***Planung und Realisierung eines Cyberphysischen Systems***

*Smarte Raum / Gebäudeüberwachung mit Alarmfunktionen*



*Projektdokumentation der Pentaguard-Solutions-UG*

***Inhaltsverzeichnis***

*1. Initiierungsphase*

*1.1 Zusammenstellung des Teams + Festlegung des Projektleiters*

*1.2 Zielidentifikation*

*1.3 Firmengründung*

*1.4 Abgabe des Projektantrags*

*2. Planungsphase*

*2.1 Erstellung der Arbeitspakete*

*2.2 Erstellung des GANTT-Diagramms*

*2.3 Materialplanung*

*2.4 Erstellung des Lasten- und Pflichtenhefts*

*2.5 Kostenkalkulation*

*2.6 Risikomanagement*

*3. Durchführungsphase*

*3.1 Frontend*

*3.1.1 Virtualisierung im Browser*

*3.2 Backend*

*3.3. Mikrocontroller*

*3.3.1 Code für ESP32*

*3.3.2 Netzwerkkommunikation herstellen*

*4. Testphase*

*4.1 Testen der Verschiedenen Systeme*

*4.2 Testprotokolle*

*4.3 Fehlerbehebung*

*5. Abschlussphase*

*5.2 Endkaufpreis*

*6. Anhänge*

***Initiierungsphase***

*1.1 Zusammenstellung des Teams + Festlegung des Projektleiters (CEO)*

Unser Team besteht aus 5 Leuten mit den unterschiedlichsten Stärken und Fähigkeiten. Wir haben uns zusammengeschlossen, da wir uns bereits aus Berufsschulzeiten kennen und sich jeder Untereinander kennt. Wir wissen, dass jeder sein ganz eigenes Spezialgebiet hat und so haben wir uns auch Untereinander aufgeteilt, damit wir eine Produktive Struktur und nahezu Perfektes Arbeitsklima entsteht. So haben wir auch gemeinsam entschieden, dass Joshua Barthel als Projektleiter und somit CEO des Unternehmens wird, somit übernimmt er alle Unternehmenswichtige Entscheidungen. Salvatore Ferreri ist als Stellvertreter eingestellt und übernimmt alle Wirtschaftlichen Geschäftsprozesse, zudem übernimmt er noch kleine Aufgabenbereiche im Bereich Netzwerk & Hardware. Felix Stüber ist für unsere Serververwaltung zuständig. Yannick Rörsch übernimmt unser Frontend und die Allgemeine später Oberfläche für die Anwender, sowie die Datenbankverwaltung und Dominique Diener ist für die Programmierung unserer Mikrocontroller und die damit verbundene Arbeit im Backend.

*1.2 Zieldefinition*

Die Zieldefinition bildet die Grundlage für eine strukturierte und zielgerichtete Projektplanung. Sie schafft Klarheit über den angestrebten Projekterfolg und dient allen Beteiligten als gemeinsame Orientierung. Eine präzise und nachvollziehbare Formulierung der Projektziele ist entscheidend, um den Umfang, die Anforderungen und die Erfolgskriterien des Projekts eindeutig festzulegen. Wir haben essenziell darauf geachtet, dass unsere Ziele SMART sind, um diese in der Testphase genau zu Prüfen.

Im Rahmen der Zieldefinition haben wir ein übergeordnete Projektziel: „Eine Funktionierende Raumüberwachung, mit Temperatur- und Luftfeuchtigkeitsanzeige, sowie ob sich jemand in diesem Raum befindet“ als auch gegebenenfalls untergeordnete Teilziele formuliert wie in unserem Fall:

* Eine Zutrittskontrolle
* Globale Überwachung (Über ein Webinterface)
* Weitere Verwaltung der Zutrittskontrolle (Zusätzliche Karten Verwaltung)
* Digitale Anzeige der Sensordaten über einem LCD-Display
* Logdatenspeicherung in einer Datenbank

Eine klare Zieldefinition war für uns wichtig, für alle weiteren Projektphasen und bildet die Basis für die spätere Erfolgskontrolle. Sie schafft Verbindlichkeit und unterstützt die Motivation und Ausrichtung des gesamten Projektteams.

*1.3 Firmengründung*

Als Projektgruppe, haben wir uns dazu entschieden eine Unternehmergesellschaft (UG) zu gründen. Eine UG bietet die gleichen Vor- bzw. Nachteile wie eine GmbH (Gesellschaft mit beschränkter Haftung) mit dem wesentlichen Unterschied, dass Gewinn des Unternehmens zu 25% wegfällt, um das Startkapital von 25.000,00€ zu erreichen, danach ist es auch möglich uns als GmbH eintragen zu lassen.

Im Zuge dessen haben wir mit 1€ das Unternehmen „Pentaguard-Solutions UG“ gegründet, mit Sitz in der Balthasar-Neumann-Straße 1.

Der Name ist ein Zusammenspiel daraus, dass wir 5 Mitglieder sind und Smarte IT-Lösungen für jeden Anwendungsfall produzieren und implementieren.

Der CEO des Unternehmens ist Joshua Barthel als sein Stellvertreter steht Salvatore Ferreri zur Verfügung. Die restlichen Mitglieder sind als Spezialisten der jeweiligen Abteilung eingestellt.

Als UG erfolgen Haftungen nur mit dem Geschäftsvermögen der

Gesellschafter, das war auch ein wichtiger Ausschlaggeber weshalb wir uns

für eine UG entschieden haben.

Der Gewinn wird prozentual auf die Einlagen der Gesellschafter aufgeteilt, da

wir allerdings alle gleich viel als Startkapital eingebracht haben, wird der

Gewinn gleichmäßig auf alle Gesellschafter aufgeteilt.

Somit entsteht kein Vorzug gegenüber anderen Gesellschaftern und alle

erhalten dieselbe Gewinnausschüttung.

Wir haben uns als die Firma „Pentaguard-Solutions UG“ im Handelsregister

eingetragen.

Alle Gesellschafter sind ebenfalls im Handelsregister eingetragen.

Die Gründung fand am 22.04.2025 statt.

