auszug aus dem Datenblatt)

Daraus ergibt sich folgende Berechnung:

$$I_{\text{Matrix}} = I_{\text{Segment}} * n = 17 \text{ mA} * 8 = 136 \text{ mA} \quad (2.1)$$

$$I_{\text{ges}} = I_{\text{Matrix}} * n = 136 \text{ mA} * 8 = 1088 \text{ mA} \quad (2.2)$$

Der Raspberry wird mit einem vom Hersteller empfohlenen Netzteil betrieben, das stabile 5 V DC und 2 A liefern kann. Da der Strom über einen USB-Anschluss gespeist wird, ist der Messaufwand zu realen Verbrauchsermittlung aufwändig. Daher wird eine Tabelle zur Verbrauchsermittlung benutzt:

Raspberry Pi	Stromverbrauch		
	Board (typisch)	USB-Peripherie	Maximal
Model A	200 mA	500 mA	700 mA
Model B	500 mA	500 mA	1,2 A
Model A+	180 mA	500 mA	700 mA
Model B+	330 mA	600 mA / 1,2 A (aktivierbar)	1,8 A
Model 2 B	330 mA	600 mA / 1,2 A (aktivierbar)	1,8 A
Model 3 B	330 mA	600 mA / 1,2 A (aktivierbar)	1,8 A

Abbildung 2.9: Offizielle Stromverbrauchswerte

Aufgrund der geringen Stromaufnahme über die Pins 2 und 6 (5 V und GND) und den geringen Eigenbedarf wird auf eine eigens konstruierte Stromquelle verzichtet.

Bauteil	Einzelstrom	Summe
Raspberry Pi	330 mA	330 mA
Schaltung MAX7219	136 mA	1088 mA
Gesamtstrom		1418 mA

Tabelle 2.2: Summe aller Ströme

2.5.2 Reale Messwerte

Die tatsächlich gemessenen Werte ergeben sich wie folgt:

	I_N	I_{MAX}	I_{MIN}
RPi alleine	250 mA	636 mA	0,5 mA
RPi + MAX7219		1665 mA	315 mA
Einzelne LED		-16,72 mA	-17,50 mA

Tabelle 2.3: Reale Messwerte

Die gemessene Spannung am Segment beträgt 2,1 V.

2.5.3 Kühlkörperberechnung

Berechnung der maximalen Verlustleistung eines MAX7219CNG-Moduls nach Formel aus Datenblatt:

$$P_D = (U_{IN} * 8mA) + (U_{IN} - U_{LED})(y_{int} * n_{Seg} * I_{Seg}) \quad (2.3)$$

$$P_D = (5V * 8mA) + (5V - 3V) \left(\frac{31}{32} * 8 * 13mA \right) = 0,2415W \quad (2.4)$$

Maximal zulässige Umgebungstemperatur des MAX7219CNG:

$$T_{J(MAX)} = T_A + P_D * \theta_{JA} \rightarrow T_A = T_{J(MAX)} - P_D * \theta_{JA} \quad (2.5)$$

$$T_A = 150^\circ\text{C} - 0,2415 \text{ W} * 75 \frac{^\circ\text{C}}{\text{W}} = 131,8875^\circ\text{C} \quad (2.6)$$

Demnach wird für die LED-Treiber kein Kühlkörper benötigt.

2.6 Steuerung der LED-Matrix-Elemente

Auf der Platine zur Verbindung der Treiberbausteine mit den LED-Modulen und dem Mikrocomputer sind lediglich die Bausteine als solche montiert, die mit Sockelleisten verbunden sind, um eine Steckverbindung mit den Matrix-Elementen und dem Raspberry herstellen zu können.

Weiterhin befinden sich zu jedem MAX7219 jeweils ein Tantalkondensator (C1 und C2) zur Spannungspufferung und ein 10 kΩ-Widerstand zum Einstellen des Beleuchtungsstroms für die Helligkeitsregulierung.

Zusätzlich ein SPI-Pin-Out, um die einzelnen Platinen einfach zu kaskadieren und im Fehlerfall modular austauschen zu können.

Die Platinen für das Projekt wurden bei <https://jlpcb.com> bestellt und gefertigt.

2.7 Bluetooth – Verbindung

Mit der Kurzstreckenfunktechnik Bluetooth wird eine 2 Byte große Information von einem beliebigen Gerät in einem Frequenzbereich zwischen 2,402 und 2,480 GHz drahtlos an den Raspberry Pi gesendet.

Bei den zu verbindenden Geräten gibt es teilweise Einschränkungen. So können Apple-Geräte nicht mit dem Mikrocomputer über die Bluetooth-Schnittstelle kommunizieren.

Bei Android-Geräten ist die Funktionalität dahingehend eingeschränkt, dass die Vertraulichkeit eingestellt und die dauerhafte Sichtbarkeit aktiviert sein muss.

Die einfachste Möglichkeit bietet ein entsprechender Adapter für Computer. Damit ist man maximal flexibel, da weitere Anwendungen erstellt werden können.

Zum Zwecke der Versuche für diese Arbeit ist ein USB-Adapter am Computer eingesteckt. Auf dem Client (hier der Computer) läuft eine eigens erstellte Anwendersoftware.

Die Software ist ebenfalls in Python erstellt und bietet die Möglichkeit, mittels bestimmter festgelegter Syntax Anzeigeparameter einzustellen und Textteile anzeigen zu lassen.

Der hier eingesetzte Raspberry Pi 3B hat bereits einen Bluetooth-Adapter integriert. Dabei handelt es sich um Bluetooth 4.1, Bluetooth Low Energy, um den Stromverbrauch zu reduzieren.

3 Software

In diesem Kapitel werden die maßgeblichen Inhalte der verwendeten Module und der Ablauf des Anwenderprogramms erläutert.

3.1 Übersicht der Software

Beim Start des Raspberry Pi wird nach Systemhochlauf die Anwendersoftware gestartet. Dazu der nachstehende Programmablauf grafisch dargestellt:

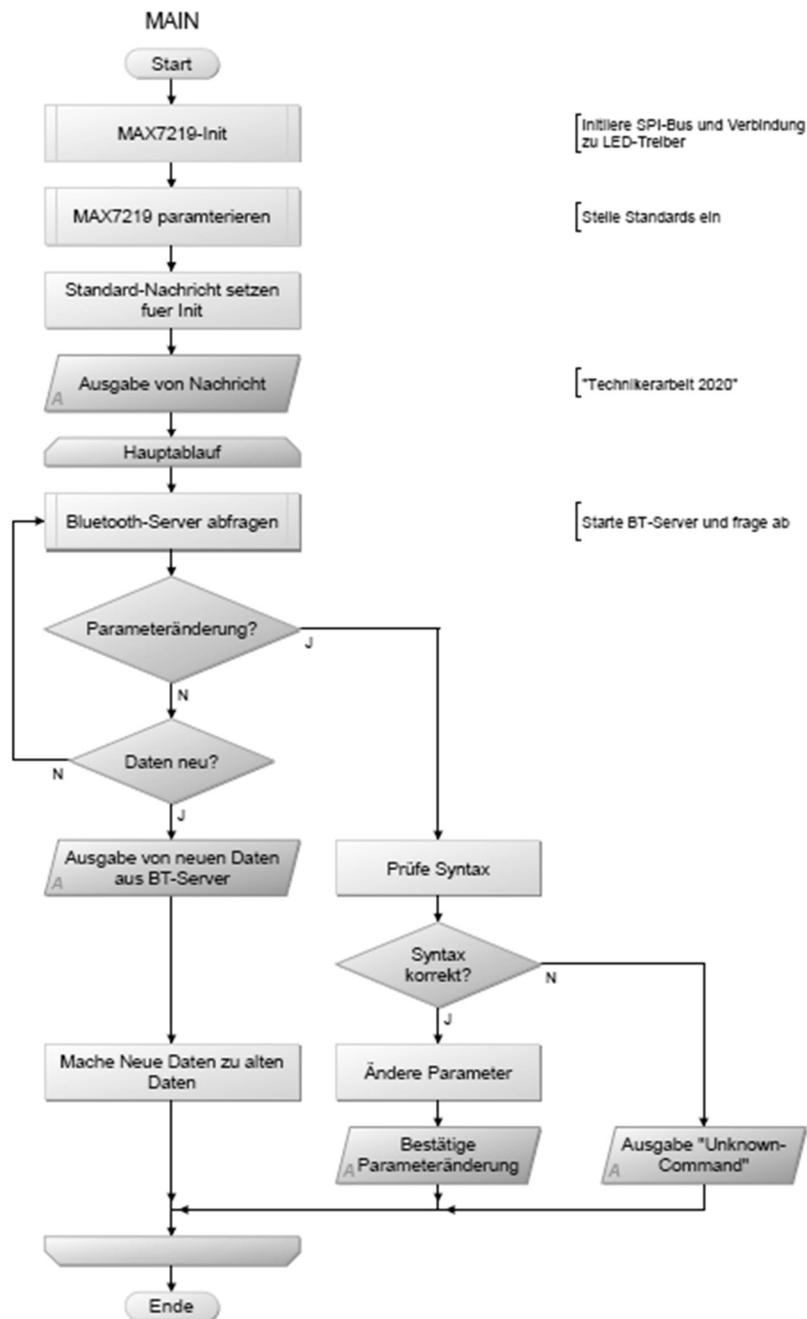


Abbildung 3.1: Ablaufbeschreibung Bildschirmausgabe komplett

Durch Aufruf der Funktion initDisplay() werden die Treiber für die Anzeigen initialisiert. Dementsprechend wird vor Beginn der eigentlichen Programmbehandlung eine Kommunikation mit vorgegebenen Parametern über den SPI-Bus hergestellt. Dabei werden der Bus geöffnet und die Busgeschwindigkeit und der Übertragungsmodus festgelegt (Modi sind dem Datenblatt SPI zu entnehmen).

