Nun wissen Sie, wie Sie Zufallszahlen erzeugen können. Als Nächstes wollen wir diese neuen Kenntnisse gleich in die Tat umsetzen, indem wir einen elektronischen Würfel konstruieren.

Projekt Nr. 15: Einen elektronischen Würfel erstellen

Unser Ziel besteht darin, willkürlich eine von sechs LEDs zum Leuchten zu bringen, um den Wurf eines Würfels zu simulieren. Dazu erzeugen wir eine Zufallszahl zwischen 1 und 6 und schalten dann die LED der entsprechenden Nummer ein. Für die Aufgabe, eine der sechs LEDs zufällig auszuwählen und eine bestimmte Zeit lang aufleuchten zu lassen, erstellen wir eine Funktion. Wenn der Arduino, auf dem der Sketch läuft, eingeschaltet oder zurückgesetzt wird, soll er eine bestimmte Zeit lang in schneller Folge zufällige LEDs einschalten und den Lichtwechsel dann verlangsamen, bis die endgültige LED leuchtet. Die LED, die der Zufallszahl entspricht, bleibt dann eingeschaltet, bis der Arduino zurückgesetzt oder ausgeschaltet wird.

Die Hardware

Um den Würfel zu bauen, brauchen Sie folgende Teile:

- sechs LEDs beliebiger Farbe (LED1 bis LED6)
- einen 560-Ω-Widerstand
- eine mittelgroße Steckplatine
- Verbindungsdrähte
- Arduino und USB-Kabel

Der Schaltplan

Da immer nur eine LED auf einmal leuchtet, reicht ein Widerstand zur Strombegrenzung zwischen den Kathoden der LEDs und dem *GND*-Anschluss. Abbildung 6–2 zeigt den Schaltplan für den Würfel.

John Boxall, Arduino-Workshops, dpunkt.verlag, ISBN 978-3-86490-106-5

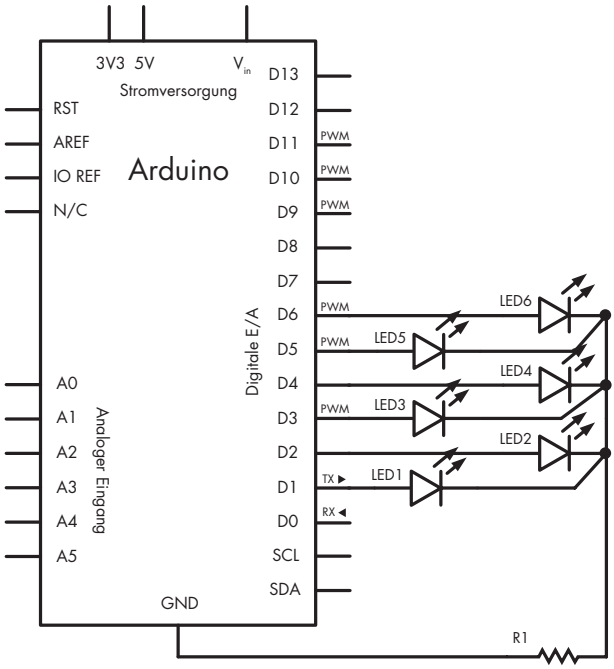

Abb. 6–2 *Schaltplan für Projekt 15*

Der Sketch

Der Sketch für den Würfel sieht wie folgt aus:

```
// Projekt 15: Einen elektronischen Würfel erstellen

void setup()
{
   randomSeed(analogRead(0));          // Gibt den Anfangswert für die
                                       // Erzeugung der Zufallszahlen vor
   for ( int z = 1 ; z < 7 ; z++ )     // Legt die LEDs an den Pins 1-6
                                       // als Ausgänge fest
   {
      pinMode(z, OUTPUT);
   }
}

void randomLED(int del)
{
   int r;
   r = random(1, 7);          // Generiert eine Zufallszahl zwischen 1 und 6
   digitalWrite(r, HIGH);     // Legt Spannung an die entsprechende LED
                              // an einem der Digitalpins 1–6 an
```

```
    if (del > 0)
    {
       delay(del);          // Lässt die LED die übergebene
                            // Verzögerungszeit ❶  lang leuchten
    }
    else if (del == 0)   ❷
    {
       do                   // Lässt die LED dauerhaft leuchten,
                            // wenn die Verzögerung 0 übergeben wird
       {}
       while (1);        ❸
    }
    digitalWrite(r, LOW);   // Schaltet die LED aus
}

void loop()
{
    int a;
    // Bringt willkürliche LEDs zum Leuchten, damit der Vorgang
    // spannender aussieht
    for ( a = 0 ; a < 100 ; a++ )
    {
       randomLED(50);
    }
    // Verlangsamt den Leuchtrhythmus
    for ( a = 1 ; a <= 10 ; a++ )   ❹
    {
       randomLED(a * 100);
    }
    // Hält bei der LED an, die der endgültigen Zufallszahl entspricht
    randomLED(0);
}
```

Um die digitalen Ausgabepins in `void setup()` zu aktivieren, verwenden wir hier eine Schleife. Die Funktion `randomLED()` nimmt einen Integerwert entgegen, der in der Funktion `delay()` bei ❶ dazu herangezogen wird, die LED die angegebene Zeit lang eingeschaltet zu lassen. Beträgt der bei ❷ empfangene Wert für die Verzögerung `0`, hält die Funktion die LED unbegrenzt am Leuchten, denn bei ❸ steht:

```
do {} while (1);
```

Da 1 stets 1 ist, wird diese Schleife endlos durchlaufen.

Um zu »würfeln«, setzen wir den Arduino zurück, womit der Sketch neu startet. Damit das willkürliche Aufleuchten der LEDs immer langsamer wird, bis schließlich der endgültige Wert angezeigt wird, lassen wir

John Boxall, Arduino-Workshops, dpunkt.verlag, ISBN 978-3-86490-106-5

bei ❹ zunächst 100 Mal jeweils eine zufällig ausgewählte LED 50 ms lang leuchten. Danach verlangsamen wir diesen Rhythmus, indem wir die Verzögerung zwischen dem Aufblitzen der Leuchtdioden von 100 auf 1000 ms erhöhen und die Leuchtdauer auf 100 ms. Damit simulieren wir die Verlangsamung der Würfelbewegung, bevor er zum Stillstand kommt und den endgültigen Wert anzeigt. An dieser Stelle gibt der Arduino das Ergebnis des Würfelvorgangs wieder, indem er mit der folgenden letzten Zeile eine der LEDs zum Leuchten bringt:

```
randomLED(0);
```

Den Sketch ändern

Dieses Projekt können Sie auf verschiedene Weise abändern. Beispielsweise können Sie weitere sechs LEDs hinzufügen, um zwei Würfel auf einmal zu werfen, oder das Ergebnis durch die Anzahl der Blinkvorgänge der eingebauten LED anzeigen. Lassen Sie Ihre Fantasie spielen, um mit Ihren neuen Fertigkeiten Spaß zu haben!

Schnellkurs in Binärzahlen

Die meisten Kinder lernen das Zählen im Dezimalsystem (Basis 10), Computer (und der Arduino) aber verwenden das Binärsystem (Basis 2). *Binärzahlen* werden nur aus den Ziffern 1 und 0 zusammengesetzt, z. B. 10101010. Jede der Stellen steht dabei für 2^n, wobei *n* die Positionsnummer ist (die Nummerierung verläuft von rechts nach links und beginnt mit 0). Die Produkte aus dem Inhalt und der Zweierpotenz der einzelnen Stellen werden dann addiert, was den Gesamtwert der Zahl ergibt.

Betrachten Sie als Beispiel die Binärzahl 111111 in Tabelle 6–1. Um sie in eine Dezimalzahl umzurechnen, müssen wir die dezimalen Entsprechungen der einzelnen Stellen addieren, die in der unteren Zeile der Tabelle aufgeführt sind:

```
128 + 64 + 32 + 16 + 8 + 4 + 2 + 1
```

Das Ergebnis lautet 255. Eine Binärzahl mit acht Stellen (oder *Bits*) enthält ein *Byte* Daten. Ein Byte Daten kann einen numerischen Wert zwischen 0 und 255 haben. Das Bit ganz links wird als das *signifikanteste Bit* (Most Significant Bit, MSB) bezeichnet, das ganz rechts als das *am wenigsten signifikante Bit* (Least Significant Bit, LSB).

Binärzahlen eignen sich hervorragend für die Speicherung ganz bestimmter Arten von Daten, z. B. der Ein/Aus-Muster für LEDs, Wahr/falsch-Einstellungen und für den Status digitaler Ausgänge. Sie bilden das Grundformat aller Arten von Daten in Computern.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	1	1	1	binär
128	64	32	16	8	4	2	1	dezimal

Tab. 6–1 *Beispiel für die Umrechnung von Binär- in Dezimalzahlen*

Bytevariablen

Eine Möglichkeit zur Speicherung von Binärzahlen besteht in der Verwendung von *Bytevariablen*. Der folgende Beispielcode legt die Bytevariable `outputs` an:

```
byte outputs = B11111111;
```

Das `B` weist den Arduino an, die nachfolgende Zahl als Binär- und nicht als Dezimalzahl zu lesen (also als binäre Entsprechung von 255 und nicht als einen Wert von mehr als elf Millionen). Mit Listing 6–2 können Sie das noch weiter veranschaulichen.

```
byte a;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   for ( int count = 0 ; count < 256 ; count++ )
   {
      a = count;
      Serial.print("Base-10 = ");
      Serial.print(a, DEC);                ❶
      Serial.print(" Binary = ");
      Serial.println(a, BIN);              ❷
      delay(1000);
   }
}
```

Listing 6–2 *Veranschaulichung von Binärzahlen*

John Boxall, Arduino-Workshops, dpunkt.verlag, ISBN 978-3-86490-106-5

Über die Funktion `Serial.print()` geben wir den Inhalt der Bytevariablen mit `DEC` als Dezimalzahl aus (❶) und mit `BIN` als Binärzahl (❷). Nach dem Hochladen des Sketches wird die Ausgabe im seriellen Monitor angezeigt (siehe Abbildung 6–3).

