|  |  |  |  |
| --- | --- | --- | --- |
| **Übungsprotokoll**  **SYTx – Systemtechnik <Teilbereich>** | | | |
|  | **Übungsdatum:**  KW 00/17 – 99/17 | **Klasse:**  5AHIT | **Name:**  Yi Liu |
| **Abgabedatum:**  13.3.2024 | **Gruppe:**  SYTI | **Note:** |
| **Leitung:**  DI (FH) Markus BRUNNER | **Mitübende:**  Gegebenenfalls hier anführen, muss mit Aufgabenteilung in der Aufgabenstellung korrespondieren! | | |
| **Übungsbezeichnung**:  Messomat 7k | | | |

**Inhaltsverzeichnis:**

[1 Aufgabenstellung 2](#_Toc492801999)

[2 Abstract (English) 2](#_Toc492802000)

[3 Theoretische Grundlagen 2](#_Toc492802001)

[4 Übungsdurchführung 2](#_Toc492802002)

[5 Ergebnisse 2](#_Toc492802003)

[6 Code 2](#_Toc492802004)

[7 Kommentar 2](#_Toc492802005)

# Aufgabenstellung

Ziel ist die Entwicklung eines intelligenten Temperatur- und Feuchtigkeitsüberwachungssystems, das (Mess-)Daten erfasst und an einen Rechner übermittelt. Zur Visualisierung Daten und Steuerung der Messanwendung ist im ersten Schritt eine einfache Terminalanwendung einzusetzen.

Weitere genauere Anforderungen sind im Dokument „*SYTI5\_UE2-Messomat7k-v1.4.pdf“* zu sehen.

# Abstract (English)

The aim is to develop an intelligent temperature and humidity monitoring system that records (measurement) data and transmits it to a computer. The first step is to use a simple terminal application to visualize data and control the measurement application.

More detailed information can be found in the document *“SYTI5\_UE2-Messomat7k-v1.4.pdf”*.

# Theoretische Grundlagen

Die theoretischen Grundlagen für die Übung mit dem ATMega328p und C-Programmierung umfassen mehrere Bereiche:

## ****Mikrocontroller ATMega328p****

1. Architektur: 8-Bit RISC, 32 Register, interne Peripheriegeräte (ADC, Timer, UART etc.)
2. Speicher: Flash (für Code), SRAM (für Variablen), EEPROM (für nichtflüchtige Speicherung)
3. GPIOs: Steuerung von Sensoren, Aktoren und LCD
4. Energie-Modi: Normalbetrieb vs. Energiesparmodus

## ****Sensoren & Peripheriegeräte****

1. **Temperatur- und Feuchtigkeitssensor** (z. B. DHT11, DHT22 oder SHT3x)
2. **Hochleistungsventilator** (Steuerung per PWM oder einfacher GPIO-Schaltung)
3. **LCD-Display** (z. B. HD44780, I2C- oder SPI-Ansteuerung)
4. **Taster-Eingaben** (z. B. Interrupt-gesteuert oder per Polling)

## ****Kommunikation & Datenübertragung****

1. **Serielle Kommunikation (UART)** zur Datenübertragung an den Host-Computer
2. **Protokoll mit STX/ETX** zur Rahmenbildung der Datenpakete
3. **Fehlermanagement**: Erkennung von Verbindungsabbrüchen, ACK/NACK-Prüfungen

## ****Embedded-C Programmierung****

1. Verwendung von **Header-Dateien (.h)** für systemweite Definitionen
2. **Bedingte Kompilierung** (Mock-Mode für Testzwecke)
3. **Interrupts & Timer** für präzise Steuerung (z. B. 1s vs. 4s Messintervall)
4. **EEPROM-Speicherung** zur Sicherung der letzten 10 Messwerte

## ****Watchdog & Fehlerbehandlung****

1. **Watchdog-Timer (avr/wdt.h)** für System-Reset bei Fehlern
2. **Fehlermanagement** durch Reset-Taste oder Software-Reset
3. **LED-Statusanzeige** für Verbindungsstatus

# Schaltplan

# Übungsdurchführung

Detaillierte Beschreibung der ausgeführten Tätigkeiten.

Z. B. bei DHCP: Definition der (Sub-)Netze, Installationsschritte, Konfiguration.

# Ergebnisse

Beschreibung und Darstellung der Ergebnisse in geeigneter Form, z. B. Screenshots, Tabellen, ...

# Code

Optionales Kapitel für Source Codes von Programmen, Skripten o. ä.

# Kommentar

Persönlicher Kommentar und eigene Interpretation der gewonnenen Ergebnisse.