Code 1: Connecting to Wi-Fi and Pinging the Server

#include <WiFi.h>

#include <Ping.h> // Include the Ping library

const char\* ssid = "Your\_SSID"; // Your Wi-Fi SSID

const char\* password = "Your\_PASSWORD"; // Your Wi-Fi password

const char\* pingAddress = "8.8.8.8"; // Google DNS Server (or any other server)

int maxReconnectAttempts = 3; // Maximum number of Wi-Fi connection attempts

int timeoutSeconds = 5; // Timeout for each Wi-Fi connection attempt (in seconds)

void setup() {

Serial.begin(115200);

// Try connecting to Wi-Fi

if (!connectToWiFi()) {

Serial.println("Failed to connect to Wi-Fi after maximum attempts.");

return;

}

// Ping the server to check internet connectivity

Serial.println("Pinging the server...");

if (Ping.ping(pingAddress, 3)) { // Send 3 ping requests

Serial.println("Ping successful! Internet is available.");

} else {

Serial.println("Ping failed. Internet is not available.");

handleError(); // Handle the error gracefully

}

}

void loop() {

// Main program loop

}

// Function to connect to Wi-Fi

bool connectToWiFi() {

int attempts = 0;

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED && attempts < maxReconnectAttempts) {

Serial.print("Attempt ");

Serial.print(attempts + 1);

Serial.println(" to connect...");

unsigned long startTime = millis();

// Wait for Wi-Fi connection or timeout

while ((WiFi.status() != WL\_CONNECTED) && (millis() - startTime < timeoutSeconds \* 1000)) {

delay(500); // Check connection status every 500 ms

}

if (WiFi.status() == WL\_CONNECTED) {

Serial.println("Connected to Wi-Fi!");

Serial.print("IP Address: ");

Serial.println(WiFi.localIP());

return true;

} else {

Serial.println("Connection timed out, retrying...");

WiFi.disconnect();

WiFi.begin(ssid, password); // Retry Wi-Fi connection

attempts++;

}

}

return WiFi.status() == WL\_CONNECTED;

}

// Function to handle errors

void handleError() {

Serial.println("Error: Unable to connect to the Internet.");

// Add additional error handling here, such as:

// - Retry mechanism

// - Logging the error

// - Restarting the device

// - Sending a failure signal to another system

}

Code 2: Checking for OTA Updates

#include <HTTPClient.h>

#include <Update.h>

const char\* updateServerURL = "http://your-server.com/firmware"; // URL to check for OTA updates

void setup() {

Serial.begin(115200);

// Check for OTA updates

if (checkForOTAUpdate(updateServerURL)) {

Serial.println("OTA update available, starting the update process...");

if (downloadAndApplyOTA("http://your-server.com/firmware.bin")) { // URL to download firmware

Serial.println("OTA update completed successfully!");

} else {

Serial.println("Failed to apply OTA update.");

}

} else {

Serial.println("No OTA update available.");

}

}

void loop() {

// Main program loop

}

// Function to check for OTA updates

bool checkForOTAUpdate(const char\* updateServerURL) {

HTTPClient http;

http.begin(updateServerURL); // Begin HTTP request to check for updates

int httpCode = http.GET();

if (httpCode == HTTP\_CODE\_OK) {

String payload = http.getString();

Serial.println(payload); // Print the response from the server

// Look for a specific keyword in the response (indicating new update)

if (payload.indexOf("new\_version\_available") != -1) {

return true; // New version available

}

} else {

Serial.println("Failed to check for updates.");

}

http.end();

return false; // No update found

}

// Function to download and apply OTA update

bool downloadAndApplyOTA(const char\* firmwareURL) {

WiFiClient client;

HTTPClient http;

// Start downloading the firmware

http.begin(client, firmwareURL);

int httpCode = http.GET();

if (httpCode == HTTP\_CODE\_OK) {

int contentLength = http.getSize();

if (contentLength > 0) {

// Start the OTA update process

if (Update.begin(contentLength)) {

size\_t written = Update.writeStream(http.getStream());

if (written == contentLength) {

Serial.println("Firmware written successfully!");

if (Update.end()) {

if (Update.isFinished()) {

Serial.println("Update completed successfully!");

return true; // Firmware applied successfully

} else {

Serial.println("Update failed!");

}

}

} else {

Serial.println("Firmware write error.");

}

} else {

Serial.println("Not enough space for OTA update.");

}

}

} else {

Serial.println("Failed to download firmware.");

}

http.end();

return false; // Download or update failed

}