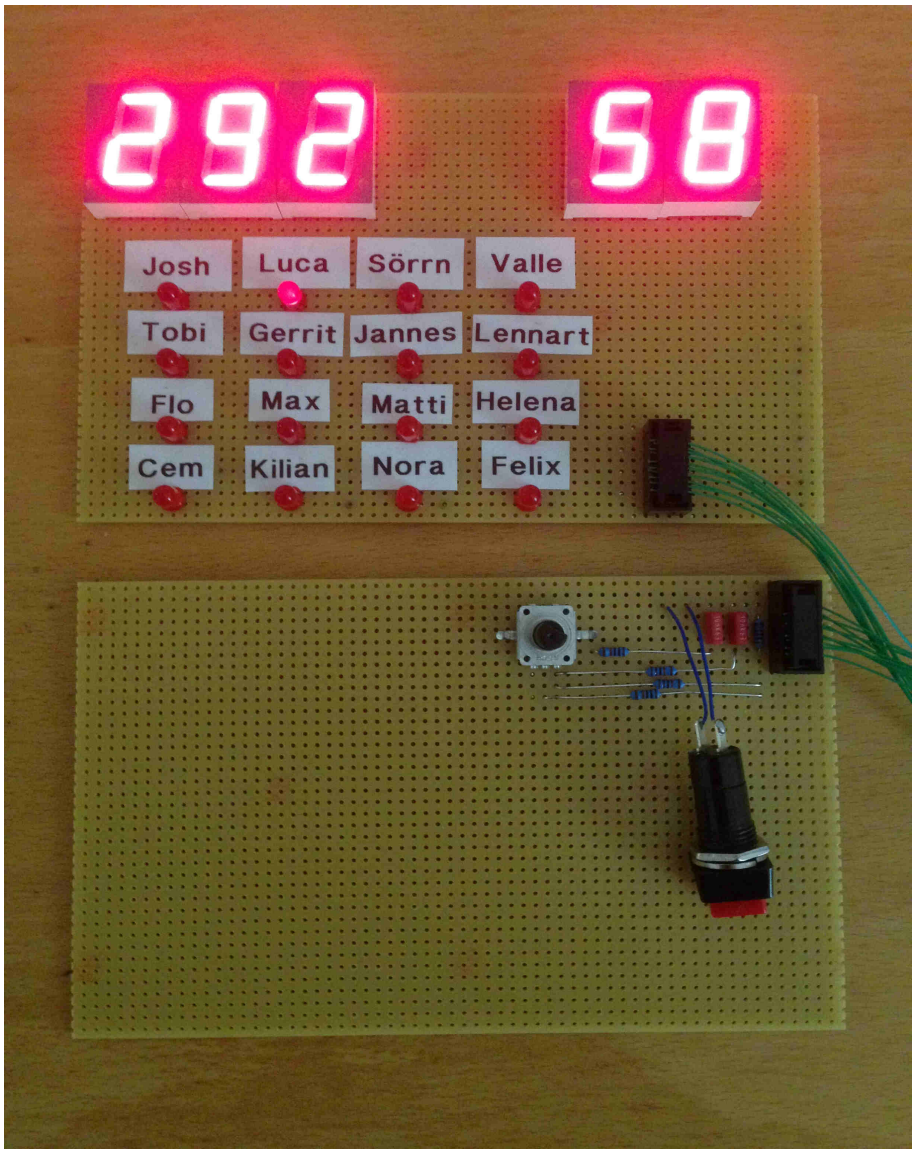
Dokumentation zum Seminar "Mikrocontroller – Realisierung in Hard- und Software"

Projekt
Digitale Getränkeliste



Valentin Resapow (21598712)
& Luca Zolldann (21597136)