Eingestellte Werte sind:

bus	0	//SPI-Bus CS0 wird verwendet
device	0	//Gerät 0 am Bus wird angesprochen
max_speed_hz	100000	//100 kHz Maximalwert
mode	0	//Bus-Modus 0
cascaded	8	//Kaskadierung von 8 Geräten
brightness	3	//Intensität von 3 bei 0..15

Es wird festgelegt, dass alle Digits benutzt werden, der Decodemodus ausgeschaltet ist, kein Displaytest stattfindet und das Gerät nicht in den Standbymodus wechselt. Im Anschluss kann und sollte man die Beleuchtungsintensität festlegen.

Nachdem diese Schritte abgeschlossen sind, sollte der Anzeigeninhalt per clear() geleert werden. Zusätzlich wird die Drehrichtung der Zeichen per orientation() auf 90° gestellt.

Ab diesem Punkt kann das Gerät in eine Schleife übergehen und auf den Empfang von neuen Daten warten. Allerdings wird im beschriebenen Ablauf nochmals eine einzelne Nachricht ausgegeben, bevor der Bluetooth-Server startet.

Kommt es zum Empfang, werden die Daten auf Plausibilität geprüft, bei FALSCH verworfen und bei WAHR verwendet. Im Anschluss werden die Textdaten, sofern in Ordnung, an die Ausgabefunktion übergeben und angezeigt. Der Empfang und das Prüfen werden also als Interrupt der Anzeige genutzt.

Die Anzeige wiederholt die Schrift so lange wie festgelegt oder der Modus per Bluetooth umgestellt wird.

Außerdem gibt es die Möglichkeit, über eine eigens angelegte Syntax die Parameter der Anzeigefunktionen zu beeinflussen. Die Auswertung erfolgt unmittelbar nach dem Empfang des Strings. Die Python-Software prüft also nach Empfang, ob es sich um einen parameterbeeinflussenden Wert handelt oder lediglich um ein anzuzeigendes Wort.

3.2 Bibliotheken

max7219.led

Es handelt sich um eine selbstgeschriebene Bibliothek zum Betreiben des MAX7219 über SPI von Richard Hull.

Die Bezugsquelle kann dem Quellenverzeichnis entnommen werden.

Benutzte Funktionen aus dem Modul led:

`brightness(brightness)`

Definiert die Helligkeit des Displays

`clear()`

Leert den Inhalt des Displays und setzt die Anzeigen zurück

Weitere mögliche Funktionen:

`orientation(angle)`

Die Ausrichtung des Zeichens pro Display. Mögliche Werte: 0, 90, 180, 270

Sind die Anschlüsse der Kathoden horizontal, ist die Ausrichtung des Elements bei 90 Grad.

`letter(device, msg)`

Gibt ein einzelnes Zeichen aus. Device legt die fortlaufende Nummer der Anzeige fest und msg das anzuzeigende Zeichen. Hinweis: Der Datentyp muss im ASCII-Format übergeben werden!

max7219.font

Das Font-Modul innerhalb der MAX7219 Bibliothek ist eine Erweiterung zur Bibliothek, ebenfalls von Richard Hull. Hier sind einige Schriftarten vordefiniert, die in der festgelegten Klasse parametrisiert werden können.

Standard: CP347

time

Bibliothek zur Formatierung der Systemzeit

Benutzte Funktionen:

`time.sleep(secs)`

Unterbricht für die angegebenen Sekunden den aufgerufenen Thread. Argument kann als eine durch einen Punkt getrennte Gleitkommazahl als Ruhezeit angeben werden.

spidev

Keine Standard Python Bibliothek! (<https://github.com/doceme/py-spidev>)

Modul, um eine Verbindung über den SPI-Bus herzustellen.

Benutzte Funktionen:

`open(bus, device)`

Startet den SPI Bus mit den Parametern „bus“ für die Nummer des benutzten Anschlusses. (Raspberry hat 2) und „device“ für die Geräteadresse der angeschlossenen Module.

`xfer2(list of values[, speed_hz, delay_usec, bits_per_word])`

SPI-Transaktion durchführen. Chip Select wird zwischen den Blöcken gehalten.

Bei xfer wird zwischen den Blöcken Chip Select freigegeben und reaktiviert.

Zusätzlich wird eine Verzögerungszeit in Mikrosekunden übergeben.

`close()`

Beendet die SPI-Verbindung.

datetime

Das Modul stellt Funktionen zur einfachen und komplexen Bearbeitung des Datums und der Uhrzeit bereit.

Benutzte Funktionen:

`datetime.now()`

Gibt die aktuelle Zeit und das aktuelle Datum zurück. Format: date(yy-mm-dd hh:mm:ss:sss)

<code>datetime.now().year:</code>	Gibt das aktuelle Jahr als Integer zurück
<code>datetime.now().month:</code>	Gibt den aktuellen Monat als Integer zurück
<code>datetime.now().day:</code>	Gibt den aktuellen Tag als Integer zurück
<code>datetime.now().hour:</code>	Gibt die aktuellen Stunden als Integer zurück
<code>datetime.now().minute:</code>	Gibt die aktuellen Minuten als Integer zurück
<code>datetime.now().second:</code>	Gibt die aktuellen Sekunden als Integer zurück
<code>datetime.now().microsecond:</code>	Gibt die aktuellen Millisekunden als Integer zurück

`date()`

Formatiert Zahlen (Integer) ins Datumsformat, Bsp: `date(2019, 6, 30)`

os

Diese Bibliothek bildet die Schnittstelle zwischen der Python-Anwendung und dem Betriebssystem.

Benutzte Funktionen:

`os.path.exists(Datei):`

Prüft, ob eine Datei im angegebenen Verzeichnis existiert.

`os.path.getmtime(Datei):`

Liest die Änderungsdaten einer Datei.

bluetooth

Diese Bibliothek bildet die Schnittstelle zwischen der Python-Anwendung und dem Betriebssystem.

Benutzte Funktionen:

`discover_devices()` :

Sucht nach Geräten in Reichweite durch Zugriff auf die Bluetooth-Schnittstelle.

`lookup_name(String)` :

Übergibt die MAC-Adresse eines Gerätes an `discover_devices`. Typ: Liste

`BluetoothSocket(Verbindungstyp)` :

Öffnet einen Kanal über angegebenen Verbindungstyp, z.B. `bluetooth.RFCOMM`

`BluetoothSocket(Verbindungstyp).connect((BT-Adresse,port))`:

stellt die Verbindung her mit MAC-Adresse und Port

`BluetoothSocket(Verbindungstyp).send(String)`:

sendet angegebenen String

`BluetoothSocket(Verbindungstyp).close()`:

schließt die Verbindung

`BluetoothSocket(Verbindungstyp).bind((BT-Adresse,port))`:

Verbindet Server mit Client, Adresse kann offen bleiben

`BluetoothSocket(Verbindungstyp).listen(port)`:

Wartet auf Daten an angegebenen Port

`BluetoothSocket(Verbindungstyp).accept()`:

Akzeptiert Verbindung, gibt zwei Argumente zurück: `ClientSocket` und `ClientAdresse`

`BluetoothSocket(Verbindungstyp).recv(Integer)`:

Gibt die Menge der zu empfangenden Daten in Bits an, z.B. 1024.

3.3 Eigene Software

Am Anfang sollten die Grundparameter, wie die Menge der kaskadierten Displays, angegeben werden.

Es gibt eine Initialisierungsfunktion (`init()`), in der die SPI-Parameter definiert werden, die Verbindung zu den MAX7219 aufgebaut wird und die laut Datenblatt benötigten Schritte softwareseitig gesetzt werden.

