Wireless Arduino Control via Mobile: Eliminating the Need for a Dedicated Wireless Communicator, Leveraging Laptop Connectivity

Sumit Chowdhury[1]

[1]Student, Department of Electronics & Tele-Communication Engineering, National Institute of Electronics & Information Technology Agartala, Tripura, India

---***---

Abstract – *In this paper we will study how we can establish wi-fi connectivity with Arduino to control its behavior without using the dedicated hardware module like (ESP8266, ESP32) The aim of this project is to create a bridge between the Arduino and smart phone over wi-fi for triggering the Arduino pins. The bridge will be the combination of software and hardware (laptop for this case), the software is a simple python script acts as a communication bridge between a mobile device and an Arduino via a laptop, facilitating wireless control of the Arduino through the mobile device by relaying commands and responses over a serial connection.*

Key Words: **Wi-fi, Arduino, Python, Script, Serial connection**

1. INTRODUCTION

The Arduino Uno, built around the ATmega328P microcontroller, is a widely adopted open-source platform recognized for its simplicity, versatility, and robustness. With an accessible user interface provided by the Arduino IDE and a rich set of I/O pins, it serves as a foundational tool for prototyping and developing diverse embedded systems and IoT applications. Its popularity stems from its ease of use, making it a cornerstone in microcontroller-based development for both educational and professional purposes.

The ESP8266 was introduced in 2014, and the ESP32 followed later in 2016. Despite their advanced features, these modules may have cost disadvantages and be unsuitable for simple prototypes due to additional expenses and space requirements. However, these challenges can be addressed by utilizing software-based wireless control, offering a cost-effective and space-efficient alternative for basic Arduino projects.

Now in order to implement software based wireless control we used a python script which will plays a pivotal role in implementing a software-based wireless control system for Arduino projects. Rather than relying on dedicated wireless hardware modules such as the ESP8266 or ESP32, the script leverages the laptop's connectivity to establish a communication link with a mobile device. Operating as a bridge, the script facilitates the seamless exchange of commands and responses between the mobile device and the Arduino through a serial connection. This approach not only reduces costs associated with additional hardware but also circumvents potential space constraints, making it a practical and efficient solution for enabling wireless control in Arduino prototypes. The script's ability to unify software and hardware components underscores its significance in simplifying and optimizing the wireless communication aspect of Arduino projects, particularly in scenarios where a lightweight and economical solution is desirable.

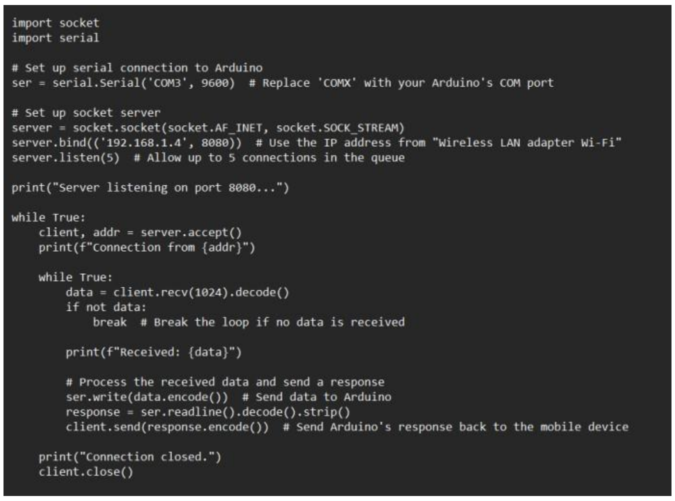
2. THE PYTHON SCRIPT

The provided Python script orchestrates a comprehensive communication framework, establishing a bridge between a mobile device and an Arduino through the intermediary of a laptop. At its core, the script leverages the `serial` module to initialize a serial connection with the Arduino, thereby establishing a conduit for two-way data exchange. Concurrently, a socket server is instantiated utilizing the `socket` module, with an assigned IP address and port, fostering a platform for wireless communication between the laptop and the mobile device.