Motivation & Ausgangssituation

Die Idee zur Erstellung eines digitalen Getränkezählers entwickelte sich bei einem abendlichen Kaltgetränk in einer universitätsnahen Wohngemeinschaft. Da hier genügend Platz für mehrere Besucher zur Verfügung stand sich diese Location dadurch in den letzten Monaten zu einem festen Treffpunkt entwickelte, wurde nach einer Lösung für eine faire Kostenaufteilung gesucht und zu diesem Zweck eine Liste für den Getränkeverzehr in Papierform angelegt. Als diese zusehends ihren physischen Grenzen entgegensteuerte, wurden Überlegungen zum weiteren Umgang beginnend bei einfacher Erweiterung um ein weiteres Blatt, bis hin zu digitalen Versionen in gängigem Tabellenformat angestellt. Leider konnte hierbei keine zufriedenstellende Lösung gefunden werden und das anstehende Projekt zum Mikrocontroller erwies sich als ideale Gelegenheit. Der niedrige Energieverbrauch, die niedrigen Kosten und die dauerhafte Stationierung sind unbestreitbare Mehrwerte gegenüber einer Tabellenlösung, für welche ein vollwertiger Rechner oder vergleichbare Endgeräte benötigt würden. Außerdem war es nach Ansicht der Entwickler ein nicht tragbarer Zustand in einer Wohngemeinschaft von Technikbegeisterten eine anloge Papierliste zu führen. Die bisherige Handhabung hinter den reinen Zahlen wurde unverändert für die digitale Variante übernommen: Jedes verzehrte Getränk wird im Zählerstand Gesamtdrinks vermerkt, wobei 25 Drinks - in Anlehnung an gängige Getränkekästen - eine planungstechnische Einheit darstellen. Sobald man eine 25-er Einheit im Zähler gefüllt hat, oder sich auch schon vorher dazu berufen fühlt, ist man an der Reihe für Nachschub zu sorgen, wodurch dann eine Einheit als beigesteuert markiert wird. Die Abweichung zum genauen Getränkebestand ist dabei bekannt und durchaus gewollt, da diese Dokumentation nicht dazu führen soll, dass jedes Getränk bezahlt und somit indirekt mit einem Preisschild versehen wird. Vielmehr soll sie helfen, einen groben Überblick zu behalten und den regelmäßigen Besuchern ein Anstoß zum gelegentlichen Beisteuern eines Beitrags sein.

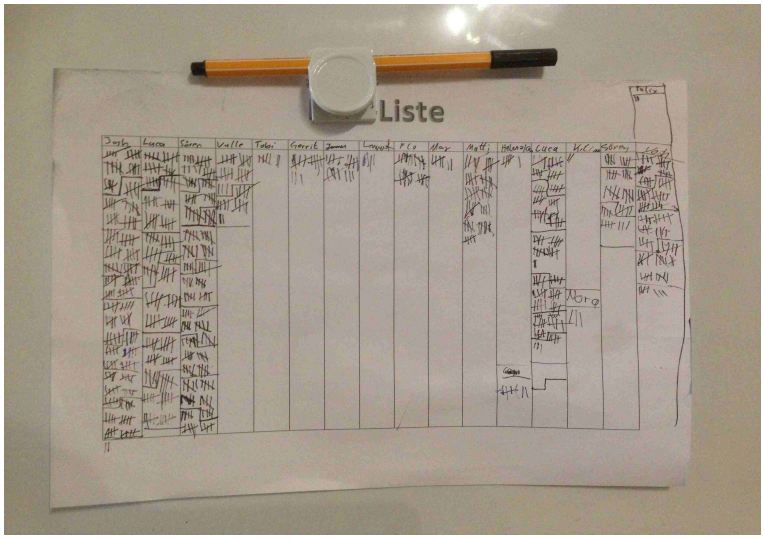


Abbildung 1: bisher geführte Liste im Papierformat

Bedienung:

Das Gerät ist mithilfe eines handelsüblichen Netzteils zum bespielsweisen Laden von Handys und einem USB-C-Kabel mit eine Verorgungsspannung von 5V zu betreiben. Zur Initialisierung wird über die fünf Sieben-Segment-Anzeigen der Schriftzug "Hello" angezeigt und die 16 Auswahl-LED's werden nacheinander mit kurzer Pause aktiviert. Sobald diese vollständig leuchten, ist die Initialisierung abgeschlossen und der Energiesparmodus wird aktiviert. In diesem reagiert die Schaltung lediglich auf Drehungen des Inkrementalgebers. Dieser wird im Folgenden dazu verwendet, den gewünschten Benutzer auszuwählen, wobei die drei linken Sieben-Segment-Anzeigen jeweils den Gesamtstand der verzehrten Getränke wiedergeben und die zwei rechten Sieben-Segment-Anzeigen den Abstand zu den bislang mitgebrachten. Hierbei leuchten die Dezimalpunkte und die blaue LED auf dem PCB mit auf, sofern der Stand des Getränke-Zählers den Stand des Bezahlt-Zählers übersteigt. Diese Anzeige bleibt für einige Sekunden bestehen und schaltet danach bei fehlender weiterer Interaktion wieder in den Energiesparmodus. Zum Erhöhen des Getränke-Zählers wird bei aktiver Anzeige der Taster des Inkrementalgebers kurz betätigt. Zur Eintragung einer mitgebrachten Getränkekiste o.ä. wird der zusätzliche Taster bei aktiver Anzeige kurz betätigt, was den Bezahlt-Zähler um 25 erhöht. Wurde ein Taster irrtümlicherweise betätigt, können die Zählerstände durch einen längeren Druck der entsprechenden Taster um eins bzw. 25 vermindert werden. Sollte einer der Zählerstände dabei unter null fallen, wird kurz "Err 00" angezeigt und die Aktion nicht ausgeführt. Ein weiterer Fehler entsteht, wenn die Anzeigen 999 bzw. 99 übersteigen. In diesem Fall wird "Err 01" angezeigt und der Zählerstand muss manuell vom Nutzer durch die Taster korrigiert werden. Weiterhin symbolisiert die rote LED auf dem PCB das Vorliegen eines Problems, die grüne LED den aktivierten Zustand und das Erwarten weiterer Eingaben und die blaue LED die bereits beschriebene Überschreitung des persöhnlichen Bezahlt-Zählers.

Abbildung 2: Ausgabemodul mit fünf Sieben-Segment-Anzeigen und 16 LED's

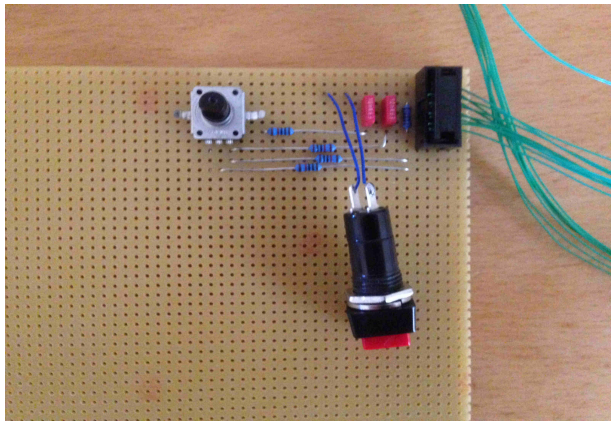


Abbildung 3: Eingabemodul mit Drehinkrementalgeber inklusive Taster und einzelnem Taster

Umsetzung:

Aufbau:
Der Gesamtzähler ist in drei Einzelmodule unterteilt, um eine platzsparende Verstauung und flexible Anordnung zu gewährleisten. Außerdem konnte dadurch bei der Entwicklung der Fokus besser auf die jeweiligen Einzelteilen gelegt werden. Die Module sind thematisch nach Ausgabe, Eingabe und Berechnung aufgeteilt und untereinander über Kabel verbunden. Dabei sind die Aus- und Eingabe-Module als Lochrasterplatinen umgesetzt und das Berechnungsmodul als printed circuit board. Die Bauteile auf den Lochrasterplatinen sind via Durchsteckmontage befestigt, während die Bauteile auf dem PCB über SMD-Technologie montiert sind. Dieser Aufbau wurde weiterhin gewählt, damit ein später geplantes Einfügen in ein Gehäuse einfacher umgesetzt werden kann.

