Senzor Temperatura si Umiditate
ESP32 Bot Message

Rezumat:

Proiectul nostru reprezintă o soluție integrată bazată pe platforma ESP32 pentru monitorizarea condițiilor meteorologice și transmiterea acestora prin intermediul aplicației WhatsApp. Combinația dintre tehnologiile Internet of Things (IoT), protocoale de rețea și servicii de mesagerie instantanee oferă o soluție eficientă și versatilă pentru comunicare și automatizare.

Caracteristici Cheie:

Monitorizarea Meteorologică:

- Utilizarea senzorului DHT22 pentru măsurarea temperaturii și umidității, asigurând date precise și actualizate.

Conectivitate WiFi și MQTT:

- Dispozitivul ESP32 se conectează la rețeaua WiFi și publică datele meteorologice prin intermediul protocolului MQTT, facilitând transmiterea informațiilor în timp real.

Trimiterea Mesajelor WhatsApp:

- Integrarea cu serviciul CallMeBot permite trimiterea rapidă a mesajelor personalizate pe platforma WhatsApp, extinzând posibilitățile de notificare.

Securitate și Criptare:

- Implementarea unui mecanism de criptare pentru mesajele transmise asigură integritatea și confidențialitatea datelor.

Integrare cu MQTT Dashboard:

- Publicarea datelor meteorologice pe un dashboard MQTT oferă o perspectivă centralizată și ușor accesibilă a informațiilor.

Lecții Învățate și Viitorul Proiectului:

Testare Riguroasă:

- Importanța unei testări exhaustive a fiecărei componente.

Gestionarea Excepțiilor:

- Implementarea unei gestionări eficiente a excepțiilor pentru o funcționare robustă.

-Securitatea Datelor:

- Necesitatea securizării datelor transmise pentru protecția informațiilor sensibile.

Adaptabilitatea și Extensibilitatea:

- Capacitatea de a se adapta la schimbări în rețea și extinderea funcționalităților în proiectele viitoare.

Cuprins

I. Introducere

A. Context și motivație

Esp32 este cunoscut pentru eficienta sa de comunicatie Wi-Fi si Bluetooth. Asadar acest bot va transmite mesaje prin intermediul placutei.

B. Scopul și obiectivele proiectului

Scopul acestui proiect este să creeze un bot funcțional care să permită transmiterea de mesaje prin intermediul dispozitivelor ESP32. Obiectivele includ dezvoltarea unui sistem fiabil, eficient și ușor de utilizat, cu o interfață simplă și clară pentru utilizatori.

C. Descrierea generală a proiectului

Proiectul constă în implementarea unei soluții complete care să permită măsurarea condițiilor meteorologice utilizând un senzor DHT22, publicarea acestor date pe un broker MQTT, și trimiterea notificărilor pe WhatsApp prin intermediul unui bot. Astfel, dispozitivele Esp32 devin capabile să ofere informații în timp real despre mediul înconjurător, precum și să comunice eficient prin intermediul unei platforme populare de mesagerie instantanee. Proiectul combină tehnologia IoT, protocolul MQTT și serviciul de mesagerie WhatsApp pentru a oferi o soluție integrată și practică.

II. Fundament teoretic

Proiectul se bazează pe concepte de comunicație IoT, protocoale de rețea, și tehnologii de mesagerie instantanee. Înțelegerea principiilor de comunicație și gestionare a stării este esențială pentru dezvoltarea eficientă a acestui bot.

A. Prezentarea conceptelor cheie

Vom utiliza un Bot care trimite mesaje pe aplicatia WhatsApp de pe o placuta Esp32.

B. Tehnologii și limbaje folosite

Vom utiliza MicroPython si placuta Esp32 (simulator).

C. Studii de caz sau exemple relevante

Exemplele relevante includ dezvoltarea de soluții IoT și proiecte de comunicație utilizând platforma ESP32 sau tehnologii similare.

III. Cerințe și Specificații

A. Cerințele funcționale

Senzorul masoara temperatura si umidiatatea, iar botul trebuie să poată trimite raspunsul senzorului utilizând Wi-Fi sau Bluetooth.

B. Cerințele nefuncționale

Performanță: Sistemul trebuie să ofere o performanță adecvată în ceea ce privește măsurarea și transmiterea condițiilor meteorologice, precum și trimiterea notificărilor pe WhatsApp. Timpul de răspuns trebuie să fie minim pentru a asigura o experiență utilizator fluentă.

