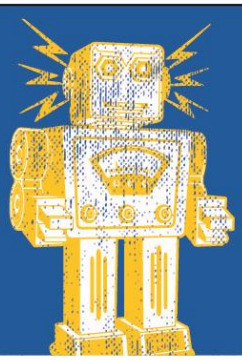
Informations- und Kommunikationssysteme



MicPos

Software Requirements Specification

Projektmitglieder: Andreas Mikula, Philip Albrecht, Daniel Schinewitz, Ibrahim Milli
Begleitender Lektor: Patrick Schmitt, BSc
BIC, 4. Semester, SS 2019

INHALTSVERZEICHNIS

1. Einführung

MicPos ist ein Embedded System Studenten Projekt, welches im Umfang der Lehrveranstaltung Embedded System Engineering umgesetzt wird. Im Allgemeinen dient die Projektausführung zur Absolvierung der genannten Lehrveranstaltung und zur Erreichung eines erweiterten Skills im Bereich des Embedded System Engineering.
Ziel der Projektumsetzung ist der Aufbau einer Client-Server Modulation, welche durch drei TIVA-TM4C1294XL und einem Raspberry Pi 2 Model B aufgebaut ist. Diese Modulation zeichnet über Mikrofone die Geräuschkulisse aus der Umgebung auf und berechnet den Ursprung der Geräuschkulisse, welche über ein Monitorboard ausgegeben werden soll.

1.1 Zweck

Der Zweck der Projektumsetzung ist die positive Absolvierung der Lehrveranstaltung Embedded System Engineering und zur Erweiterung der persönlichen Skills im selbigen Bereich.

1.2 Zielgruppe und Verwendungszweck

Die Zielgruppe umfasst das Publikum der Lehrveranstaltung, welche primär aus der Umsetzergruppe und dem begleitenden Lektor besteht. Final erweitert sich die Audience um jene Studienkollegen, welche an der Endpräsentation teilnehmen werden und möchten.
Anschließend an das Projekt dient der Umfang der Arbeit als Referenzmodell für weitere Umsetzungen in diesem Bereich und kann vor allem für spätere Teilnehmer dieser Lehrveranstaltung als Anhaltspunkt für eigene Umsetzungen herangezogen werden.
Eine zukünftige Erweiterung der Projektimplementierung ist zu diesem Zeitpunkt nicht angedacht.

1.3 Projekt Scope

Der Zweck des MicPos Projektes besteht darin, das es in verschiedensten Bereichen notwendig ist, Geräuschkulissen seinem Entstehungsursprung zuordnen zu können.

Ein Anwendungsbereich wäre die Installation einer Alarmanlage im Wohnbereich. Hier könnte bei einer Alarmauslösung das verteilte MicPos System den Ursprung der Alarmierung berechnen.

1.4 Projektstart

Der offizielle „Startschuss“ für das Projekt war der 11.02.2019, als der Projektvorschlag seitens des Lektors akzeptiert und freigegeben wurde.

1.5 Studiengang

Informations- und Kommunikationssysteme (BIC) an der
Fachhochschule Technikum Wien
Höchstädtplatz 6
1200 Wien

1.6 Lektor

Patrick Schmitt, BSc
Research & Development
Institut für Embedded System
peter.schmidt@technikum-wien.at

1.7 Entwickler / Durchführende Personen

Andreas Mikula
E-Mail: ic17b026@technikum-wien.at

Daniel Schinewitz
E-Mail: ic17b051@technikum-wien.at

Ibrahim Milli
E-Mail: ic17b063@technikum-wien.at

Philip Albrecht
E-Mail: ic16b043@technikum-wien.at

1.8 Definitionen und Akronyme

Dieses Dokument nutzt folgende Namenskonventionen.