The script unfolds within a nested loop structure, with the outer loop persistently awaiting incoming connections. Upon connection establishment, the script seamlessly transitions into an inner loop designed to continuously manage the exchange of data. Incoming data from the mobile device undergoes processing within the script before being dispatched to the Arduino through the established serial connection. The reciprocal journey of data involves the Arduino generating a response, which is subsequently transmitted back to the mobile device.

A pivotal feature of the script lies in its adept management of connections. The inner loop gracefully terminates when no further data is received, effectively signaling the completion of a data exchange cycle. Subsequently, the connection is closed, and the outer loop recommences its vigilant stance, anticipating new incoming connections.

In essence, this script embodies a sophisticated yet accessible framework, enabling seamless and wireless control of the Arduino from a mobile device through the intermediation of a laptop. The convergence of serial communication with the Arduino and socket networking within a Python environment underscores the script's capacity to harmonize hardware interactions and networking functionalities, thereby encapsulating a versatile and integrative solution.



```
import socket
import serial

# Set up serial connection to Arduino
ser = serial.Serial('COM3', 9600)  # Replace 'COMX' with your Arduino's COM port

# Set up socket server
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('192.168.1.4', 8080))  # Use the IP address from "Wireless LAN adapter Wi-Fi"
server.listen(5)  # Allow up to 5 connections in the queue

print("Server listening on port 8080...")

while True:
    client, addr = server.accept()
    print(f"Connection from {addr}")

    while True:
        data = client.recv(1024).decode()
        if not data:
            break  # Break the loop if no data is received

        print(f"Received: {data}")

        # Process the received data and send a response
        ser.write(data.encode())  # Send data to Arduino
        response = ser.readline().decode().strip()
        client.send(response.encode())  # Send Arduino's response back to the mobile device

    print("Connection closed.")
    client.close()
```

Figure 1: Python Script Overview

2.1 Serial and Socket modules

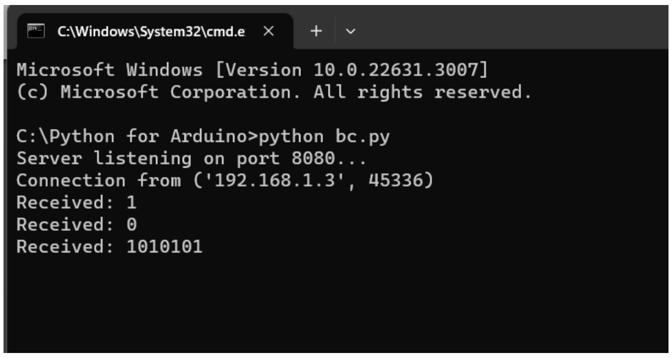
In the presented Python script, the `serial` module is utilized to establish a serial connection with an Arduino microcontroller, enabling bidirectional communication. This connection is essential for transmitting data between the Python script and the Arduino. The code specifies the COM port (`COM3`) and sets a baud rate (`9600`) to determine the speed of data exchange.

Concurrently, the `socket` module is employed to create a socket server, establishing a network communication channel for wireless interaction. The server is bound to a specific IP address (`192.168.1.4`) and port (`8080`), facilitating communication with a mobile device. The server is configured to listen for incoming connections with a queue size of 5.

The primary server loop continuously accepts connections, and upon connection establishment, an inner loop manages the reception of data from the mobile device. This data is then processed and sent to the Arduino through the serial connection. The Arduino's response is subsequently transmitted back to the mobile device over the established socket connection. This orchestration of the `serial` and `socket` modules forms the foundation for wireless control of the Arduino via a laptop, providing a seamless bridge between the mobile device and the microcontroller.