Offizielles Firmenlogo der Pentaguard-Solutions UG

(Urheberrechtlich Geschützt)

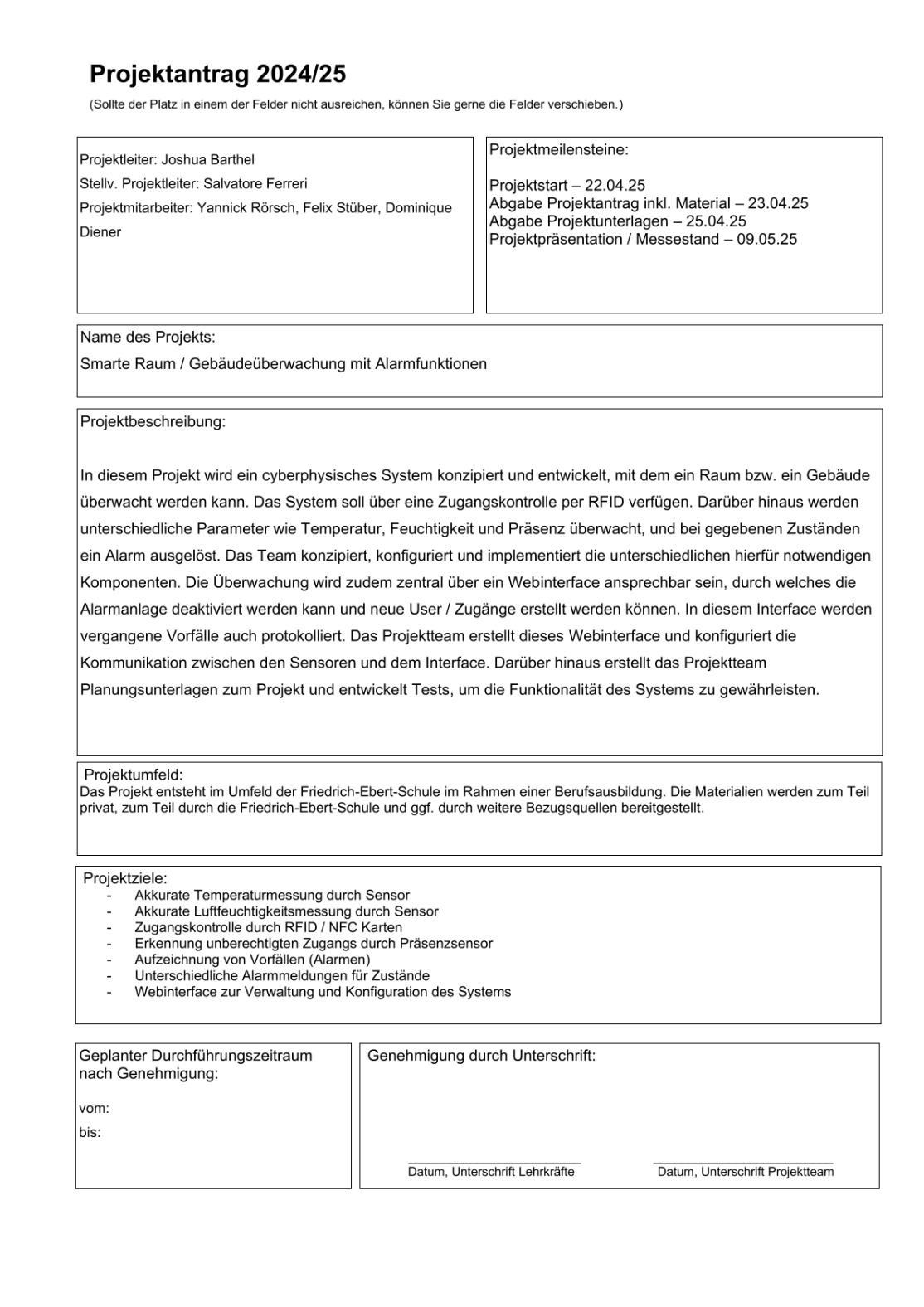
Ein Bild, das Logo, Symbol, Emblem, Text enthält.

KI-generierte Inhalte können fehlerhaft sein.

*1.4 Abgabe des Projektantrags*

Im Rahmen des Projektauftrags haben wir uns gemeinsam dazu beschlossen, eine Smarte Raum- bzw. Gebäudeüberwachung zu verwirklichen. Passend dazu haben wir einen Projektantrag mit allen nötigen Informationen, die dazugehören innerhalb der angegebenen Frist am 23.04.2025 abgegeben.

Offizieller Projektantrag:



***Planungsphase***

*2.1 Erstellung der Arbeitspakete*

Im weiteren Verlauf des Projekts, haben wir verschiedene Arbeitspakete zur Einrichtung der Gebäudeüberwachung erstellt. Die Arbeitspakete waren zur besseren Strukturierung und Planung vorgesehen.

Diese haben uns sehr dabei geholfen die Aufgaben zu definieren und unter den Mitgliedern besser zu verteilen. Durch diese strukturierte Planung konnten wir eine systematische und hochwertige Umsetzung des Projekts sicherstellen, sodass das bestmögliche Ergebnis erzielt werden konnte.

In unseren Arbeitspaketen sind alle Vorgänge mit den jeweiligen Zeitstempeln und den dazugehörigen Verantwortlichen beschrieben.

Die vollständige Liste der Arbeitspakete befindet sich im Anhang. (Arbeitspakete)

*2.2 Erstellung eines GANTT-Diagramms*

Wir haben uns gemeinschaftlich dazu entschieden zur Zeitplanung ein GANTT-Diagramm zu erstellen. Dieses veranschaulicht die Planung und Umsetzung unseres Projekts bei der „Pentaguard-Solutions UG“, zur Erstellung einer Smarten Raum- und Gebäudeüberwachung.

Mit Zeiteinheiten in Tagen sowie einer zusätzlichen Prozentangabe zu den einzelnen Aufgaben konnten wir die Projektinitiierung, Planung, Durchführung und den Abschluss mit zusätzlicher Testphase präzise steuern.

