**1. Setting Up the Wi-Fi Module**

#include <WiFi.h>

#include <WiFiClient.h>

#include <WiFiServer.h>

// Wi-Fi credentials

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

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

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

}

void loop() {

// Your main code here

}

**Explanation**: This code includes the necessary libraries for Wi-Fi functionality, sets up the Wi-Fi credentials, and attempts to connect to the specified network. The setup() function initializes the serial communication and starts the Wi-Fi connection. The loop() function can be used for the main logic of the program.

**2. Capturing and Processing Video Frames**

This code captures video frames from the camera module and processes them for transmission.

#include <Camera.h>

// Initialize the camera

Camera camera;

void setup() {

Serial.begin(115200);

camera.begin();

}

void loop() {

// Capture a frame

CameraFrame frame = camera.capture();

// Process the frame (e.g., compress)

processFrame(frame);

// Transmit the frame

transmitFrame(frame);

}

void processFrame(CameraFrame& frame) {

// Implement your compression algorithm here

}

void transmitFrame(CameraFrame& frame) {

// Implement your transmission logic here

}

**Explanation**: This code initializes the camera module and captures video frames in the loop() function. The processFrame() function is a placeholder for implementing a compression algorithm, and the transmitFrame() function handles the logic for transmitting the processed frame over the network.

**3. Transmitting Video Data**

This code snippet demonstrates how to transmit the processed video data over the Wi-Fi network using the TCP protocol.

WiFiClient client;

const char\* serverIP = "192.168.1.100";

const int serverPort = 8080;

void transmitFrame(CameraFrame& frame) {

if (client.connect(serverIP, serverPort)) {

client.write(frame.data, frame.size);

client.stop();

} else {

Serial.println("Connection to server failed");

}

}

**Explanation**: This code establishes a TCP connection to a server with the specified IP address and port. It then sends the processed video frame data to the server and closes the connection. If the connection fails, it prints an error message to the serial monitor.

**4. Optimizing Power Consumption**

This code snippet demonstrates how to put the microcontroller into a low-power mode when not actively processing or transmitting data.

void enterLowPowerMode() {

// Code to enter low-power mode

// This will vary depending on the microcontroller used

LowPower.sleep();

}

void loop() {

// Capture and process frame

CameraFrame frame = camera.capture();

processFrame(frame);

transmitFrame(frame);

// Enter low-power mode

enterLowPowerMode();

}

**Explanation**: This code puts the microcontroller into a low-power sleep mode after capturing, processing, and transmitting a video frame.