2023

NACKADEMIN

IoT-System med Temperatur och Luftfuktighets övervakning samt Larmfunktion

MJUKVARUUTVECKLARE, INBYGGDA SYSTEM OCH IOT
MAHADI HASAN
Mahadi.Hasan@yh.nackademin.se

innehåller

Abstract:

Denna akademiska rapport belyser utvecklingen av ett Internet of Things (IoT)-system som har förmågan att mäta temperatur och luftfuktighet med precision genom användning av en DHT-sensor. Dess syfte är att samla in, lagra och övervaka sensorvärden och aktivt larma om de förinställda gränsvärdena överskrids. Lösningen är avsedd för olika tillämpningar som kräver noggrann övervakning av klimatförhållanden, exempelvis inom lagerhantering, livsmedelsindustrin, växthusodling och klimatkontroll i bostäder. Rapporten beskriver detaljerat systemets arkitektur, dess komponenter och implementation.

1. Inledning:

Övervakning av temperatur och luftfuktighet är av central betydelse inom en mängd olika branscher, såsom lagerhantering, livsmedelsindustrin, växthusodling och klimatkontroll i bostäder. Detta gäller särskilt för stora industriella anläggningar som kan sträcka sig över omfattande områden och flera våningar. I sådana miljöer blir behovet av ett effektivt och pålitligt larmsystem som kan skydda hela anläggningen från temperatur- och fuktighetsavvikelser särskilt akut.

Detta projekt ämnar adressera denna utmaning genom att utveckla och implementera ett IoT-baserat system som kan exakt mäta temperatur och luftfuktighet, lagra dessa data i en centraliserad databas och utlösa varningsmeddelanden när fördefinierade tröskelvärden överskrids. Genom att förse industrier och komplexa anläggningar med en sofistikerad och integrerad lösning för temperatur- och fuktighetskontroll kan detta projekt bidra till att minimera risker och förluster.

2. Systemarkitektur:

Systemet består av tre huvudkomponenter: Datainsamling, Databas och Gränsvärdeskontroll och Larmaktivering.

2.1 System 1 - Datainsamling (IoT-enhet):

System 1 är ansvarigt för att samla in och överföra sensordata i realtid och fungerar som den första linjen i datainsamlingen.

Målet med detta system är att bygga ett IoT-system som kan mäta temperatur och luftfuktighet med hög noggrannhet och sedan överföra dessa data till en fjärr-MQTT-brokar för fjärrövervakning och analys. Systemet använder en DHT22-sensor för att samla in data och en ESP32-mikrokontrollerenhet för att hantera Wi-Fi-anslutningen och MQTT-kommunikationen.

Systemkomponenter:

Wi-Fi-anslutning: Systemet använder Wi-Fi för att ansluta till det lokala nätverket och möjliggör kommunikation med den fjärr-MQTT-brokeren.

MQTT-kommunikation: För att överföra temperatur- och luftfuktighetsdata används MQTT-protokollet. En PubSubClient-biblioteket används för att hantera MQTT-kommunikationen.

Temperatur- och Luftfuktighetssensor (DHT22): Sensorn mäter omgivande temperatur och luftfuktighet och levererar värden som kan användas för övervakning och registrering.

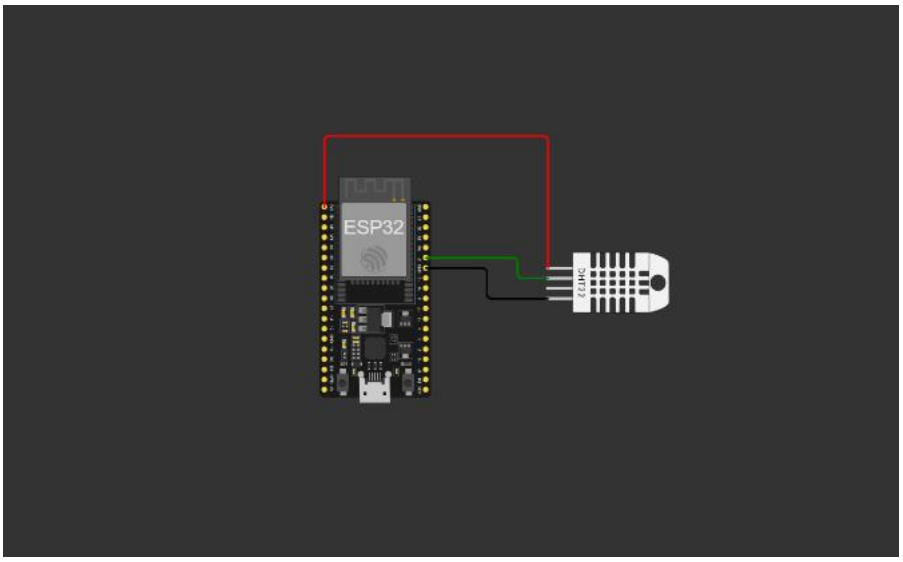
Implementation:

Anslutning till Wi-Fi: ESP32-enheten ansluts till ett Wi-Fi-nätverk genom att användaren anger SSID och lösenord för nätverket. Efter att anslutningen har upprättats, är enheten klar att kommunicera via Wi-Fi.

MQTT-kommunikation: Användaren konfigurerar systemet med adressen till MQTT-brokaren och porten som ska användas. PubSubClient-biblioteket används för att upprätta och hantera anslutningen till MQTT-brokaren.

Datainsamling: En DHT22-sensor används för att mäta temperatur och luftfuktighet. Sensordata hämtas regelbundet, formateras och skickas till MQTT-brokaren via lämpliga ämnen (topics).

Återanslutning: Om anslutningen till MQTT-brokaren går förlorad, försöker enheten återansluta sig kontinuerligt tills anslutningen är upprättad.



2.2 System 2 - Databas och Gränsvärdeskontroll:

Denna centrala komponent tar emot sensordata från System 1 via MQTT brokar och lagrar dem i en databas. Systemet är utrustat med en algoritm för gränsvärdeskontroll, som jämför de inkommande sensordata med fördefinierade tröskelvärden. Om något av de uppmätta värdena överskrider de förinställda gränserna, genereras en larmhändelse. Systemet arkiverar också historiska data för framtida referens och analys.

Det andra systemet i vår arkitektur omfattar tre distinkta komponenter som är utformade för att effektivt hantera databasrelaterade och MQTT-kommunikationsuppgifter. Dessa komponenter möjliggör samlingen och lagringen av temperatur- och luftfuktighetsdata samt överföringen av denna information till och från en MQTT-brokar.

2.2.1 Databasskapande Komponent:

Den första komponenten har som uppgift att etablera och administrera en MySQL-databas som är avsedd för att lagra temperatur- och luftfuktighetsdata. Databasen skapas med en strukturerad tabell för att organisera och bevara data. Genom denna komponent blir det möjligt att säkert lagra historiska mätningar och möjliggör framtida dataanalys och rapportgenerering.

Klass: DataHandler

Klassen är utformad för att utföra följande uppgifter:

Skapa en anslutning till databasen: I konstruktören (init-metoden) skapas en anslutning till MySQL-databasen med hjälp av användarinformation som användarnamn, lösenord och värdnamn (host). Databasnamnet "temperaturer" används i exemplet.

Skapa en tabell: Metoden "create_table" används för att skapa en tabell med namnet "TempData" i databasen. Tabellen har tre kolumner: "Id" (primärnyckel), "Tempc" (temperaturdata) och "humidity" (luftfuktighetsdata).

Sätt in data: Metoden "insert_data" möjliggör insättning av temperatur- och luftfuktighetsdata i tabellen. Data som ska infogas överförs som parametrar till funktionen och sätts in i tabellen med en lämplig SQL-fråga.

Stäng anslutningen: Med hjälp av "close_connection" kan anslutningen till databasen stängas när den inte längre behövs. Detta är viktigt för att frigöra resurser och undvika onödig belastning på databasservern.

Radera tabell: I metoden "drop_table" utförs en SQL-fråga som raderar hela tabellen "TempData". Detta kan vara användbart om det behövs en ren uppsättning för att lagra data.