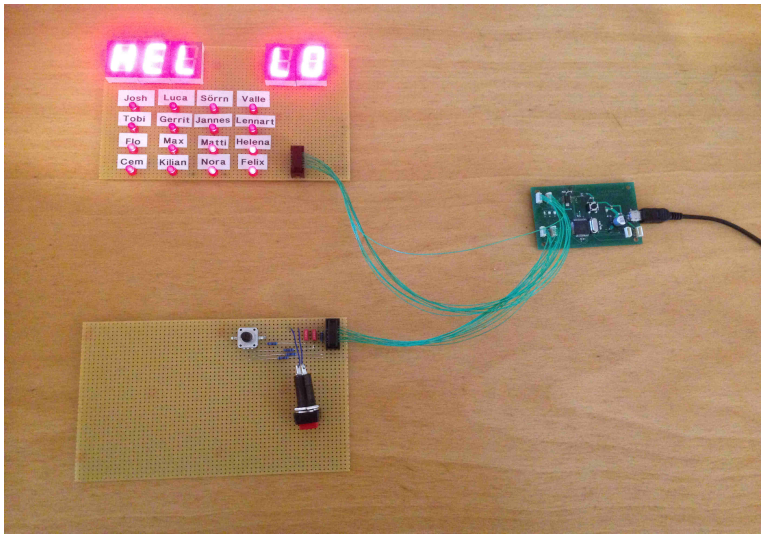


Abbildung 4: Gesamtansicht aller Module

Speicher:
Die Zählerstände sind sämtlich im EEPROM gespeichert, um einen Verlust auch bei Unterbrechung der Versorgungsspannung zu vermeiden. Die Routinen zum Lesen und Schreiben sind derart gestaltet, dass sie jeweils zwei Byte an der Adresse des Y-Pointers in die Register XH & XL schreiben bzw. umgekehrt und den Y-Pointer auf das folgende Byte erhöhen. Die Register R26 und R27 werden in der vorliegenden Anwendung als zusammenhängende Werte-Register mit zwei Byte genutzt und nicht als Pointer. Die Nutzung des EEPROM ist derart strukturiert, dass für jeden Benutzer 16 zusammenhängende Byte reserviert sind. In den ersten acht Byte ist der jeweilige Name im ASCII Format gespeichert. Er wird aktuell weder angezeigt noch kann er ohne Programmierschnittstelle geändert werden. Nichts desto trotz kann er zum späteren Nachvollziehen der Zählerstände wertvoll werden. Die Byte 9 bis 12 sind bisher ungenutzt und können später für weitere Informationen verwendet werden. Die Byte 13 und 14 enthalten schließlich den Zählerstand der gesamt verzehrten Getränke und die Byte 15 und 16 den Zählerstand der beigesteuerten Getränke.

Programmablauf:
Nach dem Reset oder bei erstmaligem Anschluss an die Versorgungsspannung durchläuft der Mikrocontroller die Initialisierungroutine, während welcher einige Register entsprechend der Anforderungen gesetzt, die Routinen zur Anzeige von "Hello" und der Aktivierung der LED's aufgerufen (für Details siehe "Ausgabe") und zuletzt die Interrupts und der Idle-Modus aktiviert werden. In diesem verbleibt der Mikrocontroller solange keine Interaktion über die Eingabegeräte erfolgt, also im ungenutzten Fall. Nach der Aktivierung über einen Interrupt und Ausführung der zunächst anstehenden Routinen (für Details siehe "Eingabe") geht der Programmablauf in den Abschnitt "wait" über, während dem die Anzeigen aktiv bleiben und die Input-Pins der beiden Taster für Dinks (PINB, 1) und Bezahlung (PIND, 5) in einer Schleife von ca. 1 Millionen (16x256x256) Durchläufen abgefragt werden (polling). Sollte während dieser Zeit keine Interaktion erfolgen, werden alle Anzeigen deaktiviert die Interrupts aktiviert und in den Idle-Modus umgeschaltet. Dieser Programmabschnitt ist nicht als Unterprogramm, sondern vielmehr als Abzweigung zu verstehen. Er wird jeweils über den "rjmp"-Befehl aufgerufen, was die Schleifenzähler nach jedem Interrupt oder Tasterdruck von Neuem beginnen lässt.