TIVA	TIVA EK-TM4C1294XL
RASPI	Raspberry Pi 2 Model B
MIC	Adafruit MAX4466-Mikrofon
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver Transmitter

1.9 Referenzen

http://www.ti.com/lit/ug/spmu365c/spmu365c.pdf
https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-2-model-b.pdf
https://www.farnell.com/datasheets/1878224.pdf
https://cdn-shop.adafruit.com/datasheets/MAX4465-MAX4469.pdf

2. Allgemeine Beschreibung

Stellen Sie sich ein System vor, welches aus mehreren Mikrocontrollern besteht, die jeweils kabelgebunden mit einer Basisstation kommunizieren. Jeder µC ist mit einem Mikrofon verbunden um Audioinformationen zu sammeln. Diese µC-Karten werden in einem Raum so platziert, dass sie genau den gleichen Abstand voneinander haben. Wenn ein lautes Signal (wie ein Handklatschen) von einem der µCs erkannt wird, sollte die genaue Position der Quelle dieses Signals bestimmt werden. Durch Analyse der Signalstärke und des Timings jedes µC, der ein Rauschen erkennt, kann die Position berechnet werden. Dies ist ein sehr ähnlicher Ansatz wie das sogenannte "Richtmikrofon", welches auch von Sprachassistenten wie Alexa verwendet wird.

2.1 User Needs

In primärer Ordnung dient die aufgebaute Modulation reinen Test und LV-immanenten Zwecken im Umfang der Projektarbeit an der FH-Technikum. Indes sind User Needs nicht weitergehend berücksichtigt. Jedoch können der Programmcode und die grundlegende Idee von weiteren Studenten aufgegriffen werden und entsprechend erweitert bzw. verbessert werden.

2.2 Voraussetzung und Abhängigkeiten

Für die Umsetzung des Projekts bedarf es keiner gesonderten Voraussetzungen bzw. es sind keine Notwendigkeiten vorhergehender Implementierung zu berücksichtigen.
Um jedoch Nachvollziehbarkeit und zukünftige Erweiterungen garantieren zu können, bedarf es der korrekten Einsetzung der Hardware in Verbindung mit tiefergehendem Wissen des Embedded System Engineering.

3. Projektplanung

3.1 Projektphasen

Für die Projektplanung, -implementierung und -testung standen den Entwicklern rund 450-540 Personenstunden. Diese wurden in der Planungsphase an die teilnehmenden Personen verteilt und zur Durchführung gebracht.

Zusammenfassend können die einzelnen Durchführungsphasen wie folgt gegliedert und in ihre Aufwandsstunden geteilt werden:

Projektphase	**Geplanter Zeitaufwand**
Planungsphase	134 Personenstunden
Umsetzungsphase	275 Personenstunden
Testphase & Administratives	116 Personenstunden

3.1.1 Planungsphase (Initialisierung des Projekts)

Die Planungsphase ist gekennzeichnet durch die Formulierung vom Projektantrag, die Analyse von Wirtschaftlichkeit, Machbarkeit und Entwurf eines Projektdurchführungsplans in Hinsicht auf Struktur, Aufbau und Umsetzung.

Analysephase	
Projektthema finden	15 PS
Projektantrag formulierne	4 PS
Projektplanung und Zeiteinteilung	12 PS
Verteilung der Verantwortlichkeiten	8 PS
Projektdaten zusammentragen	7 PS
Start der Dokumentation	9 PS

Entwurfsphase	
Zielplattform festlegen	4 PS
Architekturdesign	6 PS
Hardwareauswahl und Bestellung	10 PS
Softwareauswahl	5 PS
Aktualisierung der Dokumentation	5 PS
Fortschrittsdarbietung in Präsenzphase	5 PS

Planungsphase	
Analysephase	40 PS
Entwurfsphase	35 PS
GESAMT	*75 PS*

3.1.2 Umsetzungsphase

In der angesetzten Phase der Umsetzung erfolgt das Einarbeiten in Datenblätter und Manual zu den ausgewählten Systemkomponenten, das Aufsetzen und Konfigurieren der Hardware inklusive Implementierung der Funktionalität und Entwicklung von Prototypen.
Zudem erfolgen iterative Testaufbauten und deren Auswertung, als auch die fortlaufende Dokumentation der Fortschritte, welche im Projektteil „Administration“ nähere Beachtung finden.