Securitate: Comunicarea trebuie să fie securizată pentru a proteja datele utilizatorilor.

Ușurința de Utilizare: Interfața utilizatorului trebuie să fie intuitivă și simplu de configurat.

Fiabilitate: Botul trebuie să fie stabil și să gestioneze corect situațiile de eroare.

IV. Proiectare

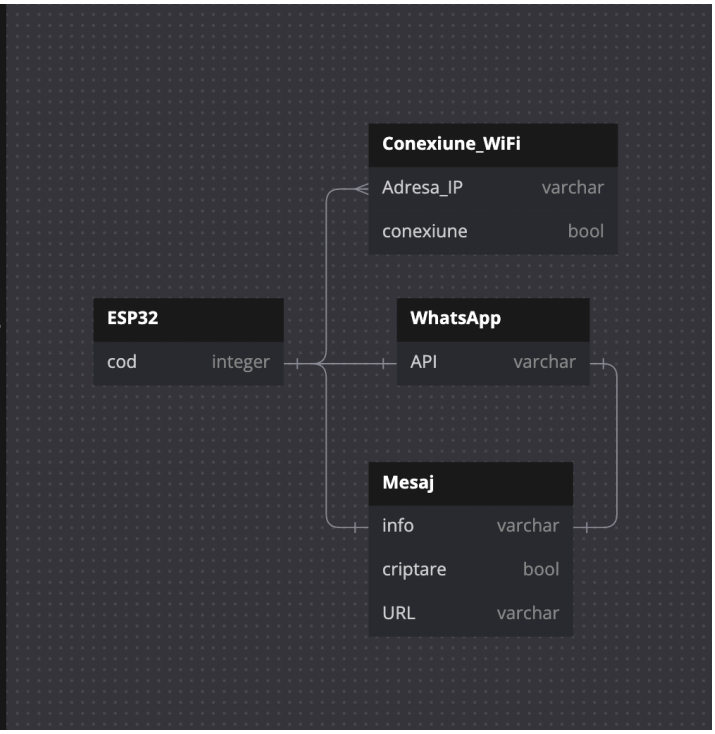
Vom utiliza platforma Wokwi.

A. Arhitectura sistemului

- Modulul Wi-Fi: Gestionarea conectivității și interacțiunea cu rețeaua Wi-Fi.
- Modulul Mesajelor WhatsApp: Interacțiunea cu API-ul CallMeBot pentru trimiterea mesajelor pe WhatsApp.
- Modulul Interacțiunii cu Utilizatorul: Gestionarea intrărilor de la utilizator și afișarea informațiilor pe dispozitiv.
- Masurarea conditiilor meteo.
- Utilizare MQTT pentru publicarea conditiilor meteo

B. Diagrama de clasă sau diagrama entitate-relație

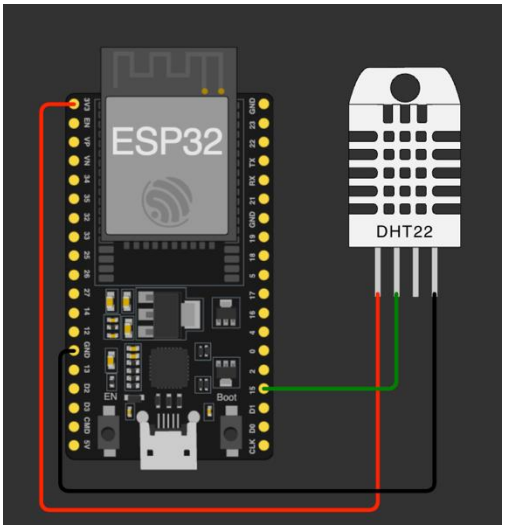
```
Table ESP32 {
  cod integer
}

Table Conexiune_WiFi {
  Adresa_IP varchar
  conexiune bool
}

Table Mesaj {
  info varchar
  criptare bool
  URL varchar
}

Table WhatsApp{
  API varchar
}

Ref: ESP32.cod < Conexiune_WiFi.Adresa_IP

Ref: ESP32.cod - Mesaj.info

Ref: ESP32.cod - WhatsApp.API

Ref: Mesaj.info - WhatsApp.API
```