Weitere Funktionen:

`brightnessPulses(cascaded, turns=1, endless=False)`

Pulsiert die Anzeigenhelligkeit in Abhängigkeit der Anzahl (turns) oder dauerhaft, sofern die Variable `endless = True` ist.

`static_text(msg, invert=False, font=None)`

Zeigt Text ohne Bewegung an (maximale Zeichenzahl = Anzahl Displays).

Dreht den Text um, wenn `invert = True`. Die Variable `msg` beinhaltet den anzuzeigenden Text. Für `font` ist ein alternativer Zeichensatz geplant, der allerdings noch nicht umgesetzt ist.

`scroll_text(msg, direction=False, invert=False, speed=0.25, turns=1, font=None)`

Lässt einen in `msg` angegebenen Text über das Display laufen. Veränderliche Parameter, wie `direction`, drehen die Startrichtung, `invert` dreht die Buchstaben, `speed` gibt die Scroll-Geschwindigkeit an. Durch `turns` wird die Anzahl der Scroll-Wiederholungen festgelegt.

Font ist noch nicht umgesetzt.

`nearby()`

Sucht alle sich in Reichweite befindlichen Bluetooth-Geräte und zeigt diese als Liste an.

Ausgabe als MAC-Adresse und Gerätename.

```
findDevice(target_adress)
```

Sucht nach einem bestimmten Gerät, dessen MAC-Adresse als Parameter als Typ: string angegeben wurde.

```
startBTClient(target_adress, msg)
```

Startet einen Bluetooth-Client, bei dem als Parameter die MAC-Adresse des Bluetooth-Servers angegeben wird und die zu übertragende Nachricht vom Typ: String.

```
startBTServer()
```

Startet einen Bluetooth-Server am Port 1, der auf den angegebenen Port hört

```
startBTServerOnce()
```

Gleiche Funktionalität wie bei vorangegangener Funktion, allerdings wird dieser genau einmal aufgerufen. Sobald die Nachricht übermittelt wurde, werden der Server und der Client geschlossen.

```
getActualTime()
```

Nutzt die Systemfunktion datetime, um im Aufruf die aktuelle Systemzeit abzurufen. Format: hh:mm:ss

3.4 Systemsoftware und Applikationen

In diesem Abschnitt wird die Einrichtung der Software auf dem eingesetzten Mikrocomputer beschrieben.

3.4.1 Raspbian

Zur Installation sollte auf die zu betreibende microSD mittels Win32DiskImager das Raspbian Image installiert werden.

Um Zugriff über SSH mit dem Kommandokonsolenprogramm Putty zu erhalten, muss im Boot-Bereich der Karte noch eine Datei `ssh` ohne Dateiendung angelegt werden.

Die WLAN-Konfiguration kann ähnlich angelegt werden: Es wird eine Datei mit dem Namen `wpa_supplicant.conf` erstellt und in die Datei kommen folgende Informationen:

```
network={  
    ssid="TestNetzwerk"  
    psk="WLANPasswort"  
}
```

Anschließend die Karte in den entsprechenden Slot am Raspberry Pi einstecken und mit dem passenden Netzteil verbinden.

Die Standard Login-Daten sind:
User: pi
Kennwort: raspberry

Nun das Dateisystem vergrößern, um die gesamte Karte nutzen zu können, und die Software auf den aktuellen Stand bringen:

```
1. sudo raspi-config --expand-rootfs
```

```
2. sudo apt-get update
```

```
3. sudo apt-get upgrade
```

Weiterhin ein Passwort für den root-User anlegen:

```
4.      sudo passwd root
```

Zum Schluss wird der Rechner neugestartet und die Einrichtung ist zunächst abgeschlossen.

```
5.      sudo reboot
```

3.4.2 SSH und SCP Zugriff

Um sich direkt als root-User innerhalb der SSH-Konsole oder innerhalb des Dateiverwaltungssystems SCP anzumelden, muss zwingend eine Konfigurationsdatei bearbeitet werden:

```
1.      sudo su
```

```
2.      nano /etc/ssh/sshd_config
```

Nun zu folgenden Zeilen scrollen:

```
# Authentication:  
LoginGraceTime 120  
PermitRootLogin prohibit-password  
StrictModes yes
```

PermitRootLogin auf yes ändern.

```
3.      shutdown -r now
```

3.4.3 SPI

Standardmäßig ist der SPI-Zugriff nicht aktiviert. Mit wenigen Konsolenbefehlen kann dies geändert werden:

```
1.      sudo raspi-config
```

Nachdem sich ein Menü geöffnet hat, wird Punkt 5 ausgewählt, dieser aktiviert unter SPI den Bus.

3.4.4 Python

Zur Installation der Python-Software müssen im Vorfeld einige Tools installiert werden:

1. sudo apt-get install build-essential
 checkinstall

2. sudo apt-get install libreadline-gplv2-dev
 libncursesw5-dev libssl-dev libsqlite3-dev tk-dev
 libgdbm-dev libc6-dev libbz2-dev

Anschließend kann die eigentliche Python-Software heruntergeladen und entpackt werden:

3. cd /usr/bin

4. sudo wget
 <https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tgz>

5. sudo tar xzf Python-3.7.4.tgz

Jetzt Python konfigurieren und installieren:

6. cd Python-3.7.4

7. sudo ./configure

8. sudo make -j4

9. sudo make altinstall

3.4.5 Remote Zugriff

Um aus Servicegründen über den Remoteservice von Windows zugreifen zu können, wird weiterhin `xrdp` installiert und ein root-User mit Administratorrechten eingerichtet.

```
1. sudo apt-get install xrdp
```

```
2. sudo su
```

```
3. passwd
```

Zugangsdaten:

root

raspberry

3.4.6 MAX7219 Bibliothek

Installation der MAX7219 Bibliothek über GitHub:

```
1. git clone https://github.com/rm-hull/max7219.git
```

```
2. cd max7219
```

```
3. sudo pip install -e .
```

Im Anschluss muss die Erweiterung installiert werden:

```
4. cd max7219
```

```
5. sudo apt-get install python-dev python-pip
```

```
6. sudo pip install spidev
```

```
7. sudo python setup.py install
```

3.4.7 Autostart

Damit die Anwendung nach jedem Neustart oder Spannungsausfall funktioniert, sollte die Software unmittelbar nach dem Hochlauf in Betrieb gehen:

Dazu müssen im Vorfeld die Dateizugriffsrechte angepasst werden:

```
1.      sudo chmod +x /home/pi/main.py
```

Nun wird die Autostartdatei modifiziert:

```
2.      sudo nano /etc/rc.local
```

Die Autostartdatei wird nun um den Eintrag mit der auszuführenden Datei ergänzt:

(*Hinweis:* Eintrag muss vor exit 0 stehen!)

```
3.      python /home/pi/main.py & exit 0
```

3.4.8 Bluetooth

Zur Installation und Konfiguration des Bluetooth-Moduls muss Folgendes beachtet werden:

```
1.      sudo apt-get install bluetooth
```

Damit mit Bluetooth gearbeitet werden kann, müssen die Standardkonfiguration geändert und die Datei /etc/bluetooth/main.conf wie folgt ergänzt werden:

```
2.      DisablePlugins = pnat
```

Anschließend wird der Bluetooth-Service neugestartet:

```
3.      service bluetooth restart
```

Zur Überprüfung, ob der Service korrekt ausgeführt wird:

```
4.      service bluetooth status
```

Weiterhin wurde erkannt, dass der SAP-Server nicht gestartet werden konnte. Da dieser Dienst nicht benötigt wird, ist die entsprechende Konfiguration zu bearbeiten:

```
5. sudo nano  
/etc/systemd/system/bluetooth.target.wants/bluetooth.service
```

Dabei wird die Zeile: ExecStart=/usr/lib/bluetooth/bluetoothd ersetzt durch:

```
6. ExecStart=/usr/lib/bluetooth/bluetoothd --  
noplugin=sap
```

Nachdem die Änderung gespeichert wurde und die Datei wieder geschlossen, müssen die Dienste neugestartet werden:

```
7. sudo systemctl daemon-reload  
  
8. sudo service bluetooth restart
```

Mit hcitool scan können alle eingeschalteten Geräte in Reichweite ermittelt werden.

Neben der Adresse werden die lesbaren Namen des Gerätes angezeigt. Android-Geräte werden standardmäßig nicht erkannt und müssen über die System-Einstellungen, zumindest temporär, sichtbar gemacht werden. Apple-Geräte können nicht mit dem Raspberry gekoppelt werden.