Implementierungsphase	
Studium der Datenblätter / Manuals	16 PS
Studium der Literatur	8 PS
Inbetriebnahme der Hardware	10 PS
Installation und Konfiguration der HW	8 PS
Familiarisierung mit HW / Zubehör	12 PS
Inbetriebnahme der Komponenten	5 PS
Konfiguration der Komponenten	15 PS
Installation der Software	10 PS
Konfiguration der Software	12 PS
Programmierung der Kommunikation	80 PS
Aktualisierung der Dokumentation	24 PS
Fortschrittsdarbietung in Präsenzphase	50 PS
	250

Umsetzungsphase	
Implementierungsphase	250 PS
GESAMT	*250 PS*

3.1.3 Testphase & Administration

Die Testphase beinhaltet den Abschluss der Umsetzungsphase, wie auch die Implementierung der Testcases (bestehend aus Unit Test, Integration Test, System Test & Exceptant Test). Zudem bildet diese die Konstruktion vom Endprodukt unter Berücksichtigung im Prototyping gewonnenen Erkenntnisse und die Abschlusspräsentation im Rahmen der Präsenzeinheit der LVA. Des Weiteren werden letzte Korrekturen an der Dokumentation vorgenommen und diese schlussendlich veröffentlicht.

Testphase	
Ressourcenplanung	7 PS
Funktionstest der Komponenten	8 PS
Test der Software	10 PS
Konnektivitätstest der Schnittstellen	5 PS
Kommunikationstest	5 PS

Abschlussphase	
Ausbesserungsarbeiten	8 PS
Korrekturlesung der Dokumentation	2 PS
Finalisierung der Dokumentation	15 PS
Planung der Abschlusspräsentation	5 PS
Testweise Präsentation	10 PS
Projektvorführung vor Unbeteiligten	3 PS
Vorbereiten der Abschlusspräsentation	16 PS
Abschlusspräsentation in Präsenzphase	3 PS

Testphase & Administration	
Testphase	35 PS
Abschlussphase	60 PS
GESAMT	*95 PS*

3.2 Ressourcenplanung

Bei der Auswahl der verwendeten Hardware und Zubehör wurde darauf Bedacht genommen, dass diese leicht zu beschaffen, universell einsetzbar und vor allem

kostengünstig sein muss. Es galt abzuwägen, welche Komponenten zwingend erforderlich waren um die angestrebten Funktionalitäten zu erfüllen und welche optional den
Funktionsumfang abrunden würden. Gleiches galt für die verwendete Software, diese wird als Open Source zu Verfügung gestellt und ist für jedermann frei erhältlich.

3.2.1 Hardware

3.2.1.1 TC4M-1294XL

Als Basis der Kommunikation dient ein Tiva C Series – TI Connected Launched mit einem TM4C129XL Cortex M4 Mikrocontroller.

Prozessortyp:	ARM Cortex-M4 CPU
Prozessortakt:	120MHz
Arbeitsspeichertyp:	1MB Flash, 256 KB SRAM, 6KB EEPROM
Schnittstellen:	10/100 Ethernet MAC+PHY, USB, Serial Communication Interfaces, Dual stackable Booster Pack

Figure 1 Tiva Board

3.2.1.2 Lowcost, Micropower, SC70/SOT23-8, Microphone

Features:

- +2.4V to +5.5V Supply Voltage Operation
- Versions with 5nA Complete Shutdown Available (MAX4467/MAX4468)
- Excellent Power-Supply Rejectionn Ratio: 112dB

- Excellent Common-Mode Rejection Ratio: 126dB
- High A_{VOL}: 125dB (R_L = 100kΩ)
- Rail-to-Rail Outputs
- Low 24µA Quiescent Supply Current

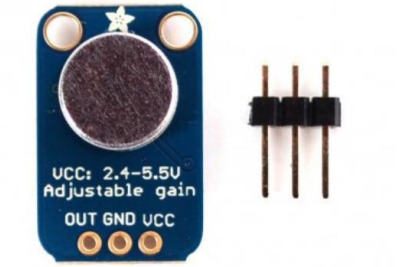


Figure 2 Microphone

3.2.1.3 Raspberry Pi 2 Model B

Als Basis für die Serverfunktionalität dient der Einplatinencomputer der britischen Raspberry Pi Foundation in der Hardware 2 Model B.

