/\*

Author : Ayush Deo

\*/

#include <WiFi.h> #include <HTTPClient.h> #include <ArduinoJson.h> #include <Wire.h> #include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h> #include <WiFiUdp.h>

#include <NTPClient.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels #define SCREEN\_HEIGHT 64 // OLED display height, in pixels

| #define | OLED\_RESET | -1 | // Reset pin # (or -1 if sharing |
| --- | --- | --- | --- |
| Arduino | reset pin) |  |  |

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

const char\* ssid = "JioFiber-DbXEB"; const char\* password = "sweethome@123"; const char\* city = "Nagpur";

const char\* apiKey = "57b28134f5959e62384d0639ff8112ae";

const char\* newsApiKey = "70cacf7ab0b0498ca6d6ee89b0c06a6e"; // Replace with your news API key

unsigned long lastDisplayChangeTime = 0;

const unsigned long displayChangeInterval = 4000; // Interval between display change in milliseconds

bool displayWeather = true; // Variable to track which data to display

bool stopDisplay = false; // Variable to track whether to stop the cycling of displays

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800); // UTC+5:30 for Indian Standard Time

void setup() {

Serial.begin(115200); // Initialize serial communication Wire.begin(21, 22); // SDA, SCL for OLED screen

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C); // Address 0x3C for 128x64

display.display(); // Display initialization delay(2000); // Pause for 2 seconds

display.clearDisplay(); // Clear the buffer display.setTextColor(SSD1306\_WHITE); // Set text color to

white

// Connect to WiFi WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) { delay(500);

Serial.print(".");

}

Serial.println("\nWiFi connected");

// Initialize NTP client timeClient.begin();

timeClient.setTimeOffset(19800); // UTC+5:30 for Indian Standard Time

// Fetch initial weather data getWeatherData();

}

void loop() {

unsigned long currentMillis = millis();

// Calculate time elapsed since the last display change unsigned long timeElapsed = currentMillis -

lastDisplayChangeTime;

// Check if it's time to switch the display

if (timeElapsed >= displayChangeInterval && !stopDisplay) {

// Toggle between weather, Bitcoin price, and clock display if (displayWeather) {

getWeatherData();

delay(3000); // Wait for 3 seconds before switching to the next display

displayWeather = false;

} else { getBitcoinPrice();

delay(3000); // Wait for 3 seconds before switching to the next display

getNewsHeadlines(); delay(3000);

displayClock();

delay(3000); // Wait for 3 seconds before switching back to weather display

displayWeather = true;

}

// Update last display change time lastDisplayChangeTime = currentMillis;

}

// Update time from NTP server timeClient.update();

// Check for serial input

if (Serial.available() > 0) { char input = Serial.read(); if (input == 's') {

stopDisplay = true; // Stop display cycling

Serial.println("Display cycling stopped. Press 'g' to resume.");

while (Serial.available() > 0) {

Serial.read(); // Consume any remaining characters

}

} else if (input == 'g') {

stopDisplay = false; // Resume display cycling

Serial.println("Display cycling resumed."); while (Serial.available() > 0) {

Serial.read(); // Consume any remaining characters

}

}

}

}

void getWeatherData() { HTTPClient http;

// Construct the URL for weather data

String weatherUrl = ["http://api.openweathermap.org/data/2.5/weather?q=";](http://api.openweathermap.org/data/2.5/weather?q)

weatherUrl += city; weatherUrl += "&appid="; weatherUrl += apiKey;

display.clearDisplay(); // Clear the display buffer

display.setTextSize(2); // Set text size display.setCursor(0, 0); // Set cursor position

Serial.print("Sending HTTP GET request to weather API: "); Serial.println(weatherUrl);

// Send GET request for weather data http.begin(weatherUrl);

int httpCode = http.GET();

// Check for a successful request if (httpCode > 0) {

// Check for a successful response if (httpCode == HTTP\_CODE\_OK) {

// Parse JSON data DynamicJsonDocument doc(1024);

deserializeJson(doc, http.getString());

// Extract relevant data

float temperature = doc["main"]["temp"].as<float>() -

273.15; // Convert Kelvin to Celsius

float humidity = doc["main"]["humidity"].as<float>();

// Print weather data display.print("Temp: "); display.print(temperature, 1); display.println("C");

display.print("Humidity: "); display.print(humidity, 1); display.println("%"); display.display(); // Display on OLED Serial.println("Weather Data:"); Serial.print("Temperature: "); Serial.print(temperature); Serial.println("°C"); Serial.print("Humidity: "); Serial.print(humidity); Serial.println("%"); Serial.println();

} else {

Serial.print("HTTP request to weather API failed with error code: ");

Serial.println(httpCode);

}

} else {

Serial.println("Failed to connect to weather server.");