Eingabe:
Die Eingabe erfolgt nach zwei unterschiedlichen Prinzipien: der Drehinkrementalgeber steuert die zwei Interrupts INT0 (PIND, 2) und INT1 (PIND, 3), die so initialisiert sind, dass sie auf eine fallende Flanke reagieren, um ein Deaktivieren des Idle-Modus zu gewährleisten, wohingegen die beiden Taster normale I/O-Pins (PIND, 5 bzw. PINB, 1) ansteuern und ständig abgefragt werden müssen (polling). Alle vier Eingabemöglichkeiten ziehen den entsprechenden Anschluss bei Aktivierung auf das Masse-Potential (pull-low). Die Besonderheit des Drehinkrementalgebers besteht darin, dass er bei Aktivierung unabhängig von der Richtung jeweils beide Anschlüsse kurzzeitig auf das Masse-Potential zieht (Abb. 5 & 6). Somit kann nicht die intuitive Handhabung, bei welcher jeder Interrupt für eine Drehrichtung steht, genutzt werden, weil dabei ein ständiges Vor- und gleichzeitig wieder Zurückschalten ausgelöst würde. Stattdessen wird ein weiterer Zwischenschritt im Interrupt-Handler verwendet, der nach Aktivierung den jeweils anderen Interrupt-Pin abfragt und erst in die Routine zum Drehen übergibt, wenn dieser auch auf logisch null liegt, also beim ersten Interrupt eine Art "Fehlalarm" auslöst und erst bei einer vollständigen Drehung eine sichtbare Reaktion zeigt. Damit ist auch ein weiteres Problem beseitigt, das während der Lösungsfindung häufig auftrat. Wegen mechanischer Reibung kommt es oft dazu, dass der Inkrementalgeber auf einer Seite hängen bleibt und einer der Eingänge dauhaft auf das Masse-Potential gezogen wird. Dies führte bei einem früheren Lösungsansatz, bei dem die Interrupts auf das dauerhaft niedrige Potential getriggert waren, zu einer wiederkehrenden Aktivierung der Interrupts und damit Aktivität im Mikrocontroller. Auch dieser Umstand ist in der letztlichen Lösung behoben, da in diesem Fall ein "Fehlalarm" ausgelöst wird, der Mikrokontroller danach aber dauerhaft im Idle-Modus verbleibt.
Bei erfolgter vollständiger Drehung wird in der entsprechenden Routine zunächst der Z-Pointer erhöht bzw. verringert um den Bereich im EEPROM des entsprechenden Benutzer zu adressieren (für Details siehe "Speicher"). Danach erfolgt eine Überprüfung anhand der grünen LED (PORTC, 6), ob vorher der Idle-Modus aktiviert war, was zum Zurücksetzen des Pointers auf den Beginn führen würde. Im Anschluss wird noch die Routine zur Anzeige über die Sieben-Segment-Anzeigen aufgerufen (für Details siehe "Ausgabe") und in die "wait" Schleife (für Details siehe "Programmablauf") übergeben. Sollte in dieser eine weitere Drehung erfolgen, wird wieder das beschriebene Prozedere ausgeführt. Sollte allerdings einer der Taster (PINB, 1 bzw. PinD, 5) betätigt werden, wird in die Routinen "drink" bzw. "pay" übergeben. Während dieser wird zunächst der aktuelle Wert geladen und in ca. 130 Tausend Zyklen überprüft, ob der entsprechende Taster

weiterhin aktiviert ist. Bei nur kurzer Aktivierung wird der Zählerstand erhöht, gespeichert und erneut angezeigt. Bei Aktivierung während aller Zyklen wird der Zählerstand verringert und anschließend überprüft, ob er null nicht unterschreitet. Dies würde in den Error-Handler 0 überführen und die Verringerung nicht in den EEPROM übergeben. Zum Abschluss wird bei erfolgter Verringerung noch eine Schleife durchlaufen, die andauert bis der Taster wieder deaktiviert wird und jeweils wieder in die "wait"-Routine übergeben.