Startet den Bluetooth-Dienst:

```
9. sudo bluetoothctl
```

Nun wird im Dienst weitergearbeitet:

```
10. [bluetooth]# agent on  
  
11. [bluetooth]# pairable on  
  
12. [Bluetooth]# discoverable on  
  
13. [bluetooth]# scan on
```

Nachdem der Dienst aktiviert wurde, Pairing eingeschaltet und das Suchen nach Geräten in Reichweite gestartet ist, sollte das zu koppelnde Gerät in der Liste mit seiner MAC-Adresse erscheinen:

```
14. [CHG] Device 88:78:73:C7:F0:0F Name: ALEX-P  
  
15. [CHG] Device 88:78:73:C7:F0:0F Alias: ALEX-P  
  
16.
```

Jetzt kann das Suchen ausgeschaltet und das Koppeln gestartet werden:

```
17. [bluetooth]# scan off  
  
18. [bluetooth]# pair 88:78:73:C7:F0:0F
```

Wenn die Rückmeldung einer erfolgreichen Verbindung kommt, bietet der Dienst einen Pin an, der zu bestätigen ist:

```
19. [CHG] Device 88:78:73:C7:F0:0F Connected: yes  
  
20. [agent] PIN code: 963064  
  
21. [CHG] Device 88:78:73:C7:F0:0F Paired: yes
```

Anschließend das Gerät auf vertrauenswürdig setzen und die Grundverbindung ist gesetzt:

```
22. [bluetooth]# trust 88:78:73:C7:F0:0F  
  
23. [bluetooth]# connect 88:78:73:C7:F0:0F
```

Hinweis: Sollte bei dem Verbindungsaufbau ein Fehler auftreten, sollte der Pi neugestartet werden. Weiterhin ist die Entfernung zu beachten!

Nachdem die Kommunikationsschnittstelle eingerichtet und getestet wurde, muss das entsprechende Software-Paket für Python installiert werden. Probleme treten bei üblicher Installation auf. Die zur Verfügung stehenden Bibliotheken werden standardmäßig in das Verzeichnis für Python2.7 installiert und können so nicht auf dem Raspberry Pi genutzt werden.

Demnach muss das Problem der Installation und Konfiguration über einige Hürden geschehen:

```
24. sudo apt-get install python-pip python-dev  
     ipython  
  
25. sudo apt-get install bluetooth libbluetooth-dev  
  
26. sudo python3 -m pip install pybluez
```

Die benötigten Pakete sind installiert und es kann versuchsweise über den Computer eine Verbindung über RFCOMM + TCP in Betrieb genommen werden. “Das Bluetooth-Protokoll RFCOMM stellt eine Befehlssteuerung dar, die dazu dient, eine oder mehrere (bis zu 60) serielle Schnittstellen zu emulieren.” – (Quelle: <https://de.wikipedia.org/wiki/RFCOMM>, abgerufen: 05.10.19).

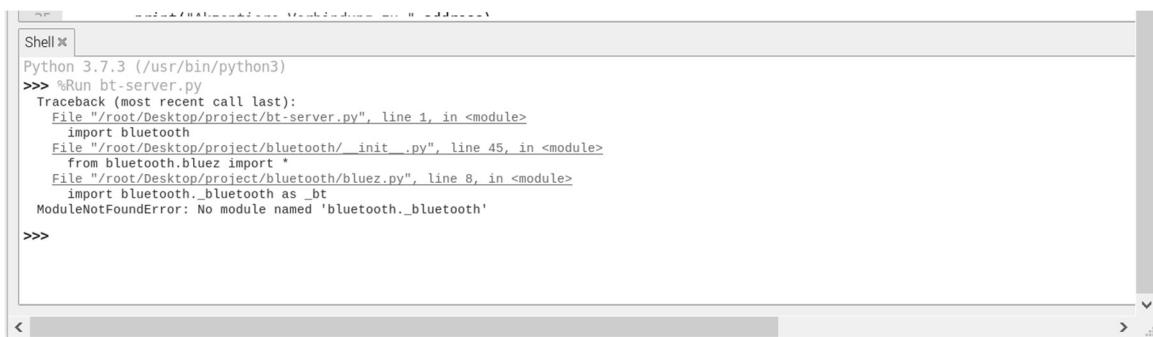
Dazu gibt es unter <http://people.csail.mit.edu/albert/bluez-intro/index.html> die Beispiele 3.2 und 3.3 von Albert Huang.

3.4.9 Software-Einrichtung

Sind alle Konfigurationen abgeschlossen, wird die gesamte Umgebung durch eine einzelne Server-Software und eine Client-Software getestet.

Wurde der Test-String erfolgreich übertragen, wird die erstellte Software gestartet.

Nach einem Neustart des Raspberry kann es zu unerwarteten Problemen kommen, indem eine Fehlermeldung innerhalb des Interpreters ausgeworfen wird, obwohl die Module zuvor problemlos liefen:



```
Shell >
Python 3.7.3 (/usr/bin/python3)
>>> %Run bt-server.py
Traceback (most recent call last):
  File "/root/Desktop/project/bt-server.py", line 1, in <module>
    import bluetooth
  File "/root/Desktop/project/bluetooth/_init_.py", line 45, in <module>
    from bluetooth.bluez import *
  File "/root/Desktop/project/bluetooth/bluez.py", line 8, in <module>
    import bluetooth._bluetooth as _bt
ModuleNotFoundError: No module named 'bluetooth._bluetooth'
>>>
```

Abbildung 3.2: Bluetooth-Modulfehler in der Raspi-Umgebung

Demnach kann eine zuvor laufende Bluetooth-Bibliothek nicht mehr gestartet werden. Abhilfe wird geschaffen, indem der installierte Ordner /bluetooth/ umbenannt wird. Innerhalb des Projekts wurde dieser in /bluetoothOnPi/ benannt.

Um dem Anwender den Betrieb grundsätzlich zu erleichtern, wird die Anwendung auf dem Mikrocomputer beim Hochlauf automatisch gestartet. Dazu muss eine Datei editiert werden, dazu wird in der Konsole eingegeben:

```
1.      sudo nano /etc/rc.local
```

Nachfolgende Zeile muss am Ende, jedoch vor `exit 0` eingetragen werden:

```
2.      python3 /home/pi/Desktop/project/main.py &
```

Es wird der Zielpfad mit der zu startenden Datei angegeben. Das Symbol „&“ wird zwecks potentieller Startprobleme eingetragen, um das Booten abzuschließen, auch wenn die Datei in einer Schleife bleibt.

3.4.10 Syntax

Die Parameter der eigenen Funktionen können durch eine eigens entwickelte Syntax beeinflusst werden. Sobald ein empfangener Datensatz mit den Zeichen „<#“ beginnt und mit „#>“ endet, wird von der Software überprüft, ob es sich um einen gültigen Befehl handelt.

Die Befehle dazu werden im Folgenden erläutert:

<#turns=9#>

Beeinflusst die Anzahl der Wiederholungen. Zahl muss zwischen 1 und 9 liegen. Wird eine Zahl kleiner 1 eingegeben, wird eine Wiederholung festgelegt. Wird eine Zahl größer 9 eingegeben, wird die maximale Anzahl auf 9 gestellt. Standard: 1. Ebenfalls wird bei unplausiblen Werten eine Wiederholung festgelegt.

<#dir=rtl#> oder <#dir=ltr#>

Der Parameter Direction wird angesprochen. Bei „rtl“ wird die Textrichtung „right-to-left“ festgelegt und sinngemäß bei „ltr“ die Richtung „left-to-right“. Standard: rtl.

<#inv=True#> oder <#inv=False#>

Bei dieser Option werden die Anzeigen der Buchstaben invertiert, also umgedreht. So kann eine Anzeige spiegelverkehrt ausgegeben werden. Standard: False.

<#speed=0.5#>

Hiermit wird die Laufgeschwindigkeit beeinflusst. Zu beachten ist, dass je kleiner der Wert, desto schneller die Geschwindigkeit, da hier mit einer Pause gearbeitet wird.

Das Parameterfenster bewegt sich zwischen 0.01 und 1.00. Werte größer 1 werden demnach auf das Maximum und Werte kleiner 0.01 auf das Minimum gestellt. Standard: 0.02.

<#default#>

Alle Werte werden auf Standard rückgesetzt.

<#time#>

Zeigt die aktuelle Zeit statisch an. Keine Laufschrift und solange, bis eine andere Nachricht oder ein anderer Befehl empfangen wird. Die Zeit ist nicht aktualisierend. Lediglich bei einem erneuten Aufruf wird die Zeit aktualisiert.

Bei unplausiblen Befehlen zeigt die Anzeige: „Unknown command“.

4 Inbetriebnahme

In diesem Kapitel wird näher auf das Starten der Software eingegangen und die Bedienung beschrieben.

4.1 Bedienung und Funktionen

Nach dem Einschalten der Spannungsversorgung startet der Raspberry Pi automatisch und öffnet nach erfolgreichem Hochlauf die benötigte Anwendersoftware.

Dies ist an dem Initialschriftzug „Technikerarbeit 2020“ erkennbar.

Um Daten an den Mikrocomputer zu senden, muss das in der Konfiguration angegebene Gerät mit der entsprechenden Software gestartet werden. Anschließend kann der Nutzer per Eingabe Zeichenfolgen senden, die dann an den Matrixelementen ausgegeben werden.

Zu Testzwecken wurde in der Arbeit als Client ein Notebook mit der im Anhang befindlichen Bluetooth-Client-Software genutzt.

Um dem Nutzer das Starten zu erleichtern, wurde eigens dazu eine Anwendung aus der Python-Quelldatei generiert.

```
#####
# Welcome to BT-Console for RaspberryPi-Project
#####
Nachricht: Message to Send...
```

Abbildung 4.1: Startbild BT-Client v1.0

Im anhängenden Datenträger, unter ..\03_Software_PC/BT-Client v1.0.exe, kann diese Anwendung gestartet werden.

Danach können die im Vorfeld angeschlossenen Module durch Senden von Daten die entsprechenden Texte anzeigen.

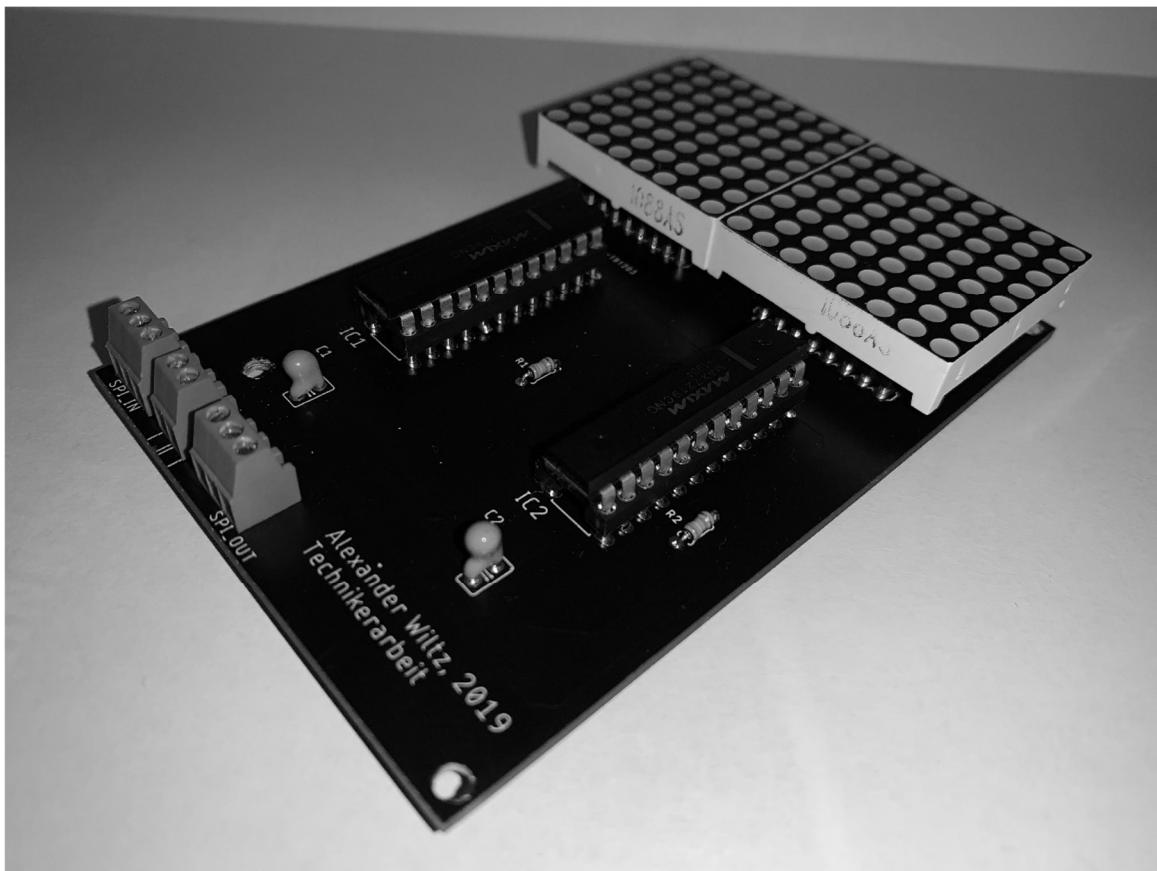


Abbildung 4.2: Matrix-Modul

5 Zusammenfassung

Nachfolgend werden alle Informationen und Ergebnisse nochmals zusammengetragen und beleuchtet.

5.1 Zeitplanung

Schritt	Tätigkeit	Zeitraum
Findungsphase	Ideensammlung für ein Projekt, Realisierbarkeit und Prüfung der Umsetzbarkeit mit vorhandenen Mitteln	03/2019
Planungsphase	Informationsbeschaffung über Möglichkeiten der Realisierbarkeit, Vor- und Nachteile zusammentragen, Festlegen der Komponenten	04/2019
Materialbeschaffung		04/2019
Platinenlayout planen und konstruieren	Schaltplan erstellen, Berechnung der Komponenten, Layout erstellen und Layout in Auftrag geben	06/2019
Erstellen der Software, Einarbeiten in Bibliotheksfunktionen	Programmierung der Module und der Abläufe für Bluetoothkommunikation und Ansprechen der SPI-Module	06/2019 - 08/2019
Testphase	Testphase der Kühlkörperberechnung, Messen der realen Ströme, Funktionalitäten und Verbesserungen im Ablauf	08/2019 – 10/2019
Dokumentation	Erstellen der finalen Arbeit	11/2019 – 12/2019

Tabelle 5.1: Zeitplanung

5.2 Resümee

Grundsätzlich kann die Arbeit als erfolgreich betrachtet werden. Es ist gelungen, in einem überschaubaren Rahmen eine kostengünstige Lösung zur Anzeige

beliebiger Zeichenketten auf LED-Matrix-Elementen ablaufen zu lassen, indem die Informationen via Bluetooth übertragen wurden.

Wird die Steuerung erweitert, kann problemlos an die vorgesehenen Pins ein weiterer Treiber angeschlossen werden. Innerhalb der Software muss lediglich der Parameter „cascaded“ angepasst werden, damit das zusätzliche Feld angesprochen wird. Zu beachten ist unter Umständen der maximal zulässige Strom der Raspberry Pins in Abhängigkeit der Gesamtgeräte.

Alles in allem ist der Aufwand zur Erstellung der Anwenderprogramme und die Inbetriebnahme des Mikrocomputers mit am größten.

Es hat sich gezeigt, dass es starke Abweichungen und Einschränkungen bei der Wahl des Clientgeräts gibt. Mit Applegeräten ist zum jetzigen Zeitpunkt nur schwer bis gar keine Verbindung möglich. Bei Android-Geräten (Handy und Tablet) konnte ein Bluetooth-Befehl nur beim Umgehen der Sicherheitsfunktionen innerhalb des Geräts aufgebaut werden.

Der einfachste und erfolgreichste Weg führte letztlich über eine eigene Software, die in der Arbeit beschrieben wurde. Damit konnten alle notwendigen Parameter und Funktionen selbst festgelegt werden.

5.3 Erweiterungsmöglichkeiten

Die hier aufgeführte Arbeit darf frei verwendet und nachgebaut werden, unter folgenden Bedingungen:

- Die Platinen bzw. Schaltungen dürfen in der hier aufgeführten Form nicht verkauft werden oder für kommerzielle Zwecke benutzt werden.
- Für wissenschaftliche oder technische Arbeiten unter Nennung des Autors und der Arbeit

Erweiterungsmöglichkeiten wären allen voran ein Wecker mit diversen Funktionen, wie Internetradio, diverse Wetterdaten auf den Displays, Spracherkennung und –steuerung, ein Python-Backend, um verschiedene Benutzerfunktionen einzusetzen, eine Weboberfläche zum Steuern und so weiter.