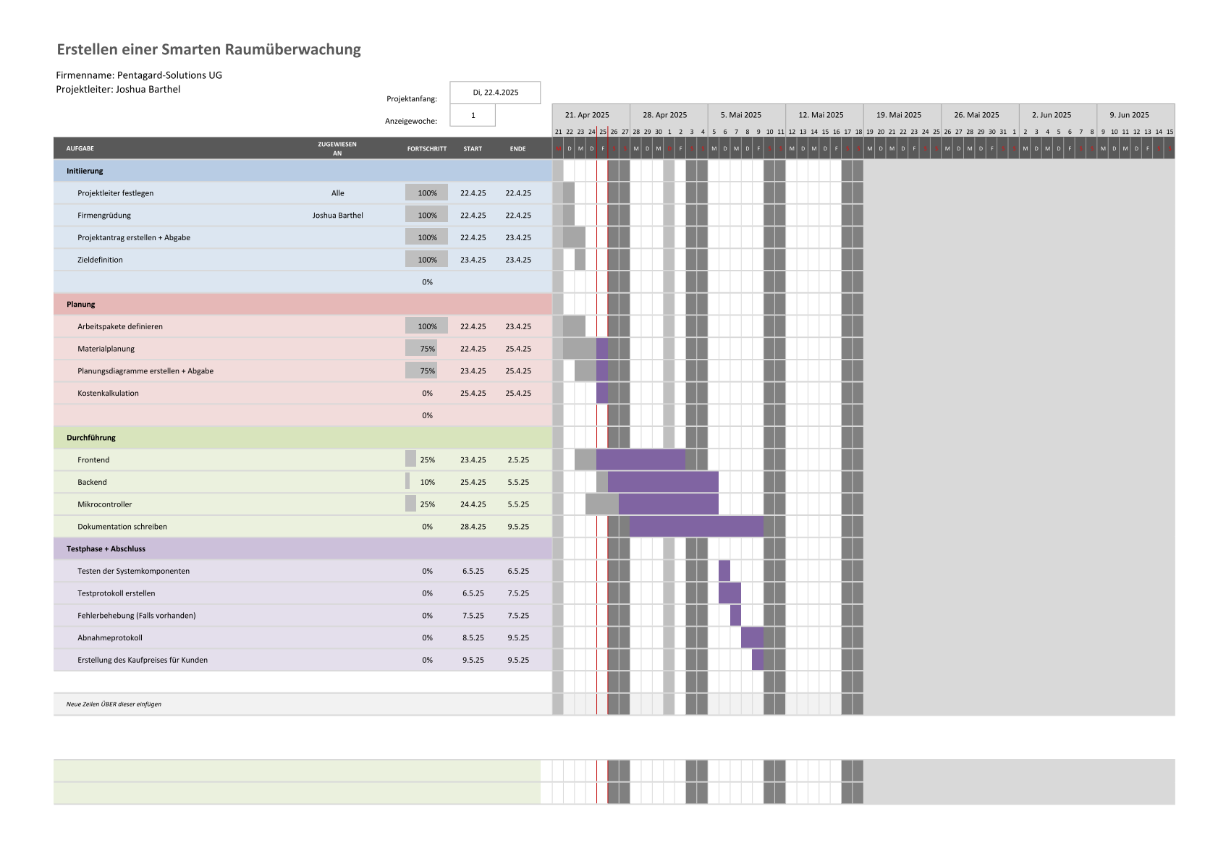
Die Projektinitiierungsphase umfasste die Teamzusammenstellung, Firmengründung, die Projektantrag Erstellung, sowie die Abgabe davon und die Zieldefinitionen.

Die Planungsphase enthielt die Erstellung der Arbeitspakete, die Materialplanung, die Erstellung der Planungsdiagramme und zusätzlich die Kostenkalkulation.

Die Durchführungsphase beinhaltet die Entwicklung des Frontend- und Backend sowie die Programmierung des Mikrocontrollers, zusätzlich kommt noch das Schreiben der Projektdokumentation hinzu.

Bei der Abschlussphase kommt die ganzen Protokolle wie Testprotokolle und Abnahmeprotokoll dazu, ebenfalls ist hier ein Zeitpuffer eingeplant für mögliche Fehlerbehebung, falls welche vorhanden sind.

Offizielles Gantt-Diagramm:



*2.3 Materialplanung*

Die Materialplanung stellt einen zentralen Bestandteil der Projektplanung dar und dient der bedarfsgerechten Versorgung mit allen notwendigen Materialien und Ressourcen. Ziel ist es, die Verfügbarkeit der benötigten Materialien zum richtigen Zeitpunkt, in der richtigen Menge und Qualität sicherzustellen, um einen reibungslosen Projektablauf zu gewährleisten und Verzögerungen zu vermeiden.

Im Rahmen der Materialplanung wurden zunächst die benötigten Materialien auf Basis der Projektanforderungen erfasst und dokumentiert. Dies haben wir mit der jeweiligen Fachkraft abgesprochen und eingetragen. Die Materialplanung erfolgt idealerweise in enger Abstimmung mit der Zeit- und Kostenplanung, da immer wieder Schwankungen bei den Preisen o.ä. Faktoren entstehen können. Eine regelmäßige Überprüfung und Aktualisierung der Materialbedarfsplanung ist notwendig, um flexibel auf Änderungen im Projektverlauf reagieren zu können.

*2.4 Erstellung Eines Lasten- und Pflichtenhefts*

*Lastenheft:*

|  |  |
| --- | --- |
| ID | Anforderung |
| LST\_01 | Erfassung von Sensordaten über Mikrocontroller und Sensoren |
| LST\_02 | Nutzung eines MQTT-Brokers zur Datenübertragung |
| LST\_03 | Verarbeitung der Sensordaten mit eigener oder angepasster Anwendung |
| LST\_04 | Speicherung der Daten in einer MySQL- oder SQLite-Datenbank |
| LST\_05 | Visualisierung und Interaktion im Browser |
| LST\_06 | Einzigartigkeit des Projekts ohne identische Online-Vorlage |
| LST\_07 | Verwendung eines geeigneten Sensors |
| LST\_08 | Integration von Aktoren zur Reaktion auf Daten |
| LST\_09 | Testkonzept zur Funktionsüberprüfung des Systems |
| LST\_10 | Projektdokumentation mit Einleitung, Planung, Durchführung, Ergebnissen |
| LST\_11 | Tägliche Dokumentation im Laborbuch |
| LST\_12 | Hardwarebeschaffung |
| LST\_13 | Daten- und IT-Sicherheit bei der Verarbeitung und Speicherung |
| LST\_14 | Beachtung des Nyquist-Shannon-Abtasttheorems bei der Sensorik |
| LST\_15 | Aufbau eines Messestands zur Präsentation |