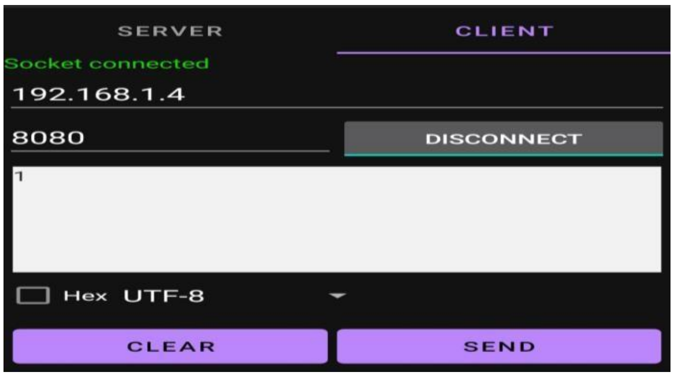
2.2 Display of the received data

The Python script creates a connection between a laptop, a mobile device, and an Arduino. As data flows in from the mobile device, the script displays it on the command prompt screen, making it easy to see what the mobile device is sending. Simultaneously, the script communicates this data to the Arduino and shows the Arduino's response on the screen. This real-time display helps monitor the wireless communication, aiding in understanding and debugging the process.



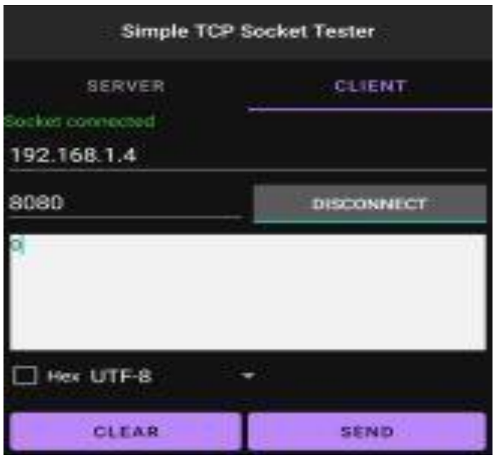
```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Python for Arduino>python bc.py
Server listening on port 8080...
Connection from ('192.168.1.3', 45336)
Received: 1
Received: 0
Received: 1010101
```

Figure 2: Displaying of Data getting received (1-0-1010101).



SERVER CLIENT
Socket connected
192.168.1.4
8080 DISCONNECT
1
Hex UTF-8
CLEAR SEND

Figure 3: Sending data (1) from mobile through [Simple TCP Socket tester App] over wifi.



Simple TCP Socket Tester
SERVER CLIENT
Socket connected
192.168.1.4
8080 DISCONNECT
0
Hex UTF-8
CLEAR SEND

Figure 4: Sending data (0) from mobile through [Simple TCP Socket tester App] over wifi.

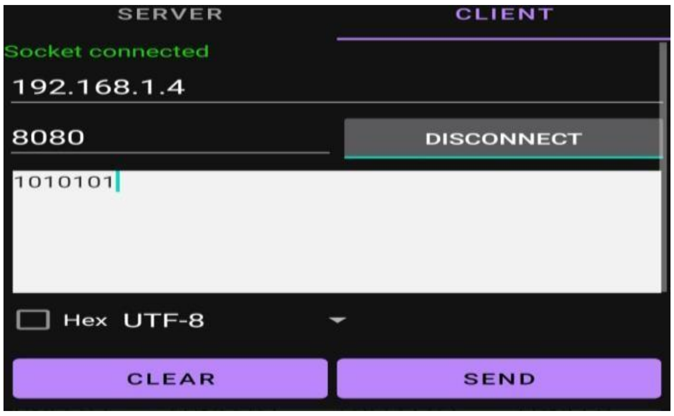


Figure 5: Sending data (1010101) from mobile through [Simple TCP Socket tester App] over wifi.

3. INTERFACING ARDUINO WITH MOBILE

Here in this section, we will demonstrate the setup of Arduino with mobile via laptop we consider connecting Arduino with a mobile device to control a LED on pin 7 introduces a practical and hands-on dimension to the interface. This setup allows users to remotely turn the LED on or off using their mobile device, providing a tangible example of the wireless control capability.

By leveraging the Arduino and mobile interface, individuals can manipulate the LED state through a user-friendly mobile application or a simple communication protocol. This interaction not only showcases the immediate control of a physical device but also serves as a foundational step for more complex projects. The blend of Arduino's hardware capabilities and mobile device interactivity empowers users to extend control beyond traditional physical interfaces, offering a glimpse into the exciting possibilities of combining microcontrollers with mobile technology.

