# Arduino und ESP Display, drahtlos ohne Cloud

|  |  |
| --- | --- |
| Material | Kosten |
| ESP32 oder Arduino mit Bluetooth modul |  |
| Neopixel Strip |  |
| 1 x Taster |  |
| Kabel |  |

Auf der Maker-Fair Ruhr ist euch vielleicht das LED-Autorennen aufgefallen (auch von Johannes in seinem Video vorgestellt). Ich finde das eine tolle Idee und dachte mir, dass es auch ein gutes Anwendungsbeispiel für die App RemoteXY ist. RemoteXY erlaubt es Daten aus einer App auf einen Arduino oder ESP per Bluetooth oder Wifi zu schicken und von diesem Daten zu empfangen. Anders als beispielsweise bei Blynk benötigt man dafür keine Cloud Anbindung, die Daten werden entweder direkt per Bluetooth zwischen dem Handy und dem ESP ausgetauscht oder per Wifi über den lokalen Router. Mit RemoteXY ist so ein Dashboard für eine vereinfachte Version des Autorennens mit Speed-Graph, Rundenzähler und Runden – Gong entstanden.

## Grundprinzip und Basiskonfiguration

Das Prinzip von RemoteXY ist schnell erklärt, über den Editor auf <https://remotexy.com/en/editor/> konfiguriert man zunächst auf welche Art kommuniziert werden soll, zur Auswahl stehen Bluetooth, Wifi, USB, Ethernet und Cloud-server. Anschliessend wählt man einen der unterstützten Mikrokontroller (diverse Boards werden unterstützt, darunter Arduino, ESP8266 und ESP32). Dann folgt die Wahl des Kommunikationsmoduls, z.B. Integrated Bluetooth beim ESP32 oder ein externes HM-10 Modul für die BLE Verbindung mit einem Arduino.

A screenshot of a computer

Description automatically generated

Figure 1 Auswahl von Mikrokontroller und Kommunikation (Im Beispiel: Bluetooth, ESP32 und Arduino IDE)

Ist die Hardware und Kommunikation definiert, beginnt man das Display mit Schalter- und Anzeigeelementen zu gestalten. Für jedes Element kann ein Variabelnamen definiert werden, den man später im Programmcode verwendet. Über den Button „Get Source Code“ erhält man dann das Skelett des Programms, es enthält bereits die Anweisung zum Import aller relevanten Libraries, das Layout des Displays in Form eines Arrays und die Variablen in einer Struktur.

Auf dem Handy muss man die App RemoteXY installieren, die sowohl für Android als auch für iOS verfügbar ist. Startet man die App, kann über das Plus-Zeichen ein neues Gerät hinzugefügt werden, je nach Kommunikationsmethode muss man beispielsweise einen Access-Point oder eine IP-Adresse angeben. Im Falle einer Bluetooth-Verbindung muss der Mikrokontroller eingeschaltet werden, die App scannt dann die Bluetoothsignale des Mikrontrollers und kann diesen per Klick hinzufügen.

## Gestaltung des Displays für das Autorennen

Für das Dashboard des Autorennens benötigen wir folgende Elemente:

Eingaben (Controls):

* Feld für gewünschte Rundenzahl (Edit field), Variable: edit\_Runden
* Start-Button (Button), Variable: button\_Go
* Reset-Button (Button), Variable: button\_Reset

Anzeigen (Indications):

* Aktuelle Runde (Text string), Variable: text\_Runde
* Aktuelle Dauer des Rennens (Text string), Variable: text\_Zeit
* Graph der Geschwindigkeit über dem Zeitverlauf (Online graph), Variable: graph\_Speed

Zudem kann man mit der Frame – Komponente unter Decoration, die Aufteilung des Bildschirms sinnvoll gestalten. Das fertige Design sieht mal in Figure 2 Gestaltung des Dashboards und Variabeldefinition

A screenshot of a computer

Description automatically generated

Figure 2 Gestaltung des Dashboards und Variabeldefinition

## Übertragung der Konfiguration auf den ESP und Test der Kommunikation

Hat man das Design und die Konfiguration(!) kontrolliert, gelangt man über den Button „Get source code“ das Grundgerüst des C-Programms. In dem wilden Zahlen – Array, der in der Variablen „uint8\_t RemoteXY\_CONF[]“ gespeichert ist, versteckt sich das ganze Display – Layout, es wird von der Mobile-App entsprechend interpretiert. In der Struktur RemoteXY (siehe Figure 3 Variablendefinition für das Display ) sind alle im Layout definierten Variablen definiert.

A screenshot of a computer program

Description automatically generated

Figure 3 Variablendefinition für das Display

Für einen ersten Test soll beim Drücken des Button „Go“ die aktuelle Zeit angezeigt werden und beim Drücken des Button „Reset“ soll diese auf Null zurückgesetzt werden. In der void loop() routine des Programmcodes wird zunächst über RemoteXY\_Handler () der aktuelle Zustand der Bedienelemente der App abgefragt bzw. Werte für die Anzeige in der App werden übertragen. Den Status des Go- Button fragt man über die Variable RemoteXY.button\_Go ab. Über RemoteXY.text\_Zeit kann man dem Zeit-Textfeld einen Wert zuweisen, folgende Programmzeile nimmt den Wert der Variable elapsedTime , interpretiert ihn als UnsignedLong und schreibt ihn als String in die Variable RemoteXY.text\_Zeit.

snprintf(RemoteXY.text\_Zeit, sizeof(RemoteXY.text\_Zeit), "%lu", elapsedTime);

Der Programmablauf ist wie folgt:  
1. Speicherung der Startzeit wenn der Button Go gedrückt wurde und der Timer noch nicht läuft

2. Errechnung der abgelaufenen Zeit durch Subtraktion der aktuellen Zeit von der gespeicherten Startzeit

3. Speicherung der abgelaufenen Zeit in der Variablen RemoteXY.text\_Zeit zur Darstellung in der App

4. Prüfung ob der Reset-Button gedrückt wurde, in dem Fall wird die timerRunning Variable auf False gesetzt und 0 in das Textfeld geschrieben.

## Programmierung des vereinfachten Spiels und Verdrahtungsschema

## Alternative Kommunikationsverbindung per WLAN