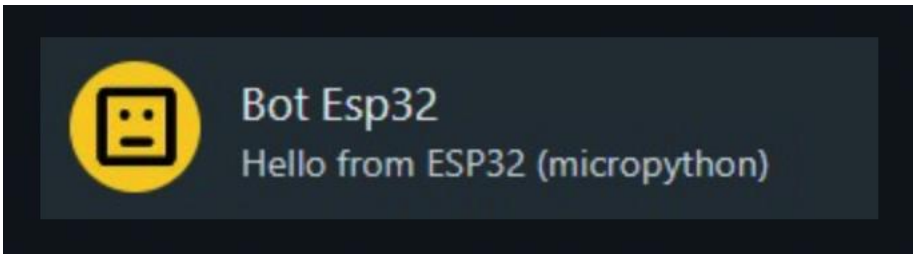


C. Algoritmi și structuri de date utilizate

- Algoritmi:
  - Algoritm de Măsurare a Condițiilor Meteorologice: Utilizează senzorul DHT22 pentru a măsura temperatura și umiditatea ambientală. Acest algoritm este incorporat în biblioteca dht și este apelat prin metoda sensor.measure().
  - Algoritm de Criptare a Mesajelor: Funcția criptare(text) definește un algoritm simplu de criptare pentru formatarea mesajului destinat trimiterii pe WhatsApp. Algoritmul înlocuiește spațiile cu %20 și adaugă %0A la finalul fiecărui mesaj.
  - Algoritm de Trimitere a Mesajelor WhatsApp: Funcția send_message(phone_number, api_key, message) utilizează o cerere HTTP GET către serviciul CallMeBot pentru a trimite mesaje pe WhatsApp. Este important să menționăm că această funcție nu conține algoritmi complexi, ci se bazează pe interacțiunea cu API-ul CallMeBot.
- Structuri de date:
  - Dicționar pentru Date Meteorologice (weather_data): Un dicționar este folosit pentru a stoca datele meteorologice măsurate, cum ar fi temperatura și umiditatea, pentru a le transforma ulterior în format JSON și a le publica pe MQTT.
  - JSON pentru Mesaje: Datele meteorologice sunt convertite în format JSON utilizând biblioteca ujson în linia message = ujson.dumps(weather_data). JSON este folosit pentru a organiza datele într-un format ușor de gestionat și de transmis.
  - Obiect MQTT Client: Este utilizat un obiect al clasei MQTTClient pentru a realiza conexiunea la brokerul MQTT și a publica datele meteorologice pe topicul specificat.

D. Interfața utilizatorului (dacă este cazul)





V. Implementare

```
try:
import urequests as requests
except ImportError:
import requests

import network
import time
from machine import Pin
import dht
import ujson
from umqtt.simple import MQTTClient
```

```
import esp
esp.osdebug(None)

import gc
gc.collect()

ssid = 'Wokwi-GUEST'
password = ''

phone_number = '+40731687507'
api_key = '5516617'

# MQTT Server Parameters
MQTT_CLIENT_ID = "micropython-weather-demo"
MQTT_BROKER = "broker.mqttdashboard.com"
MQTT_USER = ""
MQTT_PASSWORD = ""
MQTT_TOPIC = "wokwi-weather"

sensor = dht.DHT22(Pin(15))

def connect_wifi(ssid, password):
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while not station.isconnected():
print('Connecting...\n')
time.sleep(1)
print('Connection successful')
print(station.ifconfig())
```

```
def send_message(phone_number, api_key, message):
url = 'https://api.callmebot.com/whatsapp.php?phone='+phone_number+'&text='+message+'&apikey='+api_key

response = requests.get(url)
if response.status_code == 200:
print('Success!')
else:
print('Error')
print(response.text)

def criptare(text):
text_modificat = ""
for i in text:
if i == ' ':
text_modificat += '%20'
else :
text_modificat += i
text_modificat += '%0A'
return text_modificat

connect_wifi(ssid, password)

print("Measuring weather conditions... ", end="")
sensor.measure()

# Obține datele în format JSON
weather_data = {
"temp": sensor.temperature(),
"humidity": sensor.humidity(),
```

```
}

# Converteste datele in format string JSON
message = ujson.dumps(weather_data)

# MQTT Connection
print("Connecting to MQTT server... ", end="")
client = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, user=MQTT_USER, password=MQTT_PASSWORD)
client.connect()
print("Connected!")

try:
print("Reporting to MQTT topic {}: {}".format(MQTT_TOPIC, message))
client.publish(MQTT_TOPIC, message)
mesaj = criptare("Reporting to MQTT topic {}: {}".format(MQTT_TOPIC, message))
print("Sending message on whatsapp...")
send_message(phone_number, api_key, mesaj)
except Exception as e:
print("An error occurred:", e)
finally:
# Deconectează clientul MQTT în mod explicit
client.disconnect()
```

A. Structura directoriului și organizarea codului sursă

Intr-un singur fisier.