*Pflichtenheft:*

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Abgeschlossen (Wie?) |
| LST\_01 | Erfassung von Sensordaten über Mikrocontroller und Sensoren | Wir haben einzelne kleine Testsysteme aufgebaut und ausführlich getestet. Nach erfolgreichem Testen haben wir die Systeme zusammengeführt und erneut getestet |
| LST\_02 | Nutzung eines MQTT-Brokers zur Datenübertragung | Wir haben einen lokalen MQTT-Broker eingerichtet und erfolgreich zur Kommunikation zwischen Sensorik und Anwendung genutzt. Die Verbindung wurde stabil getestet |
| LST\_03 | Verarbeitung der Sensordaten mit eigener oder angepasster Anwendung | Die erfassten Sensordaten werden in unserer selbst erstellten Anwendung verarbeitet und gefiltert. Die Logik ist auf unsere Systemanforderungen abgestimmt |
| LST\_04 | Speicherung der Daten in einer MySQL- oder SQLite-Datenbank | Wir verwenden für die Speicherung der Daten eine SQLite-Datenbank |
| LST\_05 | Visualisierung und Interaktion im Browser | Wir haben eine Webvisualisierung erstellt, in der jegliche Daten der Sensoren visuell dargestellt werden |
| LST\_06 | Einzigartigkeit des Projekts ohne identische Online-Vorlage | Dies haben wir getan, indem wir uns von vornerein einen eigenen Plan ausgedacht haben und keine Hilfe von dem Internet in Anspruch genommen haben. So stellen wir sicher, dass es einzig und allein ein Konzept ist, dass nur von uns stammt |
| LST\_07 | Verwendung eines geeigneten Sensors | RFID-Sensor, Temperatur- und Luftfeuchtigkeitssensor, Bewegungssensor (LD2410B) |
| LST\_08 | Integration von Aktoren zur Reaktion auf Daten | Ein Summer sowie eine LED werden über definierte Schwellenwerte gesteuert, z. B. bei Bewegungserkennung oder Temperaturüberschreitung. |
| LST\_09 | Testkonzept zur Funktionsüberprüfung des Systems | Wir haben zu jedem Modul Testszenarien definiert und diese dokumentiert. Die Ergebnisse werden laufend in Testprotokollen festgehalten. |
| LST\_10 | Projektdokumentation mit Einleitung, Planung, Durchführung, Ergebnissen | Jedes Gruppenmitglied ist für einen bestimmten Teil der Dokumentation verantwortlich und stellt diese bis zum geplanten Zeitpunkt fertig |
| LST\_11 | Tägliche Dokumentation im Laborbuch | Wir führen seit dem 1. Tag Laborbuch, indem wir jeden Tag einen kleinen Bericht verfassten und verfassen werden |
| LST\_12 | Hardwarebeschaffung | Wir haben uns informiert welche Hardware wir für das Projekt benötigen und haben eine Liste mit allen benötigten Teilen und Sensoren abgegeben |
| LST\_13 | Daten- und IT-Sicherheit bei der Verarbeitung und Speicherung | Wir nutzen eine lokale Datenbank, sichere Passwörter und Zugriffsbeschränkungen. Zudem wurden keine sensiblen Daten unnötig gespeichert |
| LST\_14 | Beachtung des Nyquist-Shannon-Abtasttheorems bei der Sensorik | Die Abtastraten unserer Sensoren wurden so gewählt, dass sie mindestens das Doppelte der maximalen Signalfrequenz betragen und somit dem Abtasttheorem entsprechen. |
| LST\_15 | Aufbau eines Messestands zur Präsentation | Geplante Tests mit einem Testaufbau des Messestandes |

*2.5 Kostenkalkulation*

Bei dem Thema Kostenkalkulation haben wir uns für einen Weg entschieden, um unsere Werte vor und nach der Vollendung die anfallenden Kosten zu vergleichen.

Wir haben zwei Rechnungstabellen erstellt, wovon eine bereits fertiggestellt ist, diese ist unsere SOLL-Tabelle womit geplant wird. Wir haben die Kosten für die benötigten Materialien und die vollständigen Personalkosten für jedes einzelne Teammitglied präzise anhand des aktuellen Azubigehaltes berechnet.

Die Materialien haben wir mit dem Wareneinkaufspreis (mit MwSt.) und für den Händlerpreis (ohne MwSt.) berechnet.

