Brauerei mit Lagersystem in Minecraft

Tom Mangelsen

Tjell Albertsen

Jonas Kamp

Fachinformatiker für Systemintegration

KDIT4b

Berufliche Schule des Landkreises Nordfriesland
in Husum

Cyber Physisches System

Dr.-Ing. Florian Johannes Knoll

Abgabe am 30.04.2026

Inhaltsverzeichnis

Dokumentation

1. Einleitung

Im Rahmen dieses Projekts wird ein System entwickelt, das Daten aus einer eigens aufgebauten Minecraft-Welt ausliest und für die Weiterverarbeitung in einem vernetzten Hardwaresystem bereitstellt. Ziel ist es, Informationen aus der Spielumgebung automatisiert zu erfassen und in Echtzeit an mehrere Mikrocontroller zu übertragen.

Die Datenerfassung erfolgt über ein Plugin, das direkt in die Minecraft-Welt integriert ist. Dieses Plugin liest relevante Informationen, wie beispielsweise den Inhalt von Kisten, aus und übermittelt diese an einen ESP32-Mikrocontroller. Der ESP32 fungiert dabei als zentrale Schnittstelle und veröffentlicht die empfangenen Daten als sogenannte Topics auf einem MQTT-Server.

Die weiteren ESP32-Geräte abonnieren diese Topics als Subscriber und erhalten dadurch automatisch die aktuellen Daten vom MQTT-Server. Dieses Publish-Subscribe-Prinzip ermöglicht eine flexible und skalierbare Kommunikation zwischen den einzelnen Komponenten, da Sender und Empfänger voneinander entkoppelt sind.

Innerhalb des Systems übernehmen die einzelnen Module klar definierte Aufgaben. Ein Modul dient zur Bestellung von Tränken, ein weiteres verarbeitet und visualisiert Serverstatistiken, während ein drittes Modul den aktuellen Lagerbestand wiedergibt. Durch diese Aufteilung wird eine strukturierte und übersichtliche Systemarchitektur erreicht.

Das Projekt verbindet damit verschiedene Bereiche der Informationstechnik, insbesondere die Verarbeitung von Spieldaten, die Nutzung von Netzwerkprotokollen wie MQTT sowie den Einsatz von Mikrocontrollern im Kontext verteilter Systeme. Im weiteren Verlauf der Dokumentation werden die technische Umsetzung sowie die einzelnen Komponenten detailliert beschrieben.

2. Hardware:

- **SSD1306 Display (3x)**
    - Ein SSD1306-Display ist ein kleines OLED-Display, das Text, Zahlen und einfache Grafiken anzeigen kann. Es eignet sich optimal für unser Projekt, da es wenig Strom braucht und über wenige Leitungen angesteuert werden kann.
    - Die Displays benötigen folgende Bibliotheken:
        - <Wire.h> ->Kommunikation zum ESP
        - <adafruit_gfx.h> ->liefert Funktionen für Text, Grafiken, etc.
        - <adafruit_ssd1306.h> ->Treiber für SSD1306-Chip basierte Displays

- **KY-040 Drehschalter (3x)**
    - KY-040 sind Drehgeber, mit denen man Drehbewegungen und in unserem Fall auch Knopfdrücke erfassen können. Sie sind gut geeignet, um Werte einzustellen oder durch Menüs zu navigieren.

- **ESP32 (3x)**
    - Der ESP32 ist als Mikrocontroller mit integriertem WLAN und Bluetooth das Herzstück unseres Projektes. Er steuert die Sensoren und Displays, verarbeitet Eingaben und sendet Daten an die anderen ESP-Module.

- **3D-gedrucktes Gehäuse von Tom (3x)**
    - PLA 100% Infill (grau & schwarz)
        - Hierbei wurden die bereitgestellten 3D-Modelle teilweise genutzt & unseren Anforderungen entsprechend angepasst.