Quellenverzeichnis

- [WWW01]: http://gedankenlyrik.bplaced.net/www/studium/seminararbeit_2008.pdf; letzter Online-Abruf: 19.06.2019
Informationen SPI-I²C-Bus, „Seminararbeit Die Zweidrahtbussysteme I²C-Bus und SPI-Bus“
- [WWW02]: <https://docs.python.org/3/library/time.html>; letzter Online-Abruf: 28.06.2019
Dokumentation time-Bibliothek, “time – Time access and conversions”
- [WWW03]: <https://docs.python.org/3/library/datetime.html>; letzter Online-Abruf: 28.06.2019
Dokumentation datetime-Bibliothek, „datetime – Basic date and time types“
- [WWW04]: <https://docs.python.org/3/library/os.html>; letzter Online-Abruf: 26.06.2019
Dokumentation os-Bibliothek, „os-Miscellaneous operating system interfaces“
- [WWW05]: <https://github.com/coding-world/max7219>; letzter Online-Abruf: 28.06.2019
MAX7219 Bibliothek, „Raspberry Pi MAX7219 driver“
- [WWW06]: <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/images/gpio-pins-pi2.jpg>, GPIO Pins; letzter Online-Abruf: 19.06.2019
Abbildung 2.2, „Pin Header Raspberry“
- [WWW07]: <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/images/gpio-numbers-pi2.png>, GPIO Nummerierung; letzter Online-Abruf: 19.06.2019
Abbildung 2.3, „Pin-Belegung Raspberry“
- [WWW08]: <https://www.elektronik-kompendium.de/sites/raspberry-pi/1907101.htm>; letzter Online-Abruf: 24.11.2019
Abbildung 2.5, „Pin-Belegung RPi 3B“
- [WWW09]: www.netzmafia.de/skripten/hardware/RasPi/Projekt-MAX7219/index.html; letzter Online-Abruf: 19.06.2019

Abbildung 2.7, „Verbindungs-/Anschlusschema LEDs“

[WWW10]: <https://www.elektronik-kompendium.de/sites/raspberry-pi/1912111.htm>; letzter Online-Abruf: 24.11.2019

Abbildung 2.8, „Offizielle Stromverbrauchswerte“

[WWW11]: <http://people.csail.mit.edu/albert/bluez-intro/x232.html>; letzter Online-Abruf: 09.11.2019

Beispiele aus 3.5.8, „An Introduction to Bluetooth Programming“

Abbildungsverzeichnis

Abbildung 1.1: Matrix-Fertigmodul.....	4
Abbildung 2.1: Blockschaltbild	5
Abbildung 2.2: Pin-Header Raspberry	7
Abbildung 2.3: Pin-Belegung Raspberry mit GPIO-Bezeichnung	7
Abbildung 2.4: Pin-Belegung RPi 3B	8
Abbildung 2.5: Pin Belegung MAX7219 (Auszug aus dem Datenblatt)	9
Abbildung 2.6: Register Address Map MAX7219 (Auszug aus dem Datenblatt..	10
Abbildung 2.7: Verbindungs-/Anschlusschema LEDs	12
Abbildung 2.8: Ausgangsstrom am Segment.....	13
Abbildung 2.9: Offizielle Stromverbrauchswerte	14
Abbildung 3.1: Ablaufbeschreibung Bildschirmausgabe komplett	18
Abbildung 3.2: Bluetooth-Modulfehler in der Raspi-Umgebung	34
Abbildung 4.1: Startbild BT-Client v1.0	37
Abbildung 4.2: Matrix-Modul	38

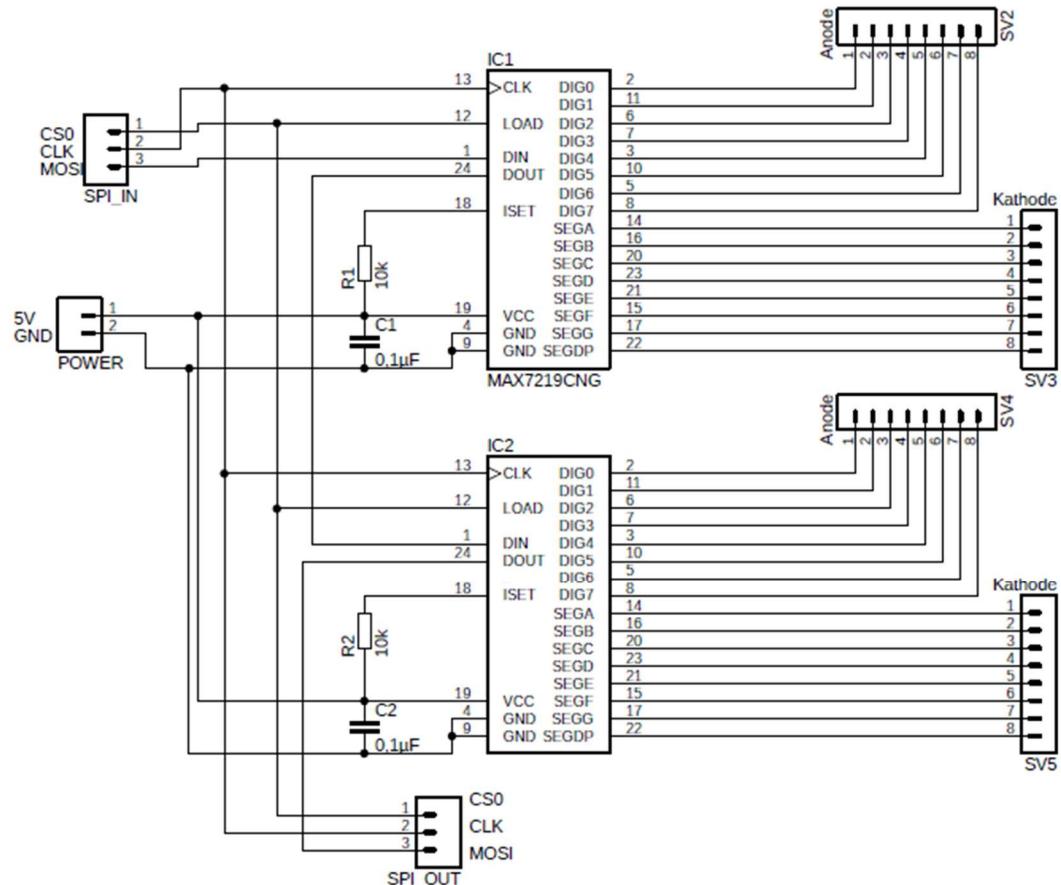
Tabellenverzeichnis

Tabelle 1.1: Gegenüberstellung SPI-I ² C-Bus.....	3
Tabelle 2.1: Belegung zw. RPi und MAX7219	8
Tabelle 2.2: Summe aller Ströme	14
Tabelle 2.3: Reale Messwerte	15
Tabelle 5.1: Zeitplanung	39
Tabelle A.3.1: Eingesetzte Software.....	64

