Projektabgabe | Technische Dokumentation

• Benutzer-Dokumentation des Systems:

o Zweck: Übergabe an andere Mitarbeiter (Fachinformatiker)

o Welche Schnittstellen wurden benutzt und wie sind sie implementiert?

o Wie sind diese Schnittstellen zu benutzen?

o Wie ist die Datenbank aufgebaut? (z.B. ER oder Ähnliches)

o Code-Dokumentation soweit zum Verständnis notwendig

# Übergabe an andere Mitarbeiter

## Einrichtung der virtuellen Maschine:

Die Stadt Fulda stellte eine virtuelle Maschine zur Verfügung. Mithilfe eines SSH-Clients wie Putty wird eine sichere Verbindung zur VM hergestellt.

Hierfür benötigt man die IP-Adresse der VM sowie die Anmeldeinformationen (Benutzername und Passwort oder SSH-Schlüssel), um dich über Putty zu authentifizieren und Zugriff auf die VM zu erhalten.

## Installation von MongoDB auf der virtuellen Maschine:

Verbindung über Putty mit der VM und öffnen eines Terminals.

Sicherstellen, dass der MongoDB-Dienst läuft und auf die richtigen Ports (standardmäßig 27017) lauscht.

## Raspberry Pi und Webcam-Integration:

Verbindung mit dem Raspberry Pi mit der Webcam und Sicherstellung, dass die erforderlichen Treiber installiert sind.

Stelle sicher, dass die Python-Anwendung die erkannten Farben und die zugehörigen Informationen an die MongoDB-Datenbank sendet.

## Daten in die MongoDB-Datenbank eintragen:

Verwende den offiziellen MongoDB-Treiber für Python, um eine Verbindung zur MongoDB-Datenbank herzustellen.

Definiere die Struktur der Datenbankdokumente und erstelle entsprechende Sammlungen oder Dokumente, um die erfassten Daten zu organisieren.

Verwende die API des MongoDB-Treibers, um Daten in die MongoDB-Datenbank einzufügen, zu aktualisieren oder abzurufen.

## Testen und Validieren der Integration:

Überprüfe, ob die Webcam korrekt funktioniert und Würfel korrekt einscannt und Farben erkennt.

Stelle sicher, dass die erfassten Daten in der MongoDB-Datenbank gespeichert werden und dass sie korrekt abgerufen werden können.

Führe umfangreiche Tests durch, um sicherzustellen, dass die Integration zuverlässig funktioniert und den Anforderungen des Roboterprojekts entspricht.

# Welche Schnittstellen wurden benutzt und wie sind sie implementiert?

SSH (Secure Shell)

SSH ist ein Netzwerkprotokoll, das eine sichere verschlüsselte Verbindung zwischen unserem lokalen Computer und der virtuellen Maschine ermöglicht. Es wurde verwendet, um über Putty eine Verbindung zur VM herzustellen und die Konfiguration durchzuführen.

MongoDB

MongoDB ist eine NoSQL-Datenbank, die in unserem Projekt verwendet wurde. Es handelt sich um eine dokumentenorientierte Datenbank, die flexible und skalierbare Speicherung von Daten ermöglicht.

## MQTT

MQTT (Message Queuing Telemetry Transport) ist ein leichtgewichtiges Nachrichtenprotokoll, das für die zuverlässige und effiziente Kommunikation zwischen Geräten in Netzwerken mit begrenzter Bandbreite und instabiler Konnektivität entwickelt wurde. Es ist ein weit verbreitetes Protokoll im Internet of Things (IoT)-Bereich.

# Wie sind diese Schnittstellen zu benutzen?

## SSH

Öffne ein Terminal oder eine SSH-Client-Anwendung.

Gib den Befehl ein, um dich mit der VM zu verbinden, normalerweise unter Verwendung der IP-Adresse und der Anmeldeinformationen, die dir für die VM bereitgestellt wurden.

Nach erfolgreicher Verbindung kannst du Befehle ausführen, um die VM zu konfigurieren, Software zu installieren oder andere Aufgaben im Zusammenhang mit dem Projekt durchzuführen.

## MongoDB

Stelle sicher, dass MongoDB auf der VM installiert ist.

Verbinde dich mit der MongoDB-Instanz auf der VM, normalerweise über eine Befehlszeilenschnittstelle oder eine MongoDB-Client-Anwendung.

Verwende Befehle oder Code in der entsprechenden Programmiersprache, um Datenbanken, Sammlungen und Dokumente zu erstellen.

Führe Operationen wie das Einfügen, Aktualisieren, Löschen oder Abfragen von Daten durch, um mit der MongoDB-Datenbank zu interagieren.

## MQTT

Implementiere eine MQTT-Client-Bibliothek oder -Komponente in deinem Python-Code, um eine Verbindung zum MQTT-Broker herzustellen.