3.1 Connection of Arduino with Laptop

Connecting an Arduino to a laptop via USB is a straightforward process. Using a USB cable, we physically link the Arduino board to one of the laptop's USB ports. This establishes a direct communication channel between the two devices, enabling the laptop to send and receive data to and from the Arduino.

Once connected, the laptop recognizes the Arduino as a USB device, and we can program the Arduino, upload code, and even communicate with it using the Arduino IDE or other programming environments. The USB connection serves as both a power source for the Arduino and a data link, facilitating seamless interaction between the laptop and the microcontroller for various projects and programming tasks.

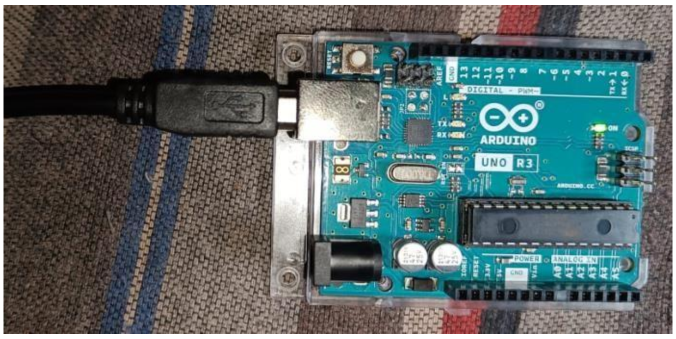
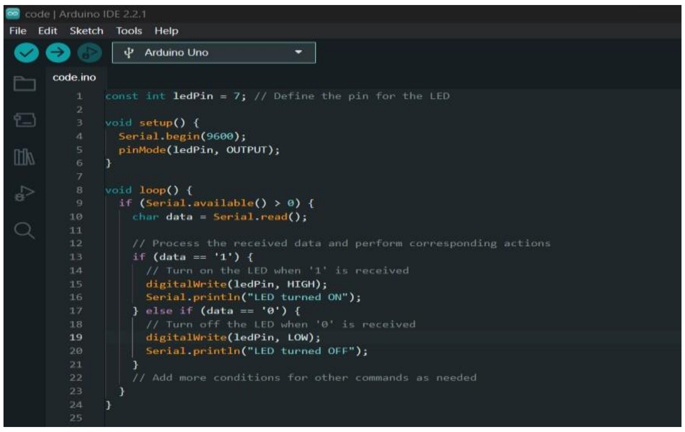


Figure 6: Setting up Arduino and connecting it with laptop via USB.

3.2 Setting up the Arduino IDE

The Arduino IDE is a user-friendly software tool for programming Arduino microcontrollers. It includes a code editor, compiler, and uploader, simplifying the process of writing and uploading code to Arduino boards. The IDE supports C and C++ languages, and its features like the Serial Monitor and Library Manager enhance the development experience. Its simplicity makes it accessible for beginners, while its versatility caters to experienced developers for prototyping various electronic projects.

This Arduino code enables wireless control of an LED connected to pin 7. It utilizes serial communication to receive commands. When '1' is received, the LED turns on, and a corresponding message is sent. Conversely, '0' turns off the LED, accompanied by a confirmation message. This simple yet effective code forms the basis for wirelessly managing the LED, providing a concise solution for remote control applications.



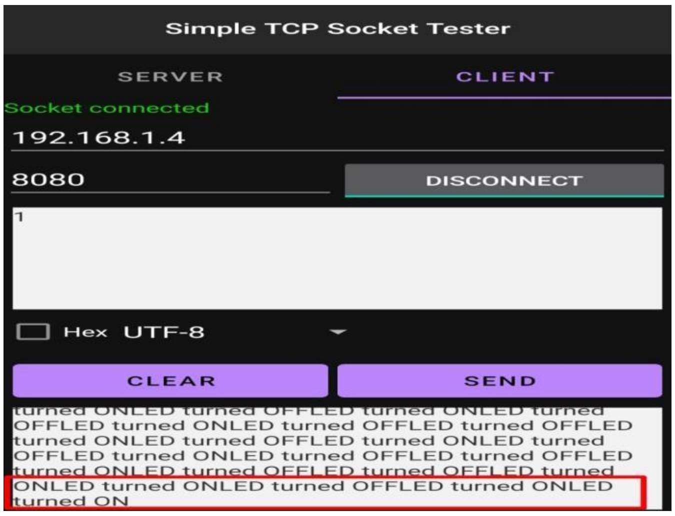
```
const int ledPin = 7; // Define the pin for the LED

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    char data = Serial.read();

    // Process the received data and perform corresponding actions
    if (data == '1') {
      // Turn on the LED when '1' is received
      digitalWrite(ledPin, HIGH);
      Serial.println("LED turned ON");
    } else if (data == '0') {
      // Turn off the LED when '0' is received
      digitalWrite(ledPin, LOW);
      Serial.println("LED turned OFF");
    }
    // Add more conditions for other commands as needed
  }
}
```