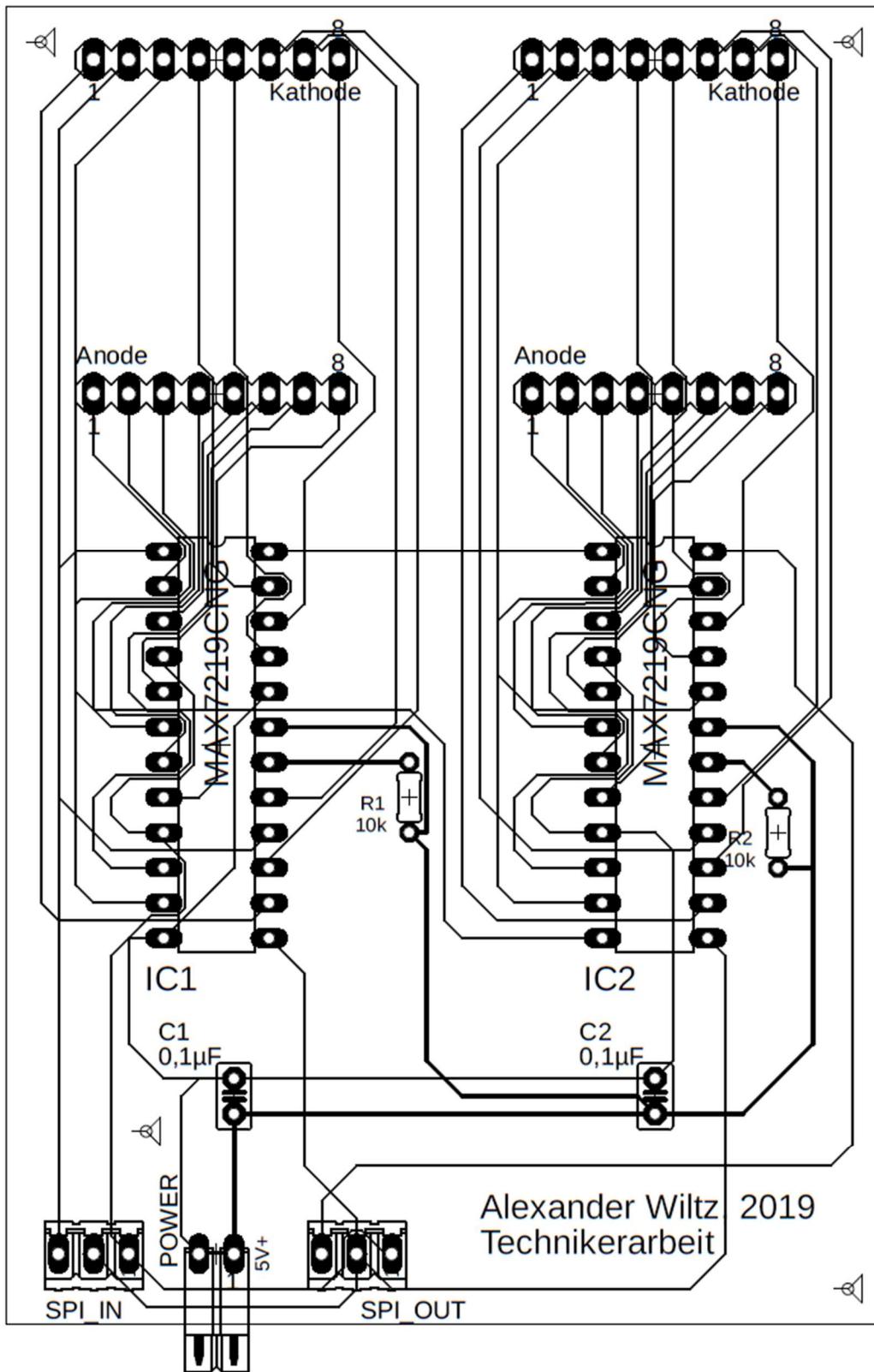
Anhang

A.1 Zeichnungen

A.1.1 Schaltplan LED-Treiber

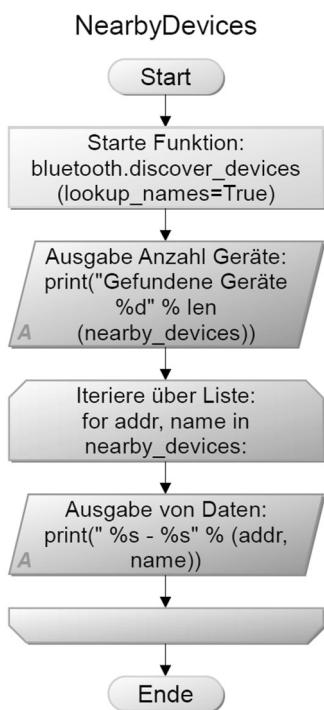


A.1.2 Layout LED-Treiber



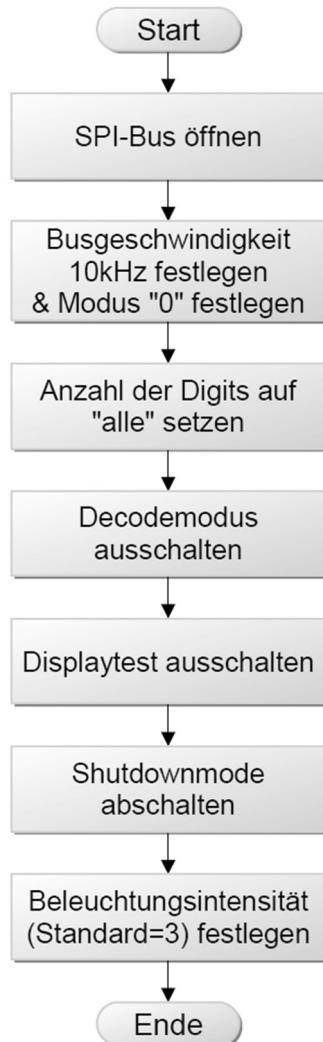
A.2 Code

A.2.1 Programmablaufplan Clientsoftware

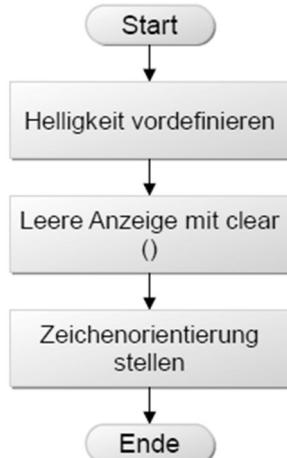


A.2.2 Programmablaufplan Raspberry

MAX7219-Init



MAX7219 paramterieren



Bluetooth-Server abfragen



A.2.3 Bluetooth-Clientsoftware

```
#!/usr/bin/python
#Alexander Wiltz, 2019

#<#code#>
#<#turns=999#> Anz WdH
#<#dir=ltr#><#dir=rtl#> Startrichtung
#<#inv=True#> Dreht Buchstaben
#<#speed=1#> Geschwindigkeit max=1 und min=0.01 (Pausenzeit)
#<#default#> Setzt alle Parameter auf Standard zurueck
#<#time#> Zeigt die aktuelle Uhrzeit an
'''

static_text(msg, invert=False, font=None)
Zeigt Text ohne Bewegung an (maximale Zeichenzahl = Anzahl Displays).

Dreht den Text um, wenn invert = True. Die Variable msg beinhaltet den anzuzeigenden Text. Für font ist ein alternativer Zeichensatz geplant, der allerdings noch nicht umgesetzt ist.

scroll_text(msg, direction=False, invert=False, speed=0.25,
turns=1, font=None)
Lässt einen in msg angegebenen Text über das Display laufen.
Veränderliche Parameter, wie direction, drehen die Startrichtung, invert dreht die Buchstaben, speed gibt die Scrollgeschwindigkeit an. Durch turns wird die Anzahl der Scrollwiederholungen festgelegt.

Font ist noch nicht umgesetzt.

'''

import bluetooth
import time

def nearby():
```

```

nearby_devices =
bluetooth.discover_devices(lookup_names=True)
print("Gefundene Geräte %d" % len(nearby_devices))
for addr, name in nearby_devices:
    print("%s - %s" % (addr, name))

def findDevice(target_adress):
    target_name = "Added Device"
    #target_adress = "B8:27:EB:51:6E:53" #MAC-Adresse
Raspberry
    nearby_devices = bluetooth.discover_devices()
    for bdaddr in nearby_devices:
        if target_name == bluetooth.lookup_name( bdaddr ):
            target_address = bdaddr
            break
    if target_address is not None:
        print("Gerät mit der Adresse", target_address,
"gefunden.")
    else:
        print("Kein Gerät mit angegebener MAC-Adresse in
Reichweite gefunden.")

def startBTClient(target_adress, msg):
    #target_adress = "B8:27:EB:51:6E:53" #MAC-Adresse
Raspberry
    #msg = "Hallo Raspberry" #Message an Pi
    port = 1
    sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
    sock.connect((target_adress,port))
    sock.send(msg)
    sock.close()

#####
#####
#####
```

```

msg = ""
oldMSG =""
target_adress = "B8:27:EB:51:6E:53"

print("#####",time.
sleep(0.2)
print("Welcome to BT-Console for RaspberryPi-
Project"),time.sleep(0.2)
print("#####",time.
sleep(0.2)

while True:
    msg = str(input("Nachricht: "))
    if msg != oldMSG:
        startBTClient(target_adress, msg)
        oldMSG = msg
    else:
        continue

```

A.2.4 Anwender-Software

```
#!/usr/bin/env python
# -----
-----
# Technikerarbeit Elektrotechnik
# Alexander Wiltz, EW16
# -----
-----
# Thema:
# RaspberryPi mit DOT-Matrix-Anzeigen betrieben mittels
MAX7219
# ueber SPI-Bus
# -----
-----
#Import der noetigen Bibliotheken
import max7219.led
from max7219.font import DEFAULT_FONT, LCD_FONT, UKR_FONT,
TINY_FONT
import time
import spidev

import bluetooth
import time

import datetime

#Zeitfunktionen
def getActualTime():
    #Zeit nehmen und splitten
    istime = datetime.datetime.now()
    stunden = istime.hour
    minuten = istime.minute
    sekunden = istime.second
```

```

#Führende Nullen für die Uhrzeit und wandeln in String
if stunden < 10:
    stunden = "0" + str(stunden)
if Minuten < 10:
    Minuten = "0" + str(Minuten)
if Sekunden < 10:
    Sekunden = "0" + str(Sekunden)

return str(stunden), str(minuten), str(sekunden)

#Funktionen fuer die Bluetooth-Server
def startBTserverOnce():
    server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
    port = 1
    server_sock.bind(("" ,port))
    server_sock.listen(1)
    client_sock,address = server_sock.accept()
    print("Akzeptiere Verbindung zu ",address[0])

    data = client_sock.recv(1024)
    print("empfange [%s]" % data)

    client_sock.close()
    server_sock.close()

def startBTServer():
    server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
    port = 1
    server_sock.bind(("" ,port))
    server_sock.listen(1)
    client_sock,address = server_sock.accept()