- **VEXUNGA 12MM Drucktaster**

- Diese Kombination aus 4 einzelnen Tastern übernimmt das Navigieren durch die Menüs

3. Hauptteil

3.1 Was passiert in Minecraft?

Das von Tjell in der Minecraft-Welt umgesetzte Lagersystem bildet die Grundlage für die Erfassung und Weiterverarbeitung der relevanten Daten. Es umfasst sowohl fertige Tränke als auch die zur Herstellung benötigten Ressourcen, wie beispielsweise Zutaten für Brauprozesse. Die Struktur des Lagers ist so aufgebaut, dass die einzelnen Inhalte eindeutig zugeordnet und automatisiert ausgelesen werden können. Dadurch ist es möglich, jederzeit den aktuellen Bestand sowie fehlende Materialien zu erfassen. Diese Informationen dienen als Basis für weitere Funktionen des Systems, insbesondere für die Anzeige des Lagerbestands und die Unterstützung bei der Bestellung neuer Tränke.

3.2 Minecraft Schnittstelle für MQTT

Im Verlauf des Projekts wurde zunächst ein MQTT-Server eingerichtet, um die Grundlage für die Kommunikation zwischen den einzelnen Komponenten zu schaffen. Zur Überprüfung und Analyse der übertragenen Daten wurde zusätzlich das Tool MQTT Explorer installiert. Dabei zeigte sich, dass ein zu niedrig gewähltes Aktualisierungsintervall zu Problemen führte, weshalb ein Delay von einer Sekunde verwendet wurde.

Anschließend wurde ein Minecraft-Server aufgesetzt und das benötigte Plugin integriert. Dabei trat zunächst ein Kompatibilitätsproblem auf, da das Plugin für einen Paper-Server entwickelt wurde, während ursprünglich ein Spigot-Server verwendet wurde. Dieses Problem wurde durch das erneute Aufsetzen eines Servers auf Basis von Paper behoben.

Nach der erfolgreichen Integration und Konfiguration des Plugins konnten erste Tests innerhalb der Minecraft-Welt durchgeführt werden. Hierbei wurden insbesondere

Kisten als Datenquelle genutzt. Die ausgelesenen Informationen wurden korrekt an den MQTT-Server übertragen und konnten dort erfolgreich eingesehen werden, womit die grundlegende Funktionalität des Systems bestätigt wurde.
Die folgende Tabelle zeigt die Zuordnung der einzelnen Minecraft-Daten zu den jeweiligen MQTT-Topics. Dabei sind die Rohwerte aus Minecraft in der Spalte ‚MC_Out' und die weiterverarbeiteten Ausgabewerte in der Spalte ‚Out' dargestellt.