}

// Close connection http.end();

}

void getBitcoinPrice() { HTTPClient http;

// Construct the URL for Bitcoin price

String bitcoinUrl = "https://api.coindesk.com/v1/bpi/currentprice.json";

display.clearDisplay(); // Clear the display buffer

display.setTextSize(2); // Set text size display.setCursor(0, 0); // Set cursor position

display.println("Bitcoin Price:");

Serial.print("Sending HTTP GET request to Bitcoin API: "); Serial.println(bitcoinUrl);

// Send GET request for Bitcoin price http.begin(bitcoinUrl);

int httpCode = http.GET();

// Check for a successful request if (httpCode > 0) {

// Check for a successful response if (httpCode == HTTP\_CODE\_OK) {

// Parse JSON data DynamicJsonDocument doc(1024);

deserializeJson(doc, http.getString());

// Extract Bitcoin price

float price = doc["bpi"]["USD"]["rate\_float"];

// Print Bitcoin price display.print("$"); display.print(price, 2); display.display(); // Display on OLED Serial.println("Bitcoin Price:"); Serial.print("Price (USD): $"); Serial.println(price);

} else {

Serial.print("HTTP request to Bitcoin API failed with error code: ");

Serial.println(httpCode);

}

} else {

Serial.println("Failed to connect to Bitcoin server.");

}

// Close connection http.end();

}

void getNewsHeadlines() { HTTPClient http;

// Construct the URL for news headlines

String newsUrl = ["http://newsapi.org/v2/top-headlines?country=in&apiKey=";](http://newsapi.org/v2/top-headlines?country=in&apiKey)

newsUrl += newsApiKey;

display.clearDisplay(); // Clear the display buffer

display.setTextSize(2); // Set text size display.setCursor(0, 0); // Set cursor position

Serial.print("Sending HTTP GET request to news API: "); Serial.println(newsUrl);

// Send GET request for news headlines http.begin(newsUrl);

int httpCode = http.GET();

// Check for a successful request if (httpCode > 0) {

// Check for a successful response if (httpCode == HTTP\_CODE\_OK) {

// Parse JSON data DynamicJsonDocument doc(1024);

deserializeJson(doc, http.getString());

// Extract news headlines

JsonArray articles = doc["articles"]; for (JsonObject article : articles) {

String title = article["title"].as<String>(); display.println(title);

}

display.display(); // Display on OLED Serial.println("News Headlines:"); for (JsonObject article : articles) {

String title = article["title"].as<String>(); Serial.println(title);

}

} else {

Serial.print("HTTP request to news API failed with error code: ");

Serial.println(httpCode);

}

} else {

Serial.println("Failed to connect to news server.");

}

// Close connection http.end();

}

void displayClock() {

display.clearDisplay(); // Clear the display buffer

display.setTextSize(2); // Set text size display.setTextColor(SSD1306\_WHITE); // Set text color to

white

// Get current time from NTP client

unsigned long epochTime = timeClient.getEpochTime();

time\_t currentTime = (time\_t)epochTime; // Convert unsigned long to time\_t

struct tm \*timeInfo;

timeInfo = localtime(&currentTime);

// Extract hours, minutes, seconds, day, month, and year int hours = timeInfo->tm\_hour;

int minutes = timeInfo->tm\_min; int seconds = timeInfo->tm\_sec; int day = timeInfo->tm\_mday;

int month = timeInfo->tm\_mon + 1; // tm\_mon is 0-based

int year = timeInfo->tm\_year + 1900; // tm\_year is years since 1900

// Display date on the first line display.setCursor(0, 0); // Set cursor position display.print(year);

display.print("-"); if (month < 10) {

display.print("0");

}

display.print(month); display.print("-"); if (day < 10) {

display.print("0");

}

display.println(day);

// Display actual time on the third line display.setTextSize(2); // Increase text size for time display.setCursor(0, 30); // Set cursor position for time display.print(hours < 10 ? "0" : ""); display.print(hours);

display.print(":"); display.print(minutes < 10 ? "0" : ""); display.print(minutes); display.print(":"); display.print(seconds < 10 ? "0" : ""); display.print(seconds); display.display(); // Display on OLED

// Print to serial monitor Serial.print("Date: "); Serial.print(year); Serial.print("-");

Serial.print(month < 10 ? "0" : ""); Serial.print(month);

Serial.print("-"); Serial.print(day < 10 ? "0" : ""); Serial.println(day); Serial.print("Time: ");

Serial.print(hours < 10 ? "0" : ""); Serial.print(hours); Serial.print(":"); Serial.print(minutes < 10 ? "0" : ""); Serial.print(minutes); Serial.print(":"); Serial.print(seconds < 10 ? "0" : ""); Serial.println(seconds);

}