```

```

print("Akzeptiere Verbindung zu ", address[0])
data = client_sock.recv(1024)
return data
time.sleep(0.02)

def nearby():
    nearby_devices =
bluetooth.discover_devices(lookup_names=True)
print("Gefundene Geräte %d" % len(nearby_devices))
for addr, name in nearby_devices:
    print("  %s - %s" % (addr, name))

def findDevice(target_adress):
    target_name = "Added Device"
    nearby_devices = bluetooth.discover_devices()
    for bdaddr in nearby_devices:
        if target_name == bluetooth.lookup_name( bdaddr ):
            target_address = bdaddr
            break
    if target_address is not None:
        print("Gerät mit der Adresse", target_address,
"gefunden.")
    else:
        print("Kein Gerät mit angegebener MAC-Adresse in
Reichweite gefunden.")

#Hilfsvariablen
oldData = ""

#Definition bzw Start des SPI-Bus
spi = spidev.SpiDev()

#Paramterdefinition
font=LCD_FONT

```

```

cascaded = 8
brightness = 1

def initDisplay(cascaded, brightness):
    #Funktion zum Initialisieren (siehe Datenblatt MAX7219)
    #Standardfunktion aus max7219 Bibliothek Problembehaftet
    bus = 0
    device = 0
    #Verbindung aufbauen
    spi.open(bus, device)
    #Taktfrequenz Bus definieren und Modus festlegen
    spi.max_speed_hz = 100000
    spi.mode = 0
    #Initschritte
    spi.xfer2([0xB,7]*cascaded) #Benutze alle Digits
    spi.xfer2([0x9,0]*cascaded) #Decodemode aus
    spi.xfer2([0xF,0]*cascaded) #Displaytest aus
    spi.xfer2([0xC,1]*cascaded) #Shutdownmode aus
    spi.xfer2([0xA,brightness]*cascaded) #Intensitaet 0-15
    (Standard =3)

def brightnessPulses(cascaded, turns=1, endless=False):
    #Pulsiert Anzeige in Abhaengigkeit von 'turns', oder
    dauerhaft wenn 'endless=True'
    assert cascaded > 1 or cascaded==None, "Anzahl der
    angegebenen Module ueberpruefen"

    a = 0
    while endless or a in range(turns):
        for i in range(0,16):
            spi.xfer2([0xA,i]*cascaded)
            time.sleep(0.05)
        for i in range(0,16):
            i = 15-i

```

```

        spi.xfer2([0xA,i]*cascaded)
        time.sleep(0.05)

        a += 1


def static_text(msg, invert=False, font=None):
    #Zeigt Text ohne Bewegung an
    #Invertiert Text bei 'invert = True'
    assert len(msg) > 0, "Es wurde kein Text angegeben"

    try:
        #Beschneide String bei Ueberlaenge, sonst Fehler (zu
        wenig Geraete!)
        if len(msg) > cascaded:
            msg = msg[:-len(msg)-cascaded]
    except:
        None
    for i,j in enumerate(msg):
        if len(msg) < cascaded:
            offset = len(msg) - cascaded + 1
        elif len(msg) == cascaded:
            offset = 1
        if invert == False:
            i = len(msg) - i - offset
        device.letter(i, ord(j), font)

def scroll_text(msg, direction=False, invert=False,
speed=0.25, turns=1, font=None):
    #Scrollt einen Text durch die Displays
    #'direction = False', scrollt von rechts nach links
    #'direction = True', scrollt von links nach rechts
    #'invert = True', dreht den Text um
    #'speed' beeinflusst die Scrollgeschwindigkeit in
    Sekunden/Wechsel
    #'turns' bestimmt die Anzahl der Wiederholungen

```

```

assert len(msg) > 0, "Es wurde kein Text zum Scrollen
angegeben"

tmp = list()
x = 0
z = 0

while z in range(turns):
    for i in msg:
        tmp.append(i)
    if invert == False and direction == True:
        tmp.reverse()
    elif direction == False and invert == True:
        tmp = [" "]*cascaded-1 + tmp
        tmp.reverse()
    elif direction == False and invert == False:
        tmp = tmp + [" "]*cascaded-1
    tmp = [" "]*cascaded-1 + tmp
    while x in range(len(tmp)):
        if len(tmp) >= cascaded:
            maxRange = cascaded
        else:
            maxRange = len(tmp)
        if direction == True:
            for i in range(maxRange):
                device.letter(i,ord(tmp[i]), font)
            if maxRange <= (cascaded-1):
                i = maxRange
                device.clear(i)
        else:
            k = cascaded - 1
            for i in range(maxRange):
                device.letter(k-i,ord(tmp[i]), font)
            if maxRange <= cascaded-1:
                k = maxRange

```

```

        device.clear(k)

    try:
        del(tmp[0])
    except:
        None

        time.sleep(speed)
device.clear()
z += 1

#Treiber initialisieren vor Beginn der Anzeigen
initDisplay(cascaded, brightness)
#Funktionen aus max7219.led Lib
device = max7219.led.matrix(cascaded) #Klassendefinition als
device
device.brightness(brightness) #Helligkeit vordefinieren
(ueberschreibt aus max7219.led lib)
device.clear() #Anzeige komplett leeren
device.orientation(90) #Orientierung auf Standard setzen

msgInit = "Technikerarbeit 2020"
scroll_text(msgInit, direction=False, invert=False,
speed=0.05, turns=1, font=font)

#Standardwerte setzen
direction=False
invert=False
speed=0.15
turns=1

oldSekunden = ""
print("Start BT-Server...")
count = 0

```

```

while True:
    #Bluetooth-Server starten und auf neue Nachrichten
    warten
    data = startBTServer()

    if "<#" and "#>" in data.decode("utf-8"):
        codeStrLst = data.decode("utf-8").split("<#")
        codeStrLst = codeStrLst[1].split("#>")
        codeStr = codeStrLst[0]
        print("Parameter detected:", codeStr)
        if "turns" in codeStr:
            try:
                turns = int(codeStr.split("=")[1])
                if turns > 9:
                    turns = 9
                elif turns < 1:
                    turns = 1
            except:
                None
            msgInfo = "Set 'turns' to " + str(turns)
        elif "dir" in codeStr:
            dirCheck = codeStr.split("=")[1]
            if dirCheck == "ltr":
                direction = True
            elif dirCheck == "rtl":
                direction = False
            else:
                None
            msgInfo = "Set 'direction' to " + str(direction)
        elif "inv" in codeStr:
            invCheck = codeStr.split("=")[1]
            if invCheck == "True":
                invert = True

```

```

elif dirCheck == "False":
    invert = False

else:
    None

    msgInfo = str("Set 'invertation' to ", invert)

elif "speed" in codeStr:
    try:
        speed = float(codeStr.split("=")[1])
        if speed > 1:
            speed = 1
        elif speed < 0.05:
            speed = 0.05
    except:
        None
    msgInfo = "Set 'speed' to " + str(speed)

elif "default" in codeStr:
    direction=False
    invert=False
    speed=0.15
    turns=1
    msgInfo = "Set all to default"

elif "time" in codeStr:
    #Wechsel in Uhrzeitanzeige
    stunden, minuten, sekunden = getActualTime()
    msgTime = stunden + ":" + minuten + ":" +
sekunden
    static_text(msgTime, invert=False, font=None)
    time.sleep(0.02)

else:
    msgInfo = "Unknown command"

if not "time" in codeStr:
    print("Feedback:", msgInfo)

```

```
    scroll_text(msgInfo, direction=False,  
invert=False, speed=0.05, turns=1, font=font)  
  
elif data != oldData:  
    print("New message received...")  
    msg = str(data.decode("utf-8"))  
    scroll_text(msg, direction, invert, speed, turns,  
font=font)  
    oldData = data
```

A.3 Eingesetzte Software

Software	Link	Bezugsdatum
Thonny v3.2.1	http://thonny.org/	24.09.2019
Python IDLE v3.7.0	https://www.python.org/downloads/	05.07.2018
Notepad++ V7.5.6 x64	https://notepad-plus-plus.org/download/v7.5.6.html	23.09.2018
AutoDesk Eagle v9.4.2	https://www.autodesk.de	19.06.2019
PAPDesig ner v2.2.08	http://friedrich-folkmann.de/papdesigner/Hauptseite.html	09.11.2019
PuTTy v0.71	https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html	16.04.2019
WinSCP v5.15.1	https://winscp.net/eng/download.php	26.09.2019
Remote- desktop- verbindung	(Windows Standard)	
Win32 DiskImage r v1.0	https://sourceforge.net/projects/win32diskimager/	23.09.2018
NPPExport v0.3.0	https://sourceforge.net/projects/npp-plugins/	10.11.2019
Pyinstaller 3.5	https://www.pyinstaller.org/	10.11.2019
Raspbian	https://www.raspberrypi.org/downloads/raspbian/	26.09.2019
Python v3.7.4 RPi	https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tar.xz	08.07.2019
Thonny v3.2 RPi	enthalten	26.09.2019
XRDP		26.09.2019
MAX7219 Lib	https://github.com/coding-world/max7219	05.07.2019
PyBluez Lib	https://github.com/pybluez/pybluez	09.11.2019

Tabelle A.3.1: Eingesetzte Software