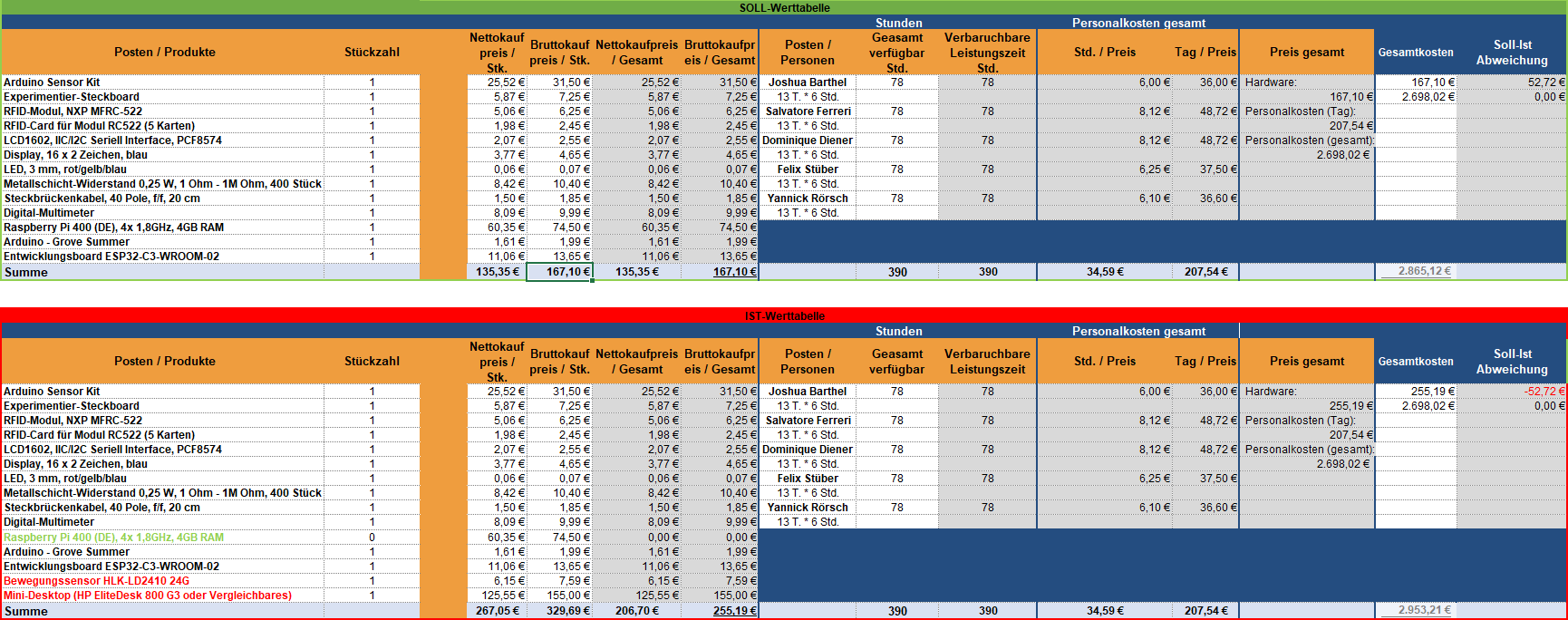
Unsere zweite Tabelle wird am Ende des Projektes gegengerechnet und ist somit unsere IST-Tabelle, dort werden die tatsächlich benötigten Materialien, sowie die tatsächliche Arbeitszeit eingetragen und berechnet.

Dadurch lässt sich dann auch genau eine Abweichung berechnen, ob man über oder unter den Erwartungen liegt.

In unserem Fall haben wir zu wenig Materialien eingeplant. Dass erkennt man daran, dass wir die Rot-Markierten Produkte im Laufe der Durchführung nachbestellen mussten, was wiederum am Ende zu einem etwas höheren Preis geführt hat.

Allerdings haben wir auch unter den Grün-Markierten Produkten eingespart, da wir diese nicht benötigt haben, wodurch sich letzten Endes eine Abweichung von -52,72€ ergab.

Damit liegen wir mit dem Verkaufspreis etwas über den Erwartungen, haben aber auch eine Effizientere Methode, um das System besser zu Administrieren und zu verwalten, was dem Endnutzer zur Gute kommt.



*2.6 Risikomanagement*

Im Rahmen der Planung unseres Projekts haben wir eine Risikoanalyse durchgeführt, um potenzielle Gefahren frühzeitig zu identifizieren und geeignete Gegenmaßnahmen festzulegen. Ziel ist es, die Zuverlässigkeit, Sicherheit und Qualität des Systems über den gesamten Projektverlauf hinweg zu sichern.

**Vorgehensweise**

Wir haben zunächst mögliche Standardrisiken gesammelt, die typischerweise bei der Entwicklung eines cyberphysischen Systems auftreten können. Für jedes Risiko wurden folgende Punkte bewertet:

* Wahrscheinlichkeit des Eintretens (z. B. niedrig, mittel, hoch)
* Auswirkungen, wenn das Risiko eintritt
* Eine Ampelfarbe (grün, gelb, rot) zur visuellen Risikobewertung
* Ursachenanalyse zur besseren Nachvollziehbarkeit
* Konkrete Maßnahmen zur Behandlung und Kontrolle

Diese Einschätzung hilft uns dabei, die Priorität einzelner Risiken zu erkennen und frühzeitig geeignete Vorkehrungen zu treffen.

**Ergebnisse**

Insgesamt wurden neun Risiken dokumentiert. Besonders kritisch (Ampelstufe 4) sind:

* **Hardwarekomponenten fallen aus**
* **Unzureichende Testabdeckung**

Diese Risiken wurden als besonders gefährlich eingeschätzt, da sie eine direkte Auswirkung auf die Funktionsfähigkeit oder Qualität des Gesamtsystems haben können. Entsprechend wurden dafür konkrete Gegenmaßnahmen eingeplant, wie z. B. ein stabiles IP-Konzept, Fehleranzeigen und strukturierte Testpläne.

**Ziel der Maßnahmen**

Die in der Analyse dokumentierten Risiken wurden jeweils mit konkreten Behandlungs- und Kontrollmaßnahmen versehen, um entweder

* das Auftreten zu verhindern (präventiv),
* oder die Auswirkungen zu minimieren, falls das Risiko dennoch eintritt (reaktiv).