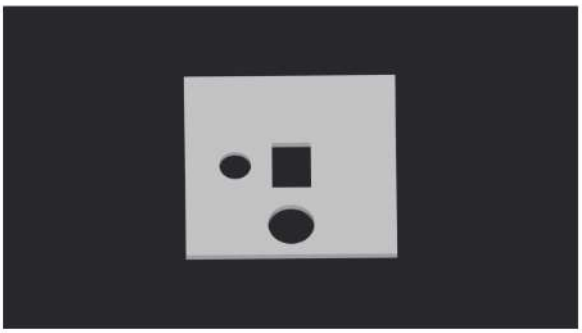
NAME	MC_Out (Unverarbeiteter Wert)	Out (Verarbeiteter Wert)
Potion_of_Invisibility	KDIT4b/Gruppe_Bier/Minecraft_Out/Potion_of_Invisibility	KDIT4b/Gruppe_Bier/Out/Potion_of_Invisibility
Potion_of_Strength_II	KDIT4b/Gruppe_Bier/Minecraft_Out/Potion_of_Strength_II	KDIT4b/Gruppe_Bier/Out/Potion_of_Strength_II
Potion_of_Healing_II	KDIT4b/Gruppe_Bier/Minecraft_Out/Potion_of_Healing_II	KDIT4b/Gruppe_Bier/Out/Potion_of_Healing_II
Potion_of_Fire_Resistance	KDIT4b/Gruppe_Bier/Minecraft_Out/Potion_of_Fire_Resistance	KDIT4b/Gruppe_Bier/Out/Potion_of_Fire_Resistance
Glass_Bottle	KDIT4b/Gruppe_Bier/Minecraft_Out/Glass_Bottle	KDIT4b/Gruppe_Bier/Out/Glass_Bottle
Glass	KDIT4b/Gruppe_Bier/Minecraft_Out/Glass	KDIT4b/Gruppe_Bier/Out/Glass
Melon_Slice	KDIT4b/Gruppe_Bier/Minecraft_Out/Melon_Slice	KDIT4b/Gruppe_Bier/Out/Melon_Slice
Gold_Nugget	KDIT4b/Gruppe_Bier/Minecraft_Out/Gold_Nugget	KDIT4b/Gruppe_Bier/Out/Gold_Nugget
Gold_Ingot	KDIT4b/Gruppe_Bier/Minecraft_Out/Gold_Ingot	KDIT4b/Gruppe_Bier/Out/Gold_Ingot
Blaze_Rod	KDIT4b/Gruppe_Bier/Minecraft_Out/Blaze_Rod	KDIT4b/Gruppe_Bier/Out/Blaze_Rod
Carrot	KDIT4b/Gruppe_Bier/Minecraft_Out/Carrot	KDIT4b/Gruppe_Bier/Out/Carrot
Spider_Eye	KDIT4b/Gruppe_Bier/Minecraft_Out/Spider_Eye	KDIT4b/Gruppe_Bier/Out/Spider_Eye
Brown_Mushroom	KDIT4b/Gruppe_Bier/Minecraft_Out/Brown_Mushroom	KDIT4b/Gruppe_Bier/Out/Brown_Mushroom
Slimeball	KDIT4b/Gruppe_Bier/Minecraft_Out/Slimeball	KDIT4b/Gruppe_Bier/Out/Slimeball
Sugar	KDIT4b/Gruppe_Bier/Minecraft_Out/Sugar	KDIT4b/Gruppe_Bier/Out/Sugar
Sugar_Cane	KDIT4b/Gruppe_Bier/Minecraft_Out/Sugar_Cane	KDIT4b/Gruppe_Bier/Out/Sugar_Cane

Sand	KDIT4b/Gruppe_Bier/Minecraft_Out/Sand	KDIT4b/Gruppe_Bier/Out/Sand
Coal	KDIT4b/Gruppe_Bier/Minecraft_Out/Coal	KDIT4b/Gruppe_Bier/Out/Coal
Water_Bottle	KDIT4b/Gruppe_Bier/Minecraft_Out/Water_Bottle	KDIT4b/Gruppe_Bier/Out/Water_Bottle
Nether_Wart	KDIT4b/Gruppe_Bier/Minecraft_Out/Nether_Wart	KDIT4b/Gruppe_Bier/Out/Nether_Wart
Glistering_Melon_Slice	KDIT4b/Gruppe_Bier/Minecraft_Out/Glistering_Melon_Slice	KDIT4b/Gruppe_Bier/Out/Glistering_Melon_Slice
Glowstone_Dust	KDIT4b/Gruppe_Bier/Minecraft_Out/Glowstone_Dust	KDIT4b/Gruppe_Bier/Out/Glowstone_Dust
Blaze_Powder	KDIT4b/Gruppe_Bier/Minecraft_Out/Blaze_Powder	KDIT4b/Gruppe_Bier/Out/Blaze_Powder
Magma_Cream	KDIT4b/Gruppe_Bier/Minecraft_Out/Magma_Cream	KDIT4b/Gruppe_Bier/Out/Magma_Cream
Golden_Carrot	KDIT4b/Gruppe_Bier/Minecraft_Out/Golden_Carrot	KDIT4b/Gruppe_Bier/Out/Golden_Carrot
Fermented_Spider_Eye	KDIT4b/Gruppe_Bier/Minecraft_Out/Fermented_Spider_Eye	KDIT4b/Gruppe_Bier/Out/Fermented_Spider_Eye
Redstone_Dust	KDIT4b/Gruppe_Bier/Minecraft_Out/Redstone_Dust	KDIT4b/Gruppe_Bier/Out/Redstone_Dust