Prozessortyp:	ARM Cortex-A7 CPU QUAD
Prozessortakt:	900 MHz
Prozessorkerne:	4
Arbeitsspeicher:	1024 MB
Arbeitsspeichertyp:	DIMM: LPDDR2-SDRAM
Schnittstellen:	1x HDMI, 1x CVDS, 1x MIPI DSI, 1x 3,5mm Klinke, WLAN, BT, 4x USB 2.0, 1x Ethernet, 1x Kartenleser (microSDXC)

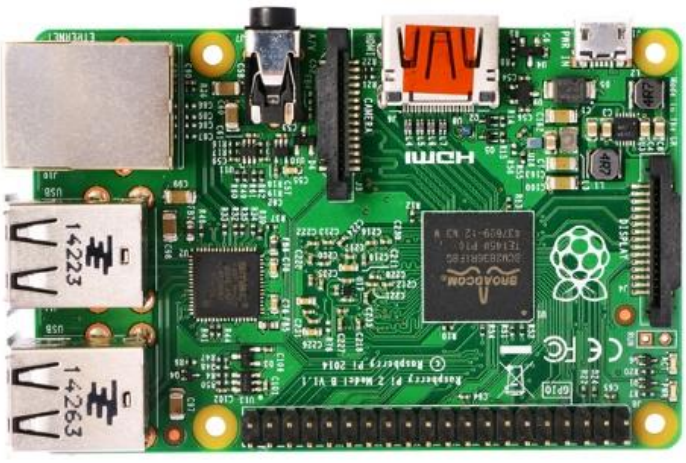


Figure 3 Raspberry Pi 2 Model B

3.2.2 Software

3.2.2.1 Code Composer Studio

Für die Implementierung der Tiva Series Boards dient CCS, da es in den vorherigen LV immanenten Fächern nähergebracht wurde und als kompatibel für TI's Microcontroller Einheiten und Embedded Processors gilt.

3.2.2.2 Eclipse (IDE)

Ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung (IDE) für die Programmiersprache Java genutzt, was auch für dieses Projekt dienlich ist.
Anzumerken ist jedoch, dass mittlerweile es wegen seiner Erweiterbarkeit auch für viele andere Entwicklungsaufgaben eingesetzt wird.

3.2.2.3 Raspbian

Als Betriebssystem (OS) des Einplatinencomputers dient das Debian-basierte, von der Raspberry Community, entwickelte Operating System „Raspbian", Release „Noobs".
Dieses wird in einschlägigen Online-Foren als prädestiniertes OS for beginners im Bereich von Raspberry Computing gehandelt.

3.2.3 Programmiersprachen

3.2.3.1 C

Die Programmiersprache C findet Anwendung im Bereich des Mikroprozessorumsetzung mittels TIVA C Board und dem Microphone.

3.2.3.2 Java

Die Programmiersprache Java dient zur Umsetzung des Serveranteils der Implementierung am Raspian OS.

3.2.3.3 HTML

Die Hypertext Markup Language ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

3.2.3.4 CSS

Cascading Style Sheets ist eine Stylesheet-Sprache für elektronische Dokumente und zusammen mit HTML und DOM eine der Kernsprachen des World Wide Webs. Sie ist ein sogenannter „living standard“ und wird vom World Wide Web Consortium (W3C) beständig weiterentwickelt.

3.2.3.4 Javascript

JavaScript ist eine Skriptsprache, die für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern. Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Microcontrollern.