**Erläuterung ausgewählter Maßnahmen**

* Sensoren liefern fehlerhafte Werte:  
  Um fehlerhafte Alarme oder das Ausbleiben kritischer Warnungen zu vermeiden, werden die Sensoren vorab getestet und kalibriert. Zusätzlich werden regelmäßige Funktionstests im Betrieb vorgesehen, um Abweichungen frühzeitig zu erkennen.
* Verbindungsprobleme zum Interface:  
  Die Verbindung ist zentral für Steuerung und Datenanzeige. Durch ein statisches IP-Konzept und die Integration von Fehlermeldungen im Interface wird sichergestellt, dass Verbindungsprobleme schnell erkennbar und behebbar sind.
* RFID-Zugänge funktionieren nicht:  
  Um ungewollte Zugangsprobleme oder unberechtigten Zutritt zu vermeiden, werden Test-User eingerichtet, ein Protokollierungssystem integriert und die Zuordnung der RFID-Tags regelmäßig überprüft.
* Alarm wird zu spät ausgelöst:  
  Die Verarbeitungskette wird so optimiert, dass eine Echtzeitverarbeitung sichergestellt ist. Schwellenwerte und Parameter werden getestet und bei Bedarf angepasst, um Verzögerungen zu minimieren.
* Fehlende Redundanz im System:  
  Um bei Ausfall einzelner Komponenten handlungsfähig zu bleiben, wird ein Notfallkonzept mit Backup-Komponenten (z. B. Bewegungssensor) eingeplant.
* Sicherheitslücke im Webinterface:  
  Zur Vermeidung unbefugter Zugriffe werden grundlegende Sicherheitsmaßnahmen wie HTTPS-Verschlüsselung, Passwortschutz eingeführt.
* Fehlende Benutzerrechteverwaltung:  
  Ein Rollenmodell (z. B. Admin vs. User) stellt sicher, dass nicht alle Nutzer dieselben Rechte haben. Das Interface wird zudem so abgesichert, dass unautorisierte Änderungen unterbunden werden.
* Hardwarekomponenten fallen aus:  
  Um Ausfälle einzelner Module (z. B. Sensoren, RFID-Leser) zu kompensieren, wird auf modulare Bauweise gesetzt und es werden alle Komponenten ausführlich getestet und als Backup noch Ersatzkomponenten angeschafft.
* Unzureichende Testabdeckung:  
  Dieses Risiko stellt eine ernsthafte Gefahr für die Qualitätssicherung dar. Durch das planvolle Aufstellen von Testszenarien und das Führen von Testprotokollen wird sichergestellt, dass alle Funktionen überprüft werden. So lassen sich Fehler bereits in der Entwicklungsphase erkennen.

***Durchführung***

*3.2 Backend*

**Ziel des Backends:**

Das Backend ist verantwortlich für die Kommunikation zwischen Datenbank, Frontend und Mikrokontroller. Das Backend stellt eine REST API zur Verfügung, über die das Frontend Daten bezieht, darüber hinaus sorgt das Backend für die Kommunikation mit dem ESP32, das Einspeisen der Daten des ESP32 in die Datenbank, sowie eine Websocket Verbindung zum Frontend für das Übertragen von Echtzeitdaten.

**Technologie:**

| **Komponente** | **Technologie / Paket** |
| --- | --- |
| Programmiersprache | Python 3.x |
| Web-Framework | Flask |
| Authentifizierung | JWT (JSON Web Tokens) |
| WebSocket | Flask-Sock |
| Datenbank | SQLite |
| Messaging | MQTT mit paho-mqtt |
| Rate-Limiting | Flask-Limiter |
| Sicherheit | Passwort-Hashing mit Werkzeug |
| CORS-Unterstützung | Flask-CORS |
| Datenformatierung | JSON |
| .env-Verwaltung | Python-dotenv |

**Aufbau:**

Das Backend besteht aus dem Hauptteil app.py, welcher für die Bereitstellung der Endpunkte, Websockets sowie Datenbankzugriffe verantwortlich ist.

Darüber hinaus wird die gesamte Kommunikation zwischen Backend und ESP32 über den MQTT\_handler.py abgewickelt. Dieser baut eine Verbindung zum MQTT Broker auf, hört auf Anfragen und kann auf vordefinierten Topics Nachrichten versenden.

**Sicherheitsaspekte:**

Das Backend ist durch unterschiedliche Methoden geschützt:.

* Admin Anmeldung – Stellt sicher, dass sich nur Admin User am Webinterface anmelden können. Die Passwörter in der Datenbank werden lediglich gehasht gespeichert.
* JSON Web Tokens (JWT) sorgen für sichere authentifizierte Sessions und schützen sensible Endpunkte vor nicht autorisierten Zugriffen
* MQTT Broker Passwort – Der MQTT Broker ist mit einem Passwort versehen, da die gesamte Kommunikation und Konfiguration über diese Schnittstelle läuft.
* Rate Limiting für Login – Das Backend begrenzt den Zugriff von Geräten (IP basiert), die sich zu häufig mit einem falschen Kennwort anmelden wollten.

**Endpunkte:**

Das Backend stellt folgende Endpunkte zur Verfügung:

**Login**

* **URL:** /login
* **Methode:** POST
* **Beschreibung:** Authentifiziert einen Benutzer und gibt ein JWT-Token zurück
* **Parameter:**
  + username: Benutzername
  + password: Passwort
* **Rückgabe:** JWT-Token bei erfolgreicher Authentifizierung
* **Hinweis:** Der Endpunkt verfügt über einen Ratenbegrenzer (10 Fehlversuche führen zu einer 5-minütigen Sperrung)

**Konfiguration abrufen**

* **URL:** /getConfig
* **Methode:** GET
* **Beschreibung:** Ruft die aktuelle Systemkonfiguration ab
* **Authentifizierung:** JWT erforderlich
* **Rückgabe:** Aktuelle Konfigurationsdaten

**Konfiguration setzen**

* **URL:** /setConfig
* **Methode:** POST
* **Beschreibung:** Aktualisiert die Systemkonfiguration
* **Authentifizierung:** JWT erforderlich
* **Parameter:** Konfigurationsdaten als JSON im Request-Body
* **Rückgabe:** Bestätigungsstatus

**Zugangs-ID hinzufügen**

* **URL:** /addAccessID
* **Methode:** POST
* **Beschreibung:** Fügt eine neue Zugangs-ID zum System hinzu
* **Authentifizierung:** JWT erforderlich
* **Rückgabe:** Bestätigungsstatus

**Daten abrufen**

* **URL:** /data
* **Methode:** GET
* **Beschreibung:** Ruft Dashboarddaten ab
* **Authentifizierung:** JWT erforderlich
* **Parameter:**
  + start\_date (optional): Startdatum für Filterung
  + end\_date (optional): Enddatum für Filterung
  + limit (optional, Standard: 100): Maximale Anzahl der zurückgegebenen Datensätze
* **Rückgabe:** Gefilterte Dashboarddaten

**Vorfälle abrufen**

* **URL:** /incidents
* **Methode:** GET
* **Beschreibung:** Ruft Vorfallsdaten ab
* **Authentifizierung:** JWT erforderlich
* **Parameter:**
  + start\_date (optional): Startdatum für Filterung
  + end\_date (optional): Enddatum für Filterung
  + type (optional): Vorfallstyp
  + limit (optional, Standard: 100): Maximale Anzahl der zurückgegebenen Datensätze
* **Rückgabe:** Gefilterte Vorfallsdaten