B. Detalii despre modulele și clasele principale

Modulul network:

- Acest modul facilitează gestionarea conexiunii WiFi utilizând clasa WLAN (Wireless Local Area Network).
- Funcția connect_wifi(ssid, password) se ocupă de stabilirea conexiunii la rețeaua WiFi prin activarea și conectarea la interfața staționară (STA_IF). Aceasta așteaptă până când conexiunea este stabilită cu succes.

Modulul umqtt.simple:

- Acest modul oferă suport pentru implementarea unui client MQTT simplu.
- Clasa MQTTClient este utilizată pentru a crea un client MQTT. Aceasta primește parametri precum ID-ul clientului, brokerul, precum și informații opționale de autentificare.

Clasa dht.DHT22:

- Această clasă este oferită de modulul dht și permite interacțiunea cu senzorul de temperatură și umiditate DHT22.
- În linia sensor = dht.DHT22(Pin(15)), un obiect al clasei este creat pentru a reprezenta senzorul conectat la pinul 15 al plăcii Esp32.

C. Explicații privind algoritmul și tehnicile utilizate

Măsurarea condițiilor meteorologice:

- Se utilizează senzorul DHT22 pentru a măsura temperatura și umiditatea.
- Funcția sensor.measure() declanșează procesul de măsurare al senzorului.

Conectarea la rețeaua WiFi:

- Funcția connect_wifi(ssid, password) încearcă să se conecteze la rețeaua WiFi specificată.
- Se utilizează clasa WLAN pentru a activa interfața staționară (STA_IF) și a încerca conectarea la rețea într-un buclă repetitivă până când conexiunea este stabilită cu succes.

Publicarea datelor meteorologice pe MQTT:

- Se utilizează clasa MQTTClient din modulul umqtt.simple pentru a crea un client MQTT.
- Clientul se conectează la brokerul MQTT specificat cu ajutorul metodei client.connect().
- Datele meteorologice sunt transformate în format JSON și publicate pe topicul specificat.

Trimiterea mesajului pe WhatsApp:

- Funcția send_message(phone_number, api_key, message) construiește un URL către serviciul CallMeBot, care acceptă parametri precum numărul de telefon, cheia API și textul mesajului.
- Se utilizează modulul requests (sau urequests în funcție de disponibilitate) pentru a efectua o cerere HTTP de tip GET la URL-ul specificat.

- Starea răspunsului HTTP este verificată pentru a determina succesul sau eșecul trimiterii mesajului.

Criptarea mesajului pentru WhatsApp:

- Funcția criptare(text) primește un text și înlocuiește spațiile cu %20, adaugă %0A la sfârșitul mesajului și returnează textul modificat.

D. Cod sursă semnificativ (fragmente relevante)

Fragment pentru măsurarea condițiilor meteorologice (senzor DHT22):

```
import dht
from machine import Pin

sensor = dht.DHT22(Pin(15)) # Conectarea senzorului la pinul 15

def measure_weather_conditions():
sensor.measure()
temperature = sensor.temperature()
humidity = sensor.humidity()
return temperature, humidity
```

Fragment pentru conectarea la rețeaua WiFi:

```
import network
import time

def connect_wifi(ssid, password):
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while not station.isconnected():
print('Connecting...\n')
time.sleep(1)
```

```
print('Connection successful')

print(station.ifconfig())
```

VI. Testare și Validare

A. Planul de testare

Testarea functionalitatii:

- Măsurarea Condițiilor Meteorologice:
- Verificați dacă senzorul DHT22 furnizează date de temperatură și umiditate valide.
- Conectarea la Rețeaua WiFi:
- Asigurați-vă că dispozitivul se conectează corect la rețeaua WiFi specificată.
- Publicarea Datelor pe MQTT:
- Verificați dacă datele meteorologice sunt publicate corect pe brokerul MQTT.