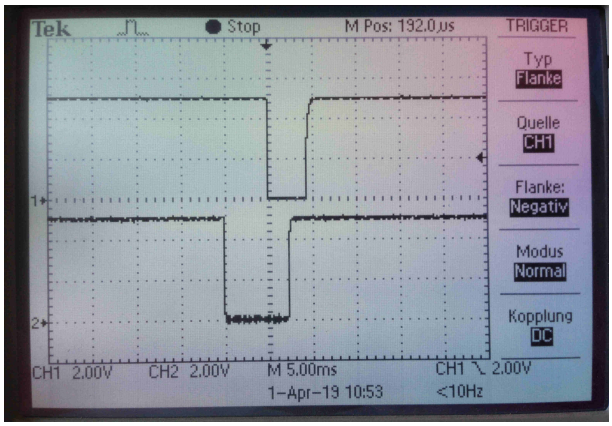


Abbildung 5: Messung via Oszilloskop nach Rechtsdrehung
Kanal 1: PIN D, 2 (Interrupt 0)
Kanal 2: PIN D, 3 (Interrupt 1)

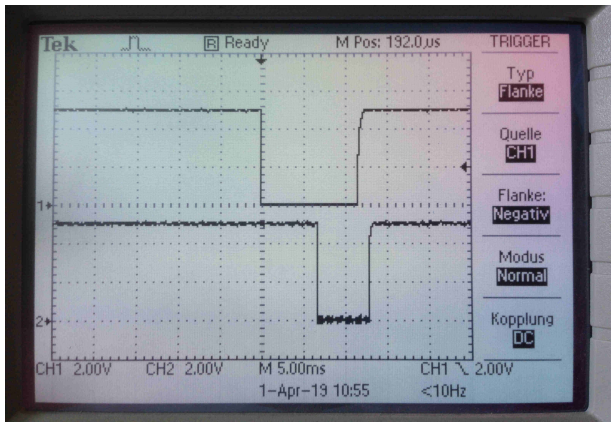


Abbildung 6: Messung via Oszilloskop nach Linksdrehung
Kanal 1: PIN D, 2 (Interrupt 0)
Kanal 2: PIN D, 3 (Interrupt 1)

Ausgabe:

Das Kernstück der Ausgabe ist ein integrierte Schaltkreis (IC) vom Typ "MAX6951". Dieser wird typischerweise über ein Serial Peripheral Interface (SPI) mit Befehlen von 16 Bit angesteuert. Da dieser Umstand während des Platinendesigns allerdings übersehen wurde, musste eine Befehlsübertragung über die normale Funktionalität der I/O-Pins realisiert werden. Über drei Pins wird eine SPI-Schnittstelle emuliert, indem die nötigen Übertragungsleitungen für die Auswahl (PORTC, 4), die Daten (PORTC, 1) und die Clock (PORTC, 0) direkt angesteuert werden. Dies funktioniert nach anfänglichen Schwierigkeiten sehr zuverlässig und die gemessenen Potentialverläufe weisen auch über die genutzten internen Pull-Up-Widerstände des Mikrocontrollers sehr saubere Flanken auf (Abb. 7). Die Implementierung hierzu erfolgt über die Subroutinen "send" und "clock", die in diversen Routinen mit verschiedenen Befehlen aufgerufen werden. Der IC steuert im weiteren Verlauf die fünf Sieben-Segment-Anzeigen sowie die 16 LED's über nacheinander geschaltete Impulse an und ermöglicht so deren Betrieb mit insgesamt 9 Leitungen. Weiterhin ist im IC eine wahlweise Decodierung integriert, wodurch die übermittelten Ziffern nicht vorher auf die einzelnen Segmente umgesetzt werden müssen, die LED's aber als einzelne Segmente betrieben werden können.
Programmseitig erfolgt die komplette Ausgabe über die Routine "show". Hierin wird zunächst die bisherige Anzeige gelöscht und die LED des ausgewählten Nutzers aktiviert, was in der Subroutine "LED" über eine Tabellenauswahl mit dem Z-Pointer geschieht. Anschließend wird der entsprechende Getränke-Zählerstand aus dem EEPROM gelesen (für Details siehe "Speicher") und sichergestellt, dass dieser 999 nicht übersteigt, was zu Error 01 überleiten würde. Nach der folgenden Umrechnung in Dezimalwerte werden diese nacheinander an den IC übersendet. Im nächsten Schritt wird der Bezahlt-Zählerstand geladen und der Abstand zum zwischengespeicherte

Getränke-Zählerstand gebildet. In der verwendeten Subroutine wird zunächst der höhere Wert festgestellt und dann der niedrigere abgezogen. Sollte der Getränke-Zählerstand den Bezahlt-Zählerstand überschreiten wird die blaue LED auf dem PCB aktiviert und im Folgenden auch die Dezimalpunkte der rechten beiden Sieben-Segment-Anzeigen. Nach Überprüfung der 99-Grenze wird auch der Abstand in Dezimalziffern umgerechnet und an den IC übersandt. Abschließend wird noch der Befehl zur Aktivierung der Anzeige über die Routine "wake" übersandt.

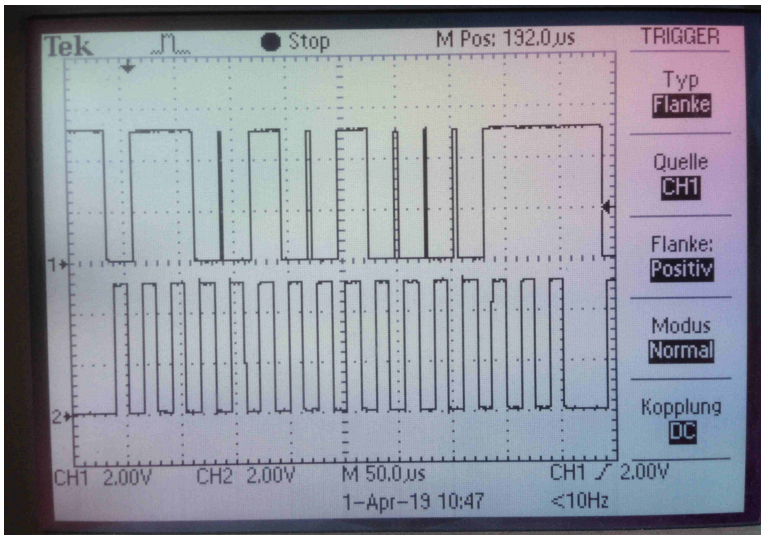


Abbildung 7: Messung via Oszilloskop an den Eingängen des "MAX6951"
während der Übertragung der Sequenz "0110010010000111"
Kanal 1: PIN C, 1 (Datenleitung); Kanal 2: PIN C, 0 (Clockleitung)

Auswertung:

Die Berechnungen zum Energieumsatz ergeben nach den mittleren Angaben aus den jeweiligen Datenblättern eine Leistungsaufnahme von jeweils rund 20 mW des Mikrocontrollers und 50 mW des IC im aktiven Zustand sowie 7,5 mW und 0,31 mW im jeweiligen Energiesparmodus. Da die Schaltung den Großteil der Zeit im deaktivierten Zustand verbringen wird und die Leistungsaufnahme dabei vergleichsweise gering ist (weniger als eine klassische LED im Betrieb), eignet sie sich für den dauerhaften Betrieb.

Quellenverzeichnis:

Datenblatt des Atmega16A:
http://ww1.microchip.com/downloads/en/devicedoc/atmel-8154-8-bit-avr-atmega16a_datasheet.pdf

Datenblatt des MAX6951:
https://datasheets.maximintegrated.com/en/ds/MAX6950-MAX6951.pdf

Datenblatt der Sieben-Segment-Anzeigen:
https://cdn-reichelt.de/documents/datenblatt/A500/SC08-11SRWA%28V6%29.pdf

Datenblatt des Drehinkrementalgebers:
https://cdn-reichelt.de/documents/datenblatt/F100/402097STEC12E08.pdf

Instruktionssatz für Assembler:
https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_instruction_list.html

Einführung zu Assemblerprogrammierung:
http://www.avr-asm-tutorial.net/avr_de/index.html