**Benutzerprofil abrufen**

* **URL:** /user/profile
* **Methode:** GET
* **Beschreibung:** Ruft das Profil des aktuellen Benutzers ab
* **Authentifizierung:** JWT erforderlich
* **Rückgabe:** Benutzerprofildaten

**Statusprüfung**

* **URL:** /health
* **Methode:** GET
* **Beschreibung:** Überprüft den Systemstatus
* **Rückgabe:** Systemstatusdaten und Zeitstempel

**WebSocket**

**WebSocket-Verbindung**

* **URL:** /ws
* **Beschreibung:** WebSocket-Endpunkt für Echtzeitkommunikation
* **Verwendung:** Verbindung über WebSocket-Protokoll

*Dokumentation zum Installieren/Aufsetzen des Appservers*

*Schritt 1:*

* Wir haben das aktuelle Ubuntu Server Image heruntergeladen und es auf den Proxmox geladen

*Schritt 2:*

* In der Weboberfläche des Proxmox haben wir eine neue VM erstellt und die VM nach den eigenen Vorstellungen und Vorgaben konfiguriert und als ISO-Image das vorher hochgeladene Ubuntu Server Image ausgewählt

*Schritt 3:*

* Danach haben wir die VM gestartet und das Image gebootet, es startet die Installation des Betriebssystems, hier haben wir meist die Standardeinstellungen verwendet. Das Einzige, was wir selbst eingestellt haben, sind die Netzwerkeinstellungen, hier haben wir eine statische IP verwendet, und außerdem haben wir bei der Installation noch angegeben, dass Docker direkt mitinstalliert wird.

*Schritt 4:*

* Nach der Installation hatten wir ein einfaches Ubuntu Server Betriebssystem, mit dem wir dann unsere Anwendungen hosten konnten.

*Schritt 5:*

* Für die verschiedenen Anwendungen haben wir Docker genutzt, hierfür haben wir eine „Docker-Compose.yml“ Datei geschrieben und mit dieser, verschiedene Container gestartet, dazu kann man mehr in der Docker Doku lesen. Danach sind unserer Container gelaufen und unser Appserver war fertig.

*3.3 Mikrocontroller*

Unser Projektziel ist die smarte Überwachung und Zugangskontrolle zu einem Rechenzentrum.  
Für dieses Vorhaben verwenden wir den **ESP32 Mikrocontroller**, da dieser nativ eine Verbindung über WiFi unterstützt. Darüber hinaus taktet der ESP32 schneller als beispielsweise ein Arduino Uno und bietet zudem mehr Ein- und Ausgänge.

Für unser cyber-physisches System setzen wir mehrere Sensoren und Aktoren ein:

**DHT11**

Der DHT11 ist ein Temperatur- und Luftfeuchtigkeitssensor. Er misst die Temperatur in Grad Celsius sowie die Luftfeuchtigkeit in Prozent. Die Daten werden digital übertragen, eine manuelle Normalisierung oder Kalibrierung war laut unseren Tests nicht notwendig. Die Einbindung des DHT11 in den Mikrocontroller-Code erfolgt über die Verwendung der Bibliothek <DHT.h>.

**LCD1602 (I²C)**

Das **LCD1602** ist ein Liquid Crystal Display, das in diesem Projekt zur Darstellung von Temperatur, Luftfeuchtigkeit sowie Systemzuständen dient. Es wurde über den **SDA- und SCL-Bus** angeschlossen, da es das I²C-Protokoll unterstützt.

**Buzzer V1.2**

Der Buzzer erzeugt hörbare Töne, deren Tonhöhe in Hertz (Hz) im Code definiert wird. Er wird primär für Zutrittssignale sowie für Alarmtöne verwendet. Der Aktor kommuniziert digital; eine zusätzliche Konfiguration ist nicht notwendig.

**LEDs**

Eine grüne und eine rote LED zeigen den aktuellen Systemstatus an:

* **Rot**: System gesichert (Alarm scharf geschaltet)
* **Grün**: System ungesichert (Alarm deaktiviert)  
  Die LEDs sind über 220-Ohm-Widerstände angeschlossen und werden über digitale Pins mit HIGH bzw. LOW geschaltet.

**RFID-RC522**

Der RFID-RC522 Sensor erkennt RFID-fähige Tags oder Geräte. Die Anbindung erfolgt über den **SPI-Bus**. Unsere Zugangskontrolle basiert auf dem Vergleich der übermittelten Tag-IDs mit freigegebenen IDs.

**Präsenzsensor**

Der Präsenzsensor LD2410 wird verwendet um menschliche Anwesenheit innerhalb des zu schützenden Raumes festzustellen. Der Sensor wurde vorab über ein UART zu USB Gerät so konfiguriert, dass die Parameter für dieses Projekt passen. Die Konfiguration des Sensors kann jederzeit über ein solches Gerät angepasst werden und sollte bei Installation des Systems überprüft werden. Der LD2410 kommuniziert sowohl über serielle Schnittstelle als auch digital über einen Pin. Im diesem Projekt wird lediglich der digitale Pin ausgelesen, da dieser Aufschluss darüber gibt ob sich jemand im Raum befindet oder nicht. Eine Messung des Abstands ist für die Überwachung nicht notwendig.

Abb.1 Konfiguration USB -> UART des Sensors:

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Verkabelung**

Die genaue Verkabelung der Komponenten ist im folgenden Schaubild dargestellt:

**Ein Bild, das Text, Diagramm, parallel, Plan enthält.

KI-generierte Inhalte können fehlerhaft sein.**

**Funktionalitäten des Mikrocontrollers**

**Grundfunktion**

Die Kernaufgaben des Mikrocontrollers sind:

* Messung von Temperatur und Luftfeuchtigkeit
* Einlesen von RFID-Karten
* Auslösung eines Alarms bei bestimmten Bedingungen

Ein Alarm wird ausgelöst, wenn:

* ein konfigurierbarer Grenzwert (z. B. Temperatur) überschritten wird
* eine Präsenz erkannt wird, während das System gesichert ist

**Konsistente Konfiguration**

Ein Teil des Speichers des ESP32 wird zur Dateiablage genutzt. Dort befinden sich u. a. die Dateien config.json und access\_IDs.json. Diese persistenten Daten ermöglichen es, Konfigurationen und Zugangs-IDs auch nach einem Neustart des ESP32 zuverlässig zu laden. Änderungen an der Konfiguration oder Zugangsberechtigungen können bequem über ein Webinterface vorgenommen werden.