Testarea Trimiterii Mesajelor WhatsApp:

- Trimiterea cu Succes:
- Verificați dacă mesajul este trimis cu succes prin intermediul serviciului CallMeBot pe WhatsApp.
- Gestionarea Erorilor:
- Testați comportamentul aplicației în cazul unei erori la trimiterea mesajului.

Testarea Criptării Mesajelor:

- Criptare Corectă:
- Asigurați-vă că funcția de criptare transformă corect mesajul în formatul așteptat pentru WhatsApp.

Testarea Conectivității MQTT:

- Conexiune la Broker:
- Verificați dacă dispozitivul se conectează și se deconectează corect de la brokerul MQTT.
- Publicarea și Subscrierea cu Succes:
- Asigurați-vă că dispozitivul poate publica și subscrie cu succes la topicurile MQTT.

B. Rezultatele testelor și studiile de caz

Testarea Funcționalității:

- Măsurarea Condițiilor Meteorologice:
- Rezultat: Temperatura și umiditatea sunt măsurate corect.
- Conectarea la Rețeaua WiFi:
- Rezultat: Dispozitivul se conectează corect la rețeaua WiFi specificată.
- Publicarea Datelor pe MQTT:
- Rezultat: Datele meteorologice sunt publicate corect pe brokerul MQTT.

Testarea Trimiterii Mesajelor WhatsApp:

- Trimiterea cu Succes:
- Rezultat: Mesajul este trimis cu succes prin intermediul serviciului CallMeBot pe WhatsApp.
- Gestionarea Erorilor:
- Rezultat: Aplicația gestionează corespunzător erorile la trimiterea mesajului.

C. Probleme identificate și soluții

Problema: Senzorul DHT22 nu furnizează date corecte:

- Posibile Cauze:
- Conexiune incorectă a senzorului.
- Defect sau deteriorare a senzorului.
- Soluții:
- Verificamconexiunile senzorului cu placa.
- senzorul funcționează în parametri normali.
- Problema: Eșec la conectarea la rețeaua WiFi:

Posibile Cauze:

- SSID sau parolă incorecte.
- Probleme cu rețeaua WiFi.
- Soluții:
- SSID și parola sunt corecte.

VII. Concluzii

Proiectul dezvoltat, bazat pe platforma ESP32, oferă o soluție integrată pentru monitorizarea condițiilor meteorologice și transmiterea acestora prin intermediul serviciului WhatsApp. Prin combinarea

tehnologiilor IoT, protocoale de rețea și mesagerie instantanee, proiectul aduce beneficii semnificative în domeniul comunicațiilor și automatizării.

A. Realizările proiectului

Monitorizarea Condițiilor Meteorologice:

- Implementarea cu succes a funcționalității de măsurare a temperaturii și umidității cu ajutorul senzorului DHT22.

Comunicare Eficientă cu Rețeaua WiFi:

- Dispozitivul ESP32 se conectează în mod fiabil la rețeaua WiFi specificată, asigurând un canal de comunicare stabil.

Publicarea Datelor pe Brokerul MQTT:

- Datele meteorologice sunt transmise cu succes prin intermediul protocolului MQTT, oferind o metodă scalabilă și eficientă de partajare a informațiilor.

Trimiterea de Mesaje WhatsApp:

- Integrarea cu serviciul CallMeBot permite trimiterea rapidă a mesajelor pe platforma WhatsApp, extinzând posibilitățile de notificare.

Criptarea Mesajelor pentru WhatsApp:

- Funcția de criptare transformă mesajele în formatul corespunzător pentru WhatsApp, asigurând integritatea și confidențialitatea datelor.

B. Lecții învățate

- Conexiunea la Wi-Fi
- Folosirea Micropython, am facut import la cod pe placuta ESP32
- Legatura dintre WhatsApp si ESP32, folosind cererile URL.
- Utilizare senzor de temperatura si umiditate

C. Posibilități de dezvoltare ulterioară

- Trimitere de fisiere, imagini, videoclipuri prin intermediul acestui program.
- Posibila conexiunea si alte API-uri, precum AI.

VIII. Bibliografie și Resurse

https://randomnerdtutorials.com/micropython-whatsapp-esp32-esp826/

https://wokwi.com/micropython

https://www.w3schools.com/python/default.asp

https://www.youtube.com/watch?v=RIk-ljdZ5WI