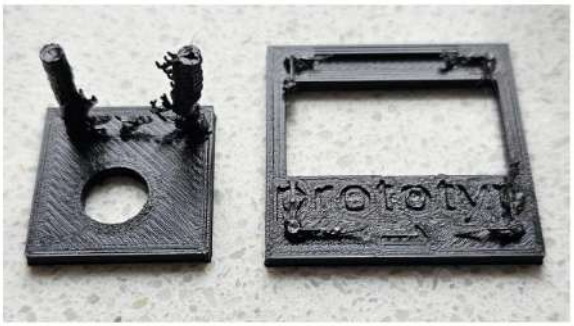
3.3 Druck der Gehäuse

Die von Tom gedruckten Teile für die Gehäuse sind alle 100% Infill und basieren auf den bereitgestellten Modellen.
Änderungen wurden beispielsweise an jeweils einer der Wände für den 4er-Schaltknopf vorgenommen.
An der Wand für die Stromzufuhr ließen wir jeweils eine Aussparung für eine USB-Typ-C Buchse, einen On/Off-Schalter und einen Durchlass für ein Mikro-USB Kabel

zum programmieren des ESP-32.



Während des Druckens sind verschiedene kleinere Komplikationen aufgetreten. Die geplanten Stifte zur Befestigung des SDD1306 Displays und KY-040 Drehschalter und waren für den Drucker zu schmal bzw. Instabil, wodurch sie nicht schnell genug aushärten konnten.



Die Wände sind teilweise zu dick geworden. Da das neue Drucken der Wände nicht mehr möglich war, schliffen wir die inneren Kanten etwas ab, um das Grundgerüst nicht zu stark zu belasten.

3.4 Programmierung der Mikrocontroller

Der Mikrocontroller dient als Bestellterminal mit grafischer Oberfläche auf einem OLED-Display. Die auswählbaren Items werden in einem Array gespeichert (String menuItems[] = {...};) und über Taster kann durch das Menü navigiert werden. Das aktuell ausgewählte Element wird dabei im Display hervorgehoben, beispielsweise durch *display.print("> ");*.

Nach der Auswahl eines Items wechselt das System in den Mengenmodus *(inQuantityMode = true;)*. In diesem Zustand wird die gewünschte Anzahl mithilfe eines Drehencoders angepasst. Die Änderung erfolgt abhängig von der Drehrichtung, etwa durch *quantity++;* bzw. *quantity--;*, wobei der Wert auf einen Bereich von 1 bis 64 begrenzt ist.

Die Hauptlogik basiert auf einer Zustandsabfrage im Loop: *if(inQuantityMode) { handleEncoder(); } else { handleButtons(); }* Dadurch wird klar zwischen Menüführung und Mengeneingabe unterschieden.

Nach der Bestätigung der Auswahl wird ein Sendevorgang ausgelöst *(showSendingScreen();)*. An dieser Stelle erfolgt im vollständigen System die Übergabe der ausgewählten Daten an den MQTT-Server, wodurch andere Geräte die Bestellung empfangen und weiterverarbeiten können. Anschließend kehrt das System wieder in das Hauptmenü zurück.

4. Fazit

Das Projekt entstand im Rahmen des Themas „Cyber-physische Systeme“ und zeigt deutlich, wie sich digitale und reale Komponenten miteinander verbinden lassen. Am Beispiel einer Minecraft-Welt werden Daten erfasst, über ein Netzwerk übertragen und anschließend in der physischen Welt auf Mikrocontrollern und Displays sichtbar gemacht. Damit wird das Prinzip eines cyber-physischen Systems – vernetzte Software- und Hardwarekomponenten, die sich gegenseitig beeinflussen – in einem kleinen, aber funktionsfähigen Setup nachvollziehbar umgesetzt.

Die Ergebnisse sind in Teilen positiv zu bewerten. Die Auslesung der Kisteninhalte aus Minecraft funktioniert stabil, die Daten werden zuverlässig per MQTT an die ESP32-Mikrocontroller übertragen und die Anzeige auf den OLED-Displays ist gut lesbar. Die Steuerung über Taster und Drehencoder ist für die Zielgruppe intuitiv und lässt sich ohne große Einarbeitung bedienen. Die Verwendung von MQTT ermöglicht nahtlose Erweiterung, da so neue Geräte später einfach hinzugefügt werden können, ohne das Gesamtsystem grundlegend umzubauen.

Im Projektverlauf traten jedoch einige Schwierigkeiten auf. Das Projekt startete eigenverschuldet später als angedacht, wobei die mangelnde Zeit besonders beim 3D-Druck und der Hardware-Montage deutlich wurde. Die 3D-Drucker waren zeitweise stark ausgelastet und konnten nicht so schnell arbeiten, wie wir es uns gewünscht hatten. Zudem traten beim Druck einzelner Gehäuseteile Probleme auf: Manche Wände waren zu dick oder zu instabil, Teile wurden zu schnell bewegt oder nicht richtig belüftet, sodass einige Teile unbrauchbar waren. Diese mussten teils nachbearbeitet oder neu gedruckt werden, was zusätzlichen Zeit- und Aufwand verursachte.

Denkbar sind mehrere Verbesserungen und Erweiterungen für künftige Projekte. Die Benutzeroberfläche der Bestellterminals könnte beispielsweise um eine einfache Bestellhistorie oder eine Statusanzeige für laufende Sendevorgänge ergänzt werden. Zudem ließen sich weitere Module hinzufügen, etwa zur Anzeige zusätzlicher Serverstatistiken oder zur Steuerung weiterer Geräte. Die Gehäuse könnten anhand der gemachten Erfahrungen optimiert werden, etwa durch angepasste Wandstärke, besseren Zugang zu Kabeln oder stabilere Befestigungsmöglichkeiten für Display und Encoder. Die Grundlage aus MQTT und modularem Aufbau bietet dabei viel Spielraum, ohne das System komplett neu zu entwerfen.

Insgesamt lässt sich sagen, dass das Projekt eine gute Mischung aus praktischer Hardwarearbeit, Softwareentwicklung und Netzwerkanbindung im Kontext eines cyber-physischen Systems darstellt. Die Ziele im Bereich Programmierung, Datenerfassung aus Minecraft und Hardware-Integration sind erreicht. Die aufgetretenen Probleme mit Zeitmanagement und 3D-Druck haben uns gezeigt, wo wir in zukünftigen Projekten besser planen und organisieren müssen.

Quellenverzeichnis

https://mqtt.org/
https://www.eclipse.org/paho/
https://mosquitto.org/
https://wiki.vg/Plugin_channels
https://hub.spigotmc.org/javadocs/spigot/
https://papermc.io/documentation
https://papermc.io/
https://github.com/wimvds/minecraft-mqtt
https://github.com/radames/Minecraft-ScriptCraft-mqtt-IoT
https://github.com/knolleary/pubsubclient
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
https://randomnerdtutorials.com/getting-started-with-esp32/
https://learn.adafruit.com/monochrome-oled-breakouts/overview
https://github.com/adafruit/Adafruit_SSD1306
https://github.com/adafruit/Adafruit-GFX-Library
https://www.cps.acm.org/
https://cacm.acm.org/research/explorations-in-cyber-physical-systems-education/

Für die Recherche, zur Unterstützung bei der sprachlichen Formulierung sowie zur Überprüfung der Rechtschreibung wurde Künstliche Intelligenz unterstützend eingesetzt.