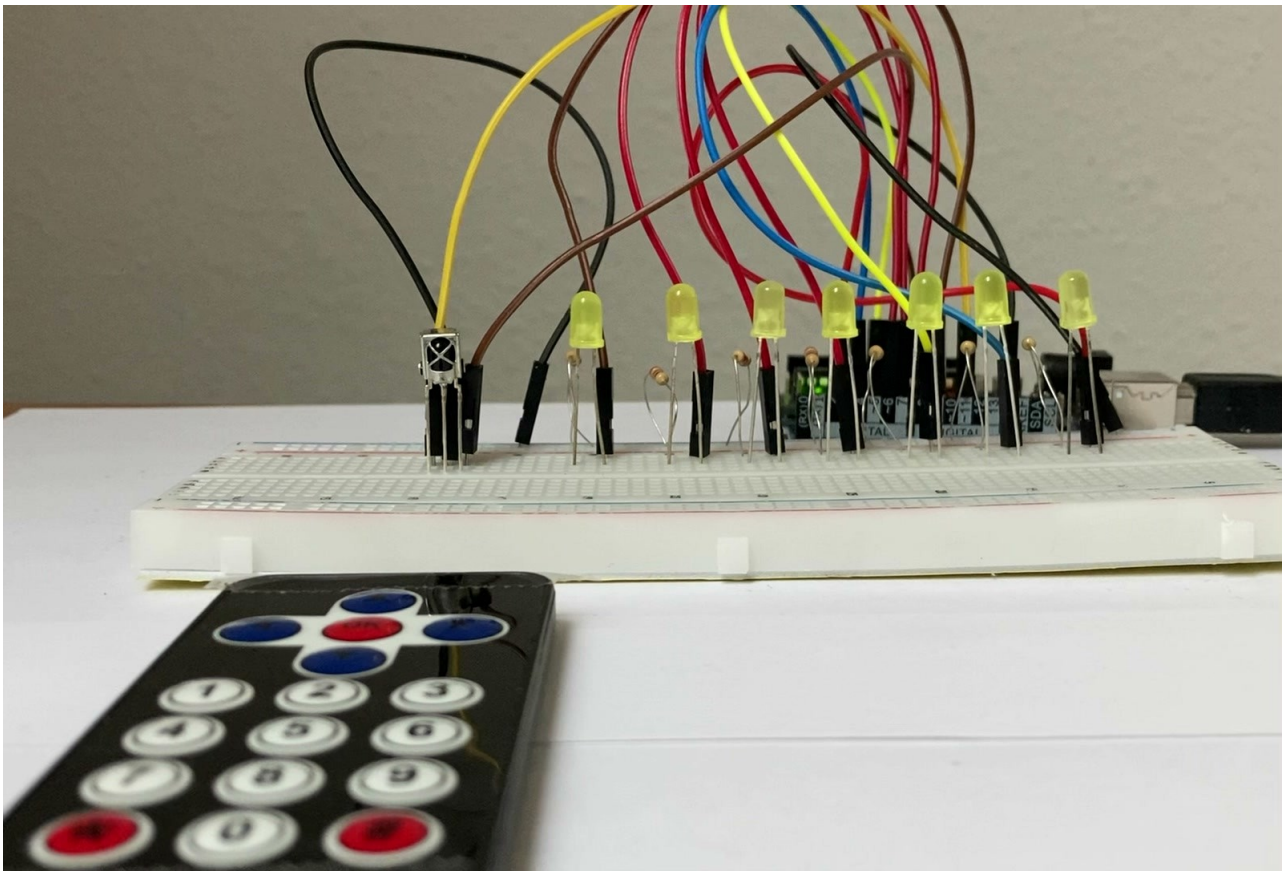
Das Programm kann über die Tasten der Fernbedienung jede LED einzeln ein- und ausschalten, alle gemeinsam schalten und ein Lauflicht starten.