4. Verantwortlichkeiten

In der Planungs- und Umsetzungsphase wurde seitens der projektdurchführenden Parteien dazu entschieden, das Projekt in sinnvolle und durchführbare Tasks zu gliedern und diese den Teammitgliedern zuzuteilen. Diese Einteilung erfolgte auf Basis von bereits gesammelten Erfahrungswerten je Person und profunder Weiterentwicklungsebene je geäußerten Wunsch.
Diese Einleitung führte zur Einführung von Verantwortlichkeitsbereichen, welche dem Projektumfang entsprach und eine gleichmäßige Aufteilung auf die mitwirkenden Personen gestaltete.

Andreas Mikula:	Projektplanung und Zeiteinteilung Aufbereitung und Nachverfolgungen Zusammentragen der Projektdaten SRS Konfiguration Raspbian OS TCP Kommunikation Tiva <-> Raspbian mittels C Webserver am Raspberry Pi mittel Apache HTTP Post Request Raspbian <-> Webserver Dokumentation
Daniel Schinewitz:	Clientaufbau Tiva Board TCP Kommunikation Tiva <-> Raspbian mittels Java Server Implementierung Microphone MAX4465 Businesslogic
Ibrahim Milli:	HTML Interface als Output Testcases
Philip Albrecht:	Hardwareauswahl Projektantrag Löten von Hardware Doxygen Testcases

5. Systemeigenschaften und -anforderungen

Die Basis der Implementierung bildet ein Client-Server Kommunikation zwischen TIVA und RASPI. Diese Kommunikation bedient sich der Technologie von TCP und wird kabelgebunden über Ethernet durchgeführt.

Die Hardwaremodulation zur Durchführung des Projektes besteht aus drei TIVA Boards gekoppelt mit jeweils einem MIC-Modul, welche über einen Switch mit einem RASPI verbunden sind.
Bei dem Kommunikationsaustausch übernimmt der RASPI die Rolle des Servers, welcher ankommende Datenstreams seinen Ursprung nach kennzeichnet und gesondert in einer Queue ablegt. Auf der anderen Kommunikationsseite übernehmen die TIVA Boards die Aufgabe des Clients und schicken Datensignale des MIC-Moduls an den Server.

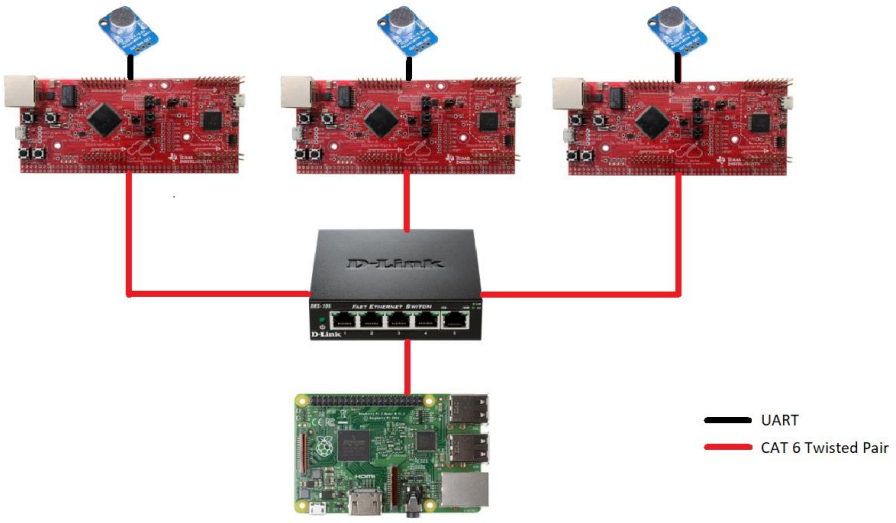


Abbildung 1 – Modulationsaufbau

5.1 Funktionale Anforderungen

Für einen funktionale Umsetzung des Projektes werden folgende Anforderungen gestellt. Die Verwendung einer µC TCP / IP-Bibliothek, mehrere Richtmikrofone, Zeitsynchronisation und ein Positionsbestimmungsalgorithmus.

Die ermittelte Position soll von der Basisstation, in unserem Fall einem Raspberry Pi, auf einer Website visualisiert werden. Diese Website wird durch einen lokalen Apache Webserver am RASPI abgedeckt.

5.2 Externe Schnittstellenanforderungen

TCP

Für die Anwendungen ist TCP transparent. Die Anwendung übergibt ihren Datenstrom an den TCP/IP-Stack und nimmt ihn von dort auch wieder entgegen. Mit der für die Übertragung nötige TCP-Paketstruktur sowie die Parameter der ausgehandelten Verbindung hat die Anwendung nichts zu tun.

Aufgaben und Funktionen von TCP:

- Segmentierung (Data Segmenting): Dateien oder Datenstrom in Segmente teilen, Reihenfolge der Segmente wiederherstellen und zu Dateien oder einem Datenstrom zusammensetzen
- Verbindungsmanagement (Connection Establishment and Termination): Verbindungsaufbau und Verbindungsabbau
- Fehlerbehandlung (Error Detection): Bestätigung von Datenpaketen und Zeitüberwachung
- Flusssteuerung (Flow Control): Dynamische Auslastung der Übertragungsstrecke
- Anwendungsunterstützung (Application Support): Adressierung spezifischer Anwendungen und Verbindungen durch Port-Nummern

Funktionsweise vom Transmission Control Protocol:

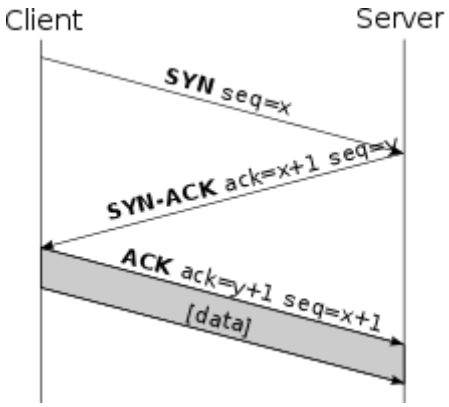


Figure 4 TCP - Quelle: https://upload.wikimedia.org/wikipedia/commons/thumb/9/98/Tcp-handshake.svg/220px-Tcp-handshake.svg.png

UART

Als UART wird der Teil eines µC oder Computers bezeichnet, der eine asynchrone serielle Datenübertragung erlaubt.
Die wichtigste Anwendung sind Schnittstellen wie RS232 oder RS485. Die Signale direkt am µC sind dabei (fast immer) noch CMOS- (bzw. TTL) Pegel. Direkt angeschlossen werden darf also eine RS232 oder ähnliches in der Regel nicht. Für die verschiedenen Schnittstellen gibt es passende Treiber/Reciever ICs wie MAX232 für RS232, MAX485 für RS485, und viele andere. Für kurze Strecken, z.B. auf einer Platine können die UART Signale direkt mit Logikpegel benutzt werden. Man hat dann aber halt keine RS232 Schnittstelle, sondern eine serielle Verbindung mit CMOS Pegeln.

5.3 Systemfeatures

In dieser Projektumsetzung gibt es keinen spezifischen Systemfeatures die zur Umsetzung dienlich sind.

5.4 Nicht-funktionale Anforderungen

Da dieses Projekt im Zuge eines Arbeitsauftrages im Lehrveranstaltungsbereich absolviert wird, wurde keine Rücksicht auf Performance, Safety, Security oder ähnliches gelegt.

6 Aussicht & Future Work

- Erweiterung um WLAN Modul bzgl. Kommunikation
- Testung mit dichterem Netzwerk an TIVA Modulen
- Testung mit anderer Hardware wie zB. qualitativ hochwertigeren Mikrophonen, Arduino anstelle von TIVA, etc.
- Implementierung in bestehende Systeme wie zB. Alarmsysteme zur Detection von auftretenden Geräuschen im Raum
- Nachbildung eines Richtmikrofons
- Implementierung von Safety / Security Features
- Implementierung in ein WAN
- Datenübertragung in die Cloud samt http Ausgabe und Datenaufbereitung über eine externe Homepage