Figure 7: Overview of Arduino code in IDE

3.3 Controlling Arduino though mobile APP

Controlling the Arduino with mobile devices involves using wi-fi and TCP Tester app. For this case we will consider a LED connected to the Digital pin 7 of Arduino and we will trigger this using the TCP Tester App wirelessly.
Here, the server is configured to listen on port 8080 for incoming TCP connections. The choice of port 8080 is a common practice for custom or alternative web services. In this context, it serves as the communication gateway between the Python script and any connecting clients. The server continuously listens for incoming data on this specific port, and upon establishing a connection, it processes the data, communicates with an Arduino microcontroller via a serial connection, and sends back responses. Port 8080 acts as the designated channel through which the server interacts with external clients, facilitating the exchange of information and control commands in the project. This specific port assignment aligns with conventions for web services and ensures a standardized point of access for communication between devices within the local network.

Integrating a simple TCP socket app with an Arduino for LED control involves enhancing the Python script to interpret commands sent from the client. The client, represented by the TCP socket app, can be designed with a straightforward interface allowing users to send commands like '0' to turn the LED off and '1' to turn it on. Upon receiving these commands, the Python script on the server side processes the data within its continuous loop. Depending on whether '0' or '1' is received, the script triggers the corresponding action to turn the LED off or on, respectively. This integration establishes a seamless communication channel between the app and the Arduino, enabling users to wirelessly control the LED by sending commands through the TCP socket connection. It provides a tangible example of how simple user inputs from the app can translate into physical actions on the Arduino-controlled hardware.



```
Socket connected
192.168.1.4
8080
1
UTF-8
turned ONLED turned OFFLED turned ONLED turned
OFFLED turned ONLED turned OFFLED turned OFFLED
turned ONLED turned OFFLED turned ONLED turned
OFFLED turned ONLED turned OFFLED turned OFFLED
turned ONLED turned OFFLED turned OFFLED turned
ONLED turned ONLED turned OFFLED turned ONLED
turned ON
```

Figure 8: Sending statement 1 to turn on the LED at pin 7.

As it received statement 1 from the mobile app (simple TCP socket tester) the LED is ON as shown in the figure 9.

Also figure 8 shows the response of Arduino displaying LED ON state in the red highlighted box.

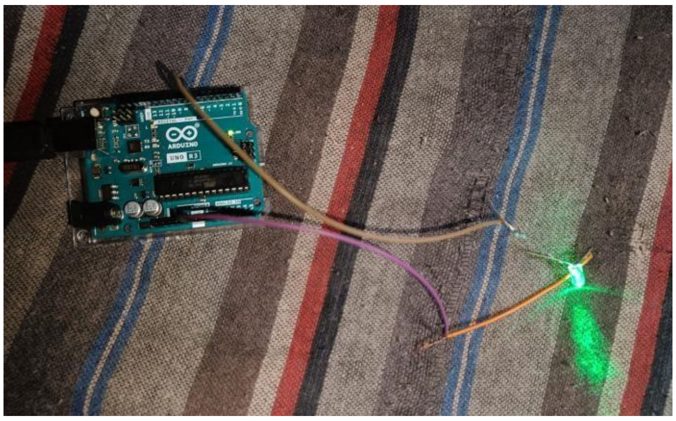
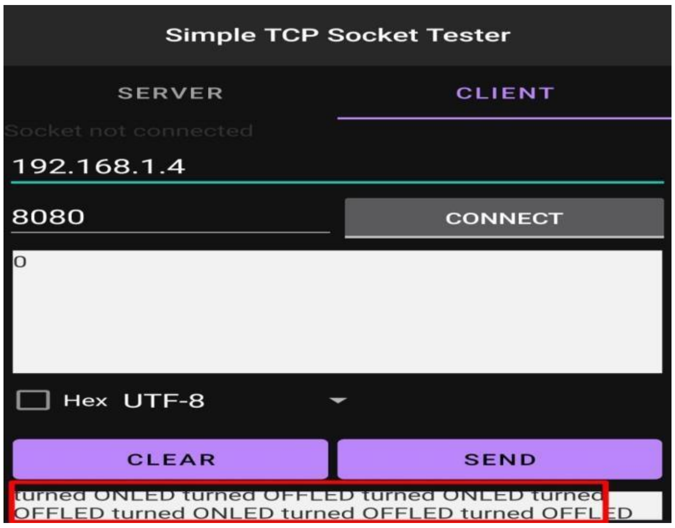