2.2.2. MQTT-Datainsamlingskomponent:

Den andra delen av systemet är ansvarig för att etablera en anslutning till en fjärr-MQTT-brokar. Den lyssnar på de ämnen (topics) som är konfigurerade för att överföra temperatur- och luftfuktighetsdata från systemet-1s sensorer. Genom att prenumerera på dessa ämnen kan den samla in och läsa data från MQTT-brokaren i realtid. Detta gör det möjligt att hålla den lokala databasen uppdaterad med färska data från sensorerna. Det här delen syftar till att skapa en lösning som effektivt mottar temperatur- och luftfuktighetsdata från en MQTT-brokar, hanterar och utvärderar dessa data och, om det behövs, publicerar en varning tillbaka till MQTT-brokaren. Lösningen består av flera komponenter som samverkar för att skapa en integrerad övervaknings- och hanteringsprocess.

Klassstruktur:

MQTTDataHandler: Detta är huvudklassen som ansvarar för att ansluta till en MQTT-brokar, prenumerera på de aktuella ämnena för temperatur och luftfuktighet, samt hantera inkommande data. Om temperaturen och luftfuktigheten hamnar utanför de förinställda gränsvärdena, aktiveras en varningsprocess.

on_connect: Metoden aktiveras när en anslutning upprättas till MQTT-brokaren och prenumererar på de relevanta ämnena.

on_message: Denna metod hanterar meddelanden som kommer in från MQTT-brokaren och extraherar temperatur- och luftfuktighetsdata.

check_condition: Metoden utvärderar temperatur- och luftfuktighetsdata mot fördefinierade gränsvärden och, om de överskrids, aktiverar en varningspublicering.

start: Huvudloopen som hanterar anslutning till MQTT-brokaren, kontinuerlig mottagning av sensordata, utvärdering av data och hantering av publicering av varningar vid behov.

MQTTDataPublisher: Denna klass är ansvarig för att publicera varningsmeddelanden till MQTT-brokaren om temperatur- eller luftfuktighetsgränsvärdena överskrids.

DataHandler: En extern klass som används för att skapa och hantera en MySQL-databas för lagring av temperatur- och luftfuktighetsdata. Klassen möjliggör även tabellskapande och datainfogning.

Funktionalitet:

När programmet startar, skapar det en databastabell och förbereder sig för att ansluta till en MQTT-brokar för att ta emot sensordata.

Programmet ansluter till MQTT-brokaren och prenumererar på de ämnen som innehåller temperatur- och luftfuktighetsdata.

Vid mottagning av data utvärderas temperatur- och luftfuktighetsvärden mot gränsvärdena. Om dessa värden överskrider gränserna, aktiveras en varningsprocess som publicerar ett meddelande tillbaka till MQTT-brokaren.

Samtidigt registreras temperatur- och luftfuktighetsdata i en MySQL-databas för framtida referens och analys.

2.2.3. MQTT-Datapubliceringskomponent:

Den tredje och sista komponenten är ansvarig för att publicera temperatur- och luftfuktighetsdata tillbaka till MQTT-brokaren. Denna återkoppling till MQTT-brokaren kan vara användbar om andra enheter eller användare vill övervaka sensordata eller om det finns en nödvändighet att utföra fjärrkommandon baserade på de mottagna mätningarna. Genom att publicera data till relevanta ämnen möjliggör denna komponent realtidsåtkomst till sensordata från MQTT-brokaren. Den här delen introducerar en enkel men effektiv lösning för publicering av varningsmeddelanden via MQTT-protokollet. Lösningen består av en egen klass, "MQTTDataPublisher", som är utformad för att ansluta till en MQTT-brokar, skicka varningsmeddelanden till ett specifikt ämne (topic) och därmed möjliggöra en snabb och pålitlig distribution av kritiska meddelanden.