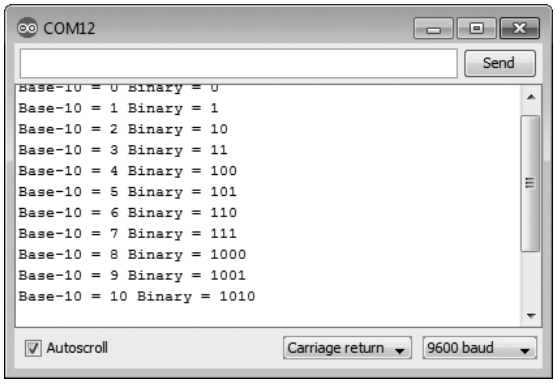

Abb. 6–3 *Ausgabe von Listing 6–2*

Erweitern der digitalen Ausgänge mit Schieberegistern

Die Platine des Arduino verfügt über 13 Digitalpins, die wir als Ausgänge nutzen können, aber manchmal reichen 13 einfach nicht aus. Um weitere Ausgänge hinzuzufügen, können wir ein *Schieberegister* verwenden und haben dabei auf dem Arduino immer noch genug Platz für weitere Ausgänge. Ein Schieberegister ist ein integrierter Schaltkreis (Integrated Circuit, IC) mit acht digitalen Ausgangspins, die durch die Übertragung eines Datenbytes gesteuert werden. In unseren Projekten verwenden wir das Schieberegister 74HC595 aus Abbildung 6–4.

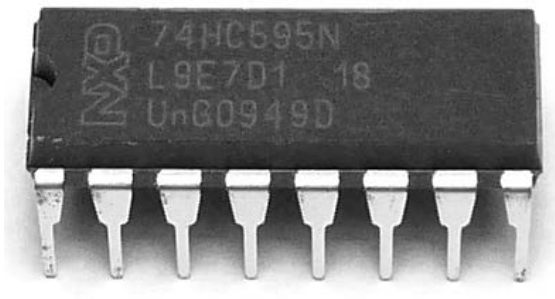

Abb. 6–4 *Der Schieberegister-IC 74HC595*

Das Schieberegister 74HC595 verfügt über acht digitale Ausgänge, die wie die digitalen Ausgangspins des Arduino verwendet werden können. Der IC selbst belegt drei Pins auf dem Arduino, sodass wir unter dem Strich fünf Ausgangspins hinzubekommen.

Das Prinzip hinter einem Schieberegister ist einfach: Wir senden ihm ein Byte Daten, also acht Bits, wobei jedes Bit einem der Ausgangspins entspricht. Von links nach rechts gesehen, stehen die Bits für die Pins in der Reihenfolge von der höchsten zur niedrigsten Nummer. Das Bit ganz links stellt also Ausgangspin 7 des Registers dar, das Bit ganz rechts Pin 0. Wenn wir also zum Beispiel `B10000110` senden, werden die Ausgänge 7, 2 und 1 eingeschaltet und die Ausgänge 0 und 3 bis 6 ausgeschaltet. Das gilt so lange, bis das nächste Datenbyte eintrifft oder der Strom abgeschaltet wird.

Sie können auch mehrere Schieberegister miteinander verbinden, um jeweils weitere acht digitale Ausgangspins zu gewinnen, während am Arduino immer noch lediglich drei Pins belegt sind. Dadurch eignen sich Schieberegister hervorragend, wenn Sie viele LEDs steuern müssen. Das wollen wir nun ausprobieren, indem wir eine Anzeige für Binärzahlen konstruieren.

Projekt Nr. 16: Eine Binärzahlenanzeige aus LEDs bauen

In diesem Projekt verwenden wir acht LEDs, um die Binärzahlen von 0 bis 255 anzuzeigen. In unserem Sketch zählen wir mit einer `for`-Schleife von 0 bis 255 und senden jeden Wert an das Schieberegister, das die binären Entsprechungen der einzelnen Zahlen über LEDs ausgibt.

Die Hardware

Sie brauchen folgende Teile:

- einen Schieberegister-IC 74HC595
- acht LEDs (LED1 bis LED8)
- acht 560-Ω-Widerstände (R1 bis R8)
- eine Steckplatine
- Verbindungsdrähte
- Arduino und USB-Kabel

John Boxall, Arduino-Workshops, dpunkt.verlag, ISBN 978-3-86490-106-5