# yArduino 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.

Figure 4 Mainloop Test Programm zeigt die Hauptschleife des Programms. A computer screen shot of a program code

Description automatically generated

Figure 4 Mainloop Test Programm

In der void loop() routine des Programmcodes wird zunächst über



der aktuelle Zustand der Bedienelemente der App abgefragt bzw. Werte für die Anzeige in der App werden übertragen.

Über die IF-Abfrage wird anschliessend der Status des Go- Button abgefragt (Variable RemoteXY.button\_Go).

Über die Variable RemoteXY.text\_Zeit kann man dem Zeit-Textfeld einen Wert zuweisen, die Zeile   


nimmt den Wert der Variable elapsedTime , interpretiert ihn als UnsignedLong und schreibt ihn als String in die Variable RemoteXY.text\_Zeit.

Die darauf folgende IF-Abfrage prüft ob der Reset-Button in der App gedrückt wurde (RemoteXY.button\_Reset == true) und setzt dann die Variable timerRunning zurück auf false somit muss erst wieder der Go-Button gedrückt werden, bevor der Timer von 0 beginnt.

## Programmierung des vereinfachten Spiels und Verdrahtungsschema

## Alternative Kommunikationsverbindung per Cloud Server

Wie beschrieben, kann die Kommunikation ganz ohne Cloud-Verbindung erfolgen, möchte man aber einen ESP in grosser Entfernung steuern oder sogar nur einen über Wokwi simulierten ESP funktioniert dies über die Cloud server Anbindung. Im folgenden Beispiel lese ich den Kompass des Mobiltelefons aus und steuere Anhand des Kompasswertes einen auf Wokwi simulierten Neopixel LED-Ring mit 16 LEDs.

Damit der RemoteXY Cloud server genutzt werden kann, muss man ein Benutzerkonto anlegen und dann unter dem Menüpunkt MyTokens einen neuen Token erstellen. Figure 5 zeigt die Beispielkonfiguration. Der eigene Token muss unter Settings (unten rechts im Bild) ausgewählt werden. Als einziges Element habe ich Compass unter Sensors auf der linken Seite ausgewählt und den voreingestellten Variablennamen nicht verändert.

A screenshot of a phone

Description automatically generated

Figure 5 Display und Konfiguration für Kompass Auswertung und Cloud derver Verbindung

Wie Wokwi zur Simulation von Mikrocontroler genutzt werden kann haben wir in Make xy beschrieben, ich habe einen standard ESP32 ausgewählt und dann den Pin DIN eines Neopixel Ring and den Pin 27 des ESP32 angeschlossen (s.a. Figure 7). Zudem habe ich im Tab Library Manager die beiden Libraries RemoteXY und Adarfruit NeoPixel hinzugefügt (s.a. Figure 6).

A screenshot of a computer

Description automatically generated

Figure 6 ESP Bibliotheken

In Figure 7 wird in den Zeilen 11 und 12 der Wifi Zugang für den simulierten ESP konfiguriert wird. In den Zeilen 13-15 wird der Cloud-Server Zugang konfiguriert, in Zeile 17 muss noch der Token eingetragen werden, den ihr über RemoteXY zuvor generiert habt.

A computer screen shot of a computer

Description automatically generated

Figure 7 ESP Verkabelung und Sketch