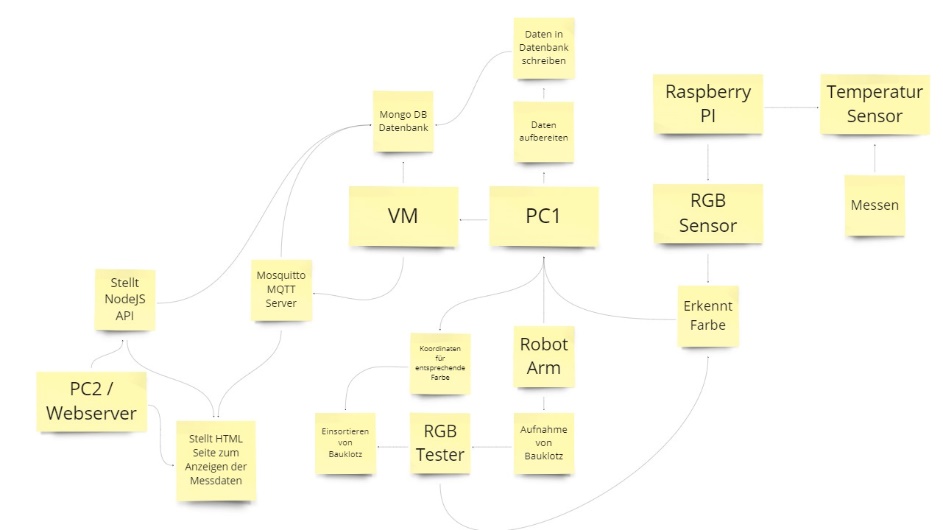
Konfiguriere den MQTT-Client, um sich mit dem MQTT-Broker auf der VM zu verbinden, normalerweise unter Verwendung einer IP-Adresse oder eines Hostnamens und Portnummern.

Abonniere MQTT-Themen, um Nachrichten zu empfangen, die von anderen Geräten oder Clients gesendet werden.

Sende MQTT-Nachrichten an bestimmte Themen, um Informationen oder Befehle an andere Geräte oder Clients zu übermitteln.

# Wie ist die Datenbank aufgebaut?

## Installation der MongoDB-Datenbank

Um die Datenbankfunktionalität bereitzustellen, haben wir MongoDB auf der virtuellen Maschine installiert. Hierbei sind folgende Schritte erforderlich:

Öffnen einer Terminalverbindung zur VM mittels SSH (Secure Shell) über Putty.

SSH bietet eine verschlüsselte Kommunikation zwischen unserem lokalen Computer und der virtuellen Maschine, um die Sicherheit zu gewährleisten.

## Installation von MongoDB auf der VM durch Ausführen der entsprechenden Befehle.

Hierbei haben wir den Paketmanager der verwendeten Betriebssystemdistribution genutzt, um MongoDB zu installieren. Genauere Informationen zu den Befehlen können der offiziellen MongoDB-Dokumentation entnommen werden.

## Konfiguration der MongoDB-Instanz.

Nach der Installation haben wir die Konfiguration der MongoDB-Instanz vorgenommen, um sicherzustellen, dass sie für unsere Anforderungen optimal funktioniert. Hierzu gehören Aspekte wie Speicherplatz, Netzwerkeinstellungen und Sicherheitseinstellungen.

## Import von Daten in die MongoDB-Datenbank.

Nach erfolgreicher Installation und Konfiguration haben wir Daten in die MongoDB-Datenbank importiert. Hierbei haben wir geeignete Dateiformate (z. B. JSON oder CSV) verwendet und die entsprechenden Befehle oder Tools genutzt, um die Importvorgänge durchzuführen.

# Code-Dokumentation

## RASPI

* Zwei Python Scripts welche als Server dienen unter /usr/local/bin/
* Service Dateien liegen im user service directory, und sollten unter dem Benutzer als Service angelegt werden.
* Fest auf die IP Adresse gebunden
* Folgende dependencies wurden nachinstalliert: asyncio, asyncua, time, adafruit\_dht, netifaces, numpy, colorsys, cv2-python
* Farben können durch COLOR\_ANCHOR ergänzt werden, es wird das HSV Schema verwendet
* Theoretisch können mehrere Geräte sich auf das gerät verbinden, aber praktisch würde trotzdem nur eine Kamera geteilt werden
* Raspi-Skripte müssen auf dem Benutzer laufen, sonst fehlen dependencies
* Bild wird leicht verschärft um etwas gegen schlechtere licht Verhältnisse zu wirken

## DOBOT

* Umgebung muss auf einer 32-bit Python-Installation laufen
* Aufgesetzt für eine Windowsinstallation
* Folgende Dependencies wurden nachinstalliert: pymongo, asyncio, paho
* Der DOBOT braucht alle anderen Komponenten (Raspi, VM) damit es funktioniert
* DOBOT schreibt in DB und teilt auch die Information live via MQTT

## Folgendes Protokoll für Handshake

* Client schickt handshake mit version
* Server schickt acknowledge (version OK)
* Server schickt bad\_version (version falsch)
* Server schickt expected\_shake (erwarte handshake, hab anderes packet gekriegt)

## Folgendes Protokoll für Scan

* Client schickt Scan-Anfrage
* Server antwortet mit Farben Klarnamen
* Kann etwas dauern, mehrere samples notwendig für linsen Kalibration

## Schließen-Protokoll

* Client schickt close-Anfrage
* Server antwortet „goodbye“

**Code Dokumentation für restliche Komponenten und aufsetzen des Projekts zum Testlauf sind in der README.md im Github Repository zu finden.**