Christian Wiegert

Arduino Workshop – Grundlagen, Elektrotechnik und Anwendungsbeispiele

Einführung: Arduino, Elektrotechnik und Programmierung

Arduino bezeichnet zweierlei: eine Programmiersprache und einen Mikrocontroller – ein elektronisches Bauteil, auf dem Code läuft. Der Vorteil: Man kann die Fähigkeiten von Computern nutzen und sie in die physische Welt einbinden. Der Nachteil: Man bewegt sich immer gleichzeitig auf mehreren Schauplätzen – Elektrotechnik, Programmierung und Bauteilwissen. Fehler können in jedem dieser Bereiche liegen, und oft sucht man stundenlang nach einem Codefehler, während einfach ein Kabel nicht richtig eingesteckt ist.

Das Ziel des Workshops ist nicht das schnelle Ergebnis per KI-generiertem Code, sondern das Grundverständnis: Was mache ich da eigentlich gerade? Wer die Basics sicher beherrscht, findet Fehler schneller und kann Projekte gezielt aufbauen.

Ressourcen und Lernquellen

Hilfreiche Quellen für die eigene Arbeit mit Arduino: die offizielle Arduino-Website mit Referenz und Beispielen in der Arduino IDE; Instructables für vollständige Tutorials mit Schaltplan und Code; YouTube; ChatGPT und ähnliche KI-Tools als Unterstützung beim Code – aber mit dem Bewusstsein, dass man den Code dann auch lesen und verstehen muss, um Fehler zu finden. Eigenbezeichnung: Copycat Skills – aus allem, was man findet, Sachen zusammenkopieren und verstehen, wo man weiteren Input findet.

Erste Übung: LED, Breadboard und Taster

Das Breadboard (Steckplatine) ermöglicht das Aufbauen von Schaltungen ohne Löten. Die Leiterbahnen verlaufen in zwei Richtungen – das Verständnis dieser Struktur ist die häufigste erste Hürde. Aufgabe: Eine LED zum Leuchten bringen, indem Pluspol und Minuspol der Knopfzellenbatterie mit den richtigen Beinchen der LED verbunden werden.

Als Erweiterung: einen Taster einbauen. Taster haben vier Pins, obwohl sie elektrisch nur ein Kabel sind, das unterbrochen werden kann. Zwei Pins sind dauerhaft verbunden, zwei nicht. Sicher funktioniert immer die Diagonalverbindung. Das Verständnis des Tasters ist entscheidend für alle späteren Projekte: Man kann einen Taster durch Arduino ersetzen – und damit das Schalten automatisieren.

Spannung, Widerstände und Parallel-/Reihenschaltung

Zwei Batterien mit je 1,5 Volt in Reihe ergeben 3 Volt (Spannung addiert sich). Parallel geschaltet bleibt die Spannung gleich, aber die Kapazität (Ampere) verdoppelt sich. LEDs in Reihe benötigen mehr Spannung; LEDs parallel leuchten gleich hell, entladen aber die Batterie schneller.

Widerstände schützen LEDs vor zu hoher Spannung: Bei 3 Volt (Knopfzelle) ist kein Widerstand nötig; bei 5 Volt (Arduino) ist er zwingend, sonst brennt die LED durch. Widerstand funktioniert wie ein langes, dünnes Kabel – er bremst den Stromfluss und schützt empfindliche Bauteile.

Anwendungsbeispiele: Bewässerungssysteme und Sensoren

Arduino kann als intelligenter Schalter verstanden werden: Er erkennt, ob ein Sensor einen bestimmten Wert meldet, und schaltet daraufhin einen Ausgang an oder aus. Klassisches Beispiel: Ein Feuchtigkeitssensor im Boden meldet Trockenheit → Arduino aktiviert eine Pumpe. Dieses Grundprinzip lässt sich beliebig erweitern: Ein Wasserstandssensor verhindert, dass die Pumpe trocken läuft; eine SMS-Funktion via SIM-Karten-Shield informiert den Nutzer, wenn das Reservoir leer ist; eine IoT-Verbindung ermöglicht die Fernsteuerung per Smartphone.

Ein weiteres Beispiel: Ein Solar-Tracker mit zwei Fotosensoren vergleicht die Lichtstärke auf beiden Seiten und dreht einen Servomotor in die hellere Richtung. Sehr simple Bauteile, elegante Logik. Wichtig: Oft gibt es mehrere Wege zum gleichen Ziel – Pumpe oder Hahn, Arduino oder reiner Transistorschaltkreis. Das Verstehen der Alternativen hilft bei der Auswahl der richtigen Lösung.

Arduino als programmierbarer Schalter – die IDE

Im Kern macht Arduino genau das, was vorher der Taster gemacht hat: Es empfängt ein Signal an einem Pin und gibt ein Signal an einem anderen Pin aus. Die Arduino IDE (Integrated Development Environment) ist der Texteditor, in dem der Code geschrieben und auf den Mikrocontroller übertragen wird. Grundbefehle: digitalRead / digitalWrite für digitale Signale (an/aus); analogRead für Sensorsignale mit Abstufungen. Mit diesem Wissen lassen sich die meisten Alltagsprojekte umsetzen.