Figure 9: LED is turned ON upon receiving statement **1** from mobile APP.



```
Socket not connected
192.168.1.4
8080
0
UTF-8
turned ONLED turned OFFLED turned ONLED turned
OFFLED turned ONLED turned OFFLED turned OFFLED
```

Figure 10: Sending statement 0 to turn off the led at pin 7

Similarly, as it received statement 0 from the mobile app (simple TCP socket tester) the LED is OFF as shown in the figure 11.

Also figure 10 shows the response of Arduino displaying LED OFF state in the red highlighted box.

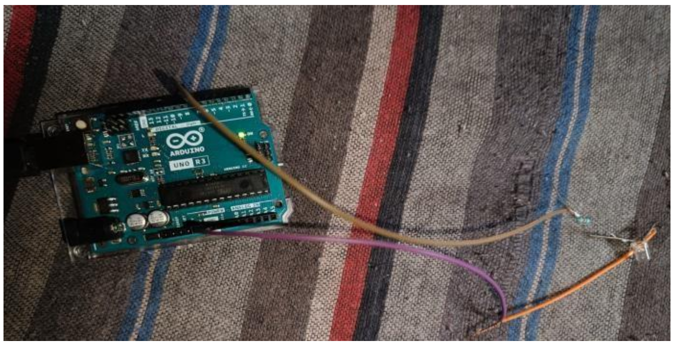


Figure 11: LED is turned OFF upon receiving statement **0** from mobile APP

4. CONCLUSIONS

In conclusion, the project "Wireless Arduino Control via Mobile" presents a streamlined and cost-effective IoT solution by utilizing laptop connectivity. By eliminating the need for a dedicated wireless communicator, the project showcases a practical approach to wireless Arduino control. The integration of a simple TCP socket app facilitates seamless communication, making it a versatile and accessible solution for various IoT applications. This project not only demonstrates the efficiency of leveraging existing resources but also underscores the potential for widespread adoption of such solutions in the evolving landscape of IoT implementations**.** As of this paper we have tested the project with led but we can also extend this and it can be used to trigger various devices like relay modules , actuators and various devices as per the requirements.

REFERENCES

[1] Exploring Arduino: Tools and Techniques for Engineering Wizardry Published by John Wiley & Sons, Inc. 10475 Crosspoint Boulevard Indiananolis. IN 46956|.

[2] Programming IoT Projects with ESP32 and ESP8266" by Neil Kolban.

[3] Make: Getting Started WithArduino - The Open SourceElectronics PrototypingPlatform Massimo Banziand Michael ShilohShroff/Maker Media; Third edition (27December 2014).

[4] K Building Smart Homes with Raspberry Pi Zero, ESP8266, and Arduino" by Marco Schwartz.

BIOGRAPHIES

Sumit Chowdhury is a student of National Institute of Electronics & Information Technology Agartala, Tripura, India. Currently he is pursuing final year UG Diploma in Electronics & Tele-communication Engineering Department. He's Research Interests are Electrical circuit Design, Embedded System Design & Optimization and Electronic Device (study).