**WiFi- & MQTT-Verbindung**

Der ESP32 stellt beim Start automatisch eine Verbindung zum konfigurierten WiFi-Netzwerk sowie zum MQTT-Broker her. Die Verbindung wird kontinuierlich überwacht; im Falle eines Abbruchs wird sie automatisch wiederhergestellt.

**Kommunikation über MQTT**

Die gesamte Netzwerkkommunikation des ESP32 läuft über **MQTT**. Es wurde ein passwortgeschützter MQTT-Broker eingerichtet.  
Sowohl die Kommunikation mit der Datenbank als auch die Konfiguration des Systems erfolgen ausschließlich über MQTT.

Da das System eine bidirektionale Kommunikation benötigt, kann der ESP32 sowohl MQTT-Nachrichten senden als auch empfangen und auswerten. Der Datenaustausch zwischen Backend und Mikrocontroller basiert auf MQTT-Kommandos und einer eigens entwickelten Nachrichtenstruktur.

**ArduinoMQTTClient.h**

Die ArduinoMQTTClient.h Bibliothek wird zum verbinden und übertragen der Daten über MQTT genutzt. Aufgrund der extensiven MQTT Kommunikation hat dies die Problematik hervorgerufen, dass MQTT Nachrichten auf 256 Bytes begrenzt waren. Die ArduinoMQTTClient.h Bibliothek wurde daher manuell so angepasst, dass größere Nachrichten unterstützt werden.

*Testphase*

*4.1 Testen der Verschiedenen Systeme*

Nachdem wir mit der Gesamten Durchführung des Projekts fertig waren, haben wir Systeme, die wir verwendet haben, getestet und uns auf die Angesprochenen Ziele aus *2.1* bezogen, um zu sehen, ob wir diese ordnungsgemäß nach unseren Vorstellungen erreicht haben oder ob wir nochmals Hand anlegen müssen.

Dazu haben wir ein Detailliertes Testprotokoll erstellt, welches alle angesprochenen Ziele aufzeigt und ob wir diese erreicht haben oder wo es Schwierigkeiten gab.

*4.2 Testprotokoll*

Projektname: Zutrittskontrolle mit Sensorik und Webinterface

Testzeitraum: 05.05.2025 – 09.05.2025

Tester: Gruppe 5

**Ziel des Tests**

Überprüfung der Funktionalität des gesamten Systems bestehend aus:  
- Bewegungserkennung  
- Temperatur- und Luftfeuchtigkeitsmessung  
- Zutrittskontrolle per Kartenleser  
- MQTT-Kommunikation zwischen Sensorik und Backend  
- Anzeige der Sensordaten im Webinterface

**Testumgebung**

Sensoren:  
- Präsenzsensor (LD2410)  
- DHT22 (DHT11)  
- RFID-Kartenleser (MFRC522)  
  
**Software:**  
- MQTT-Bibliothek (angepasst wegen Initialisierungsproblem)  
- Python-Skript zur Datenverarbeitung  
- Webserver   
- HTML/JS/TS-Frontend  
  
**Netzwerk:**  
- Lokales LAN mit MQTT-Broker (z. B. Mosquitto)

**Durchgeführte Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testfall | Beschreibung | Erwartetes Ergebnis | Ergebnis | Bemerkung |
| 1 | Bewegung erkannt | Bewegung wird erkannt und über MQTT übertragen | ✔️ | Funktioniert zuverlässig |
| 2 | Keine Bewegung | Kein MQTT-Signal wird gesendet | ✔️ | OK |
| 3 | Temperaturmessung | Temperaturwert wird korrekt angezeigt | ✔️ | Abweichung <2°C |
| 4 | Luftfeuchtigkeit | Luftfeuchtigkeitswert wird korrekt angezeigt | ✔️ | OK |
| 5 | Temperaturänderung simulieren | Wert ändert sich im Webinterface | ✔️ | Verzögerung ca. 2 Sek. |
| 6 | RFID-Karte korrekt | Zugang wird gewährt, Log im Webinterface | ✔️ | Testkarte funktioniert |
| 7 | Unbekannte RFID-Karte | Zugang wird verweigert | ✔️ | OK |
| 8 | MQTT-Kommunikation | Alle Daten erreichen den Broker korrekt | ✔️/❌ | Anfangs Problem mit Bibliothek |
| 9 | Konfiguration des Mikrocontrollers über das Webinterface | Konfiguration wird korrekt übernommen | ✔️ | OK |
| 10 | Webinterface Datenanzeige | Live-Daten erscheinen korrekt | ✔️ | Formatierung angepasst |
| 11 | Uhrzeit-Anzeige | Korrekte Uhrzeit im Webinterface | ✔️ | Zeitsynchronisierung aktiv |

*Abschlussphase*

*5.1 Erstellung des Kaufpreises*

Bei der finalen Preiserstellung kann man sich größtenteils auf unsere Kostenkalkulation beziehen. Hierbei handelt es allerding nur um ein Konzept, je nach Anforderung des Kunden und der aktuellen Marktlage, kann der Preis natürlich Schwanken.

Laut Rechnung wären wir dann bei einem Verkaufspreis von 2.952,21€.

Da wir auch noch einen kleinen Gewinnzuschlag von 10% berechnen, wird das Endangebot etwas höher ausfallen. In diesem Fall liegen wir dann bei einem Preis von 3.248,30€.

Allerdings ist dieser Preis auch gerechtfertigt, da der Kunde einen 1 Jahres Support und Garantie von uns bekommt. Damit wollen wir Konkurrenzfähig bleiben und uns von anderen Unternehmen abheben.