Klassstruktur:

MQTTDataPublisher: Huvudklassen är ansvarig för att hantera anslutningen till en MQTT-brokar och publicera varningsmeddelanden. Klassen är konfigurerbar och tillåter användaren att ange MQTT-brokarens adress, port och ämnet som varningsmeddelanden ska publiceras till.

init: Konstruktormetoden upprättar anslutningen till MQTT-brokaren vid skapandet av ett objekt från klassen.

publish_data: Denna metod används för att publicera varningsmeddelanden till det fördefinierade ämnet. Användaren kan ange det meddelande som ska publiceras som en parameter till metoden.

Funktionalitet:

När en instans av "MQTTDataPublisher" skapas, skapar den en anslutning till en MQTT-brokar med hjälp av angiven adress och port.

Med metoden "publish_data" kan användaren skicka ett varningsmeddelande till ett specifikt ämne (topic) på MQTT-brokaren.

Varningsmeddelanden publiceras på ämnet och är därmed tillgängliga för abonnenter som kan reagera och vidta nödvändiga åtgärder baserat på de publicerade meddelandena.

2.3 System 3 - Larmaktivering (IoT-enhet):

I händelse av en larmhändelse från System 2 är System 3 ansvarigt för att aktivera larmåtgärder. System 3 är en ytterligare IoT-enhet som är ansluten till ett larmsystem, exempelvis ljud- eller ljussignaler, eller som kan skicka meddelanden via nätverket.

Denna enhet tar emot larmmeddelanden från System 2 och genomför de nödvändiga larmåtgärderna för att varna relevanta intressenter. Det här del fokuserar på att utveckla en lösning som kan ta emot larmmeddelanden från en MQTT-brokar och aktivera ett larmsystem med hjälp av en ESP32-mikrokontroller. Huvudmålet är att reagera på larmmeddelanden och aktivera en ljud- eller ljusvarning i en viss tidsperiod för att signalera en händelse. Nedan beskrivs de viktigaste komponenterna och funktionerna i koden:

Kodstruktur:

Wi-Fi-anslutning: Programmet börjar med att ansluta ESP32-enheten till det lokala Wi-Fi-nätverket. Detta möjliggör anslutning till den fjärr-MQTT-brokaren för att ta emot larmmeddelanden.

MQTT-kommunikation: Programmet använder PubSubClient-biblioteket för att etablera anslutning till en fjärr-MQTT-brokar med specificerad adress och port. Det ställer också in en callback-funktion som kommer att köras när ett larmmeddelande mottas.

Callback-funktion: Callback-funktionen utför följande åtgärder när ett larmmeddelande mottas:

Skriver ut ämnet (topic) och meddelandet som mottagits.

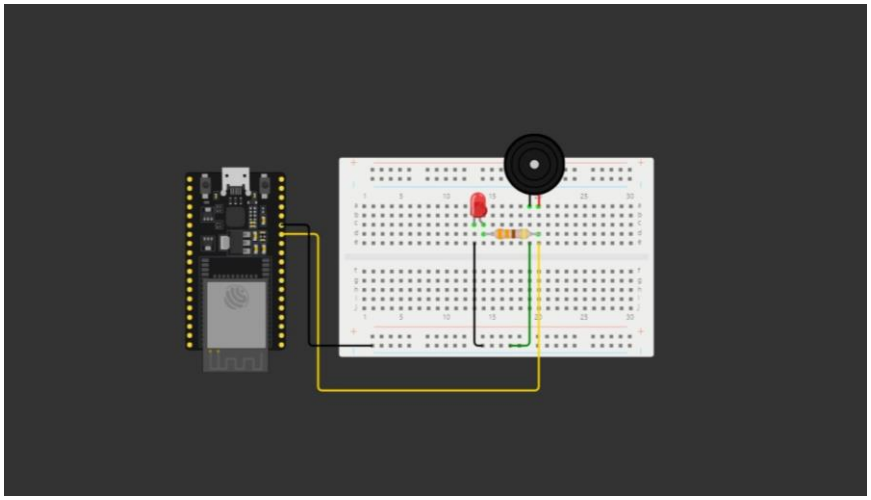
Aktiverar ett anslutet larmsystem genom att ändra statusen på en specifik pin (i detta fall pin 21) till "HÖG" (HIGH).

Väntar i 5 sekunder (5000 ms) för att simulera ett larm.

Återställer statusen på larmsystemet genom att ändra pinnens status till "LÅG" (LOW).

Återanslutning till MQTT-brokaren: Om anslutningen till MQTT-brokaren går förlorad, försöker programmet kontinuerligt återansluta sig tills en framgångsrik anslutning upprättas. En unik klientidentitet skapas för att säkerställa att anslutningen kan göras.

Huvudloop: Huvudloopen övervakar kontinuerligt MQTT-anslutningen genom att kalla "mqttClient.loop()". Dessutom prenumererar programmet på ämnet "/iot22tmos/alarm" för att vara redo att ta emot larmmeddelanden från MQTT-brokaren.



3. Diskussion:

Utvecklingen av detta Internet of Things (IoT)-system möjliggör noggrann mätning och övervakning av temperatur- och luftfuktighetsvärden, med förmågan att varna om fördefinierade gränsvärden överskrids. Lösningen är mångsidig och kan användas inom olika tillämpningsområden, inklusive lagerhantering, livsmedelsindustrin, växthusodling och klimatkontroll i bostäder. Nedan följer en diskussion om systemets styrkor, möjliga förbättringar och dess potentiella användningsområden:

Styrkor:

Precision och Noggrannhet: Systemet använder DHT22-sensorer för temperatur- och luftfuktighetsmätningar, vilket ger hög precision och noggrannhet i datainsamlingen.

Realtidsövervakning: Systemet kan övervaka sensorvärden i realtid och reagera på avvikelser från förinställda gränsvärden omedelbart genom larmaktivering.

Historisk Data: Lösningen sparar temperatur- och luftfuktighetsdata i en databas, vilket möjliggör historisk spårning och dataanalys för att identifiera mönster och trender över tiden.

Anpassningsbarhet: Systemet är skalbart och anpassningsbart för olika användningsområden. Användare kan enkelt ändra tröskelvärden och larmåtgärder enligt sina specifika krav.

Möjliga Förbättringar:

Energiförbrukning: Systemet kan optimeras för att minska energiförbrukningen, särskilt om det används i batteridrivna enheter. Detta kan uppnås genom att implementera sömlöst sömnläge för IoT-enheter när de inte mäter eller överför data.

Säkerhet: Ytterligare lagerskydd och säkerhetsåtgärder kan implementeras för att skydda systemet mot obehörig åtkomst eller dataintrång.

Fjärråtkomst: Möjligheten att fjärr ställa och justera tröskelvärden samt larma på distans skulle vara en värdefull funktion för vissa användningsscenarier.

Potentiella Användningsområden:

Livsmedelsindustrin: Systemet kan användas för att övervaka temperatur- och luftfuktighetsförhållandena i livsmedelslager och produktionsanläggningar, vilket är kritiskt för att säkerställa livsmedelssäkerhet och kvalitet.

Växthusodling: Inom växthusodling kan lösningen hjälpa till att skapa optimala klimatförhållanden för växttillväxt och skördeeffektivitet.

Lagerhantering: Inom lagerhantering och distributionscentrum kan systemet användas för att skydda känsliga varor mot temperatur- och luftfuktighetsavvikelser, vilket minimerar förluster.

Klimatkontroll i Bostäder: Lösningen kan integreras i hemautomationssystem för att övervaka och kontrollera inomhusklimatet, och aktivera luftkonditionering eller värmesystem vid behov.

4. Slutsats:

Utvecklingen av detta IoT-system för temperatur- och luftfuktighetsövervakning erbjuder en mångsidig lösning för att möta behoven inom olika branscher och tillämpningar. Genom noggrann datainsamling, historisk lagring och realtidslarm ger systemet användarna verktyg att hantera och kontrollera kritiska klimatförhållanden. Med ytterligare optimering och anpassningar kan systemet effektivt bidra till att minska risker, förluster och underlätta beslutsfattande baserat på insamlade data. Framtida arbete kan rikta sig mot att förbättra systemets energieffektivitet och säkerhet samt att utforska fler användningsområden och scenarier där temperatur- och luftfuktighetsövervakning är avgörande.