- die Tasten 1 bis 7 schalten jeweils eine LED
- ← Lauflicht nach links
- → Lauflicht nach rechts
- # alle LEDs ausschalten

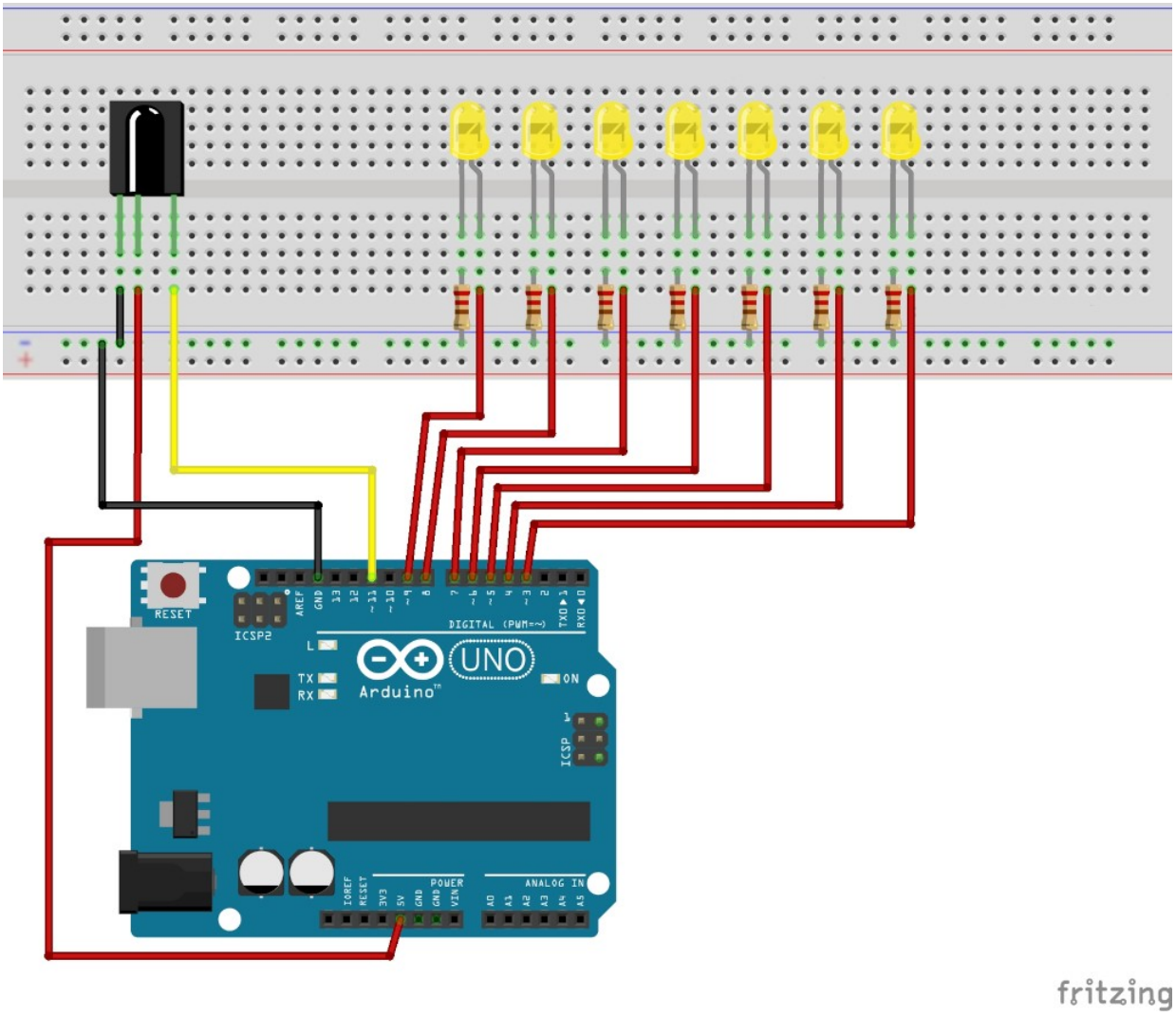
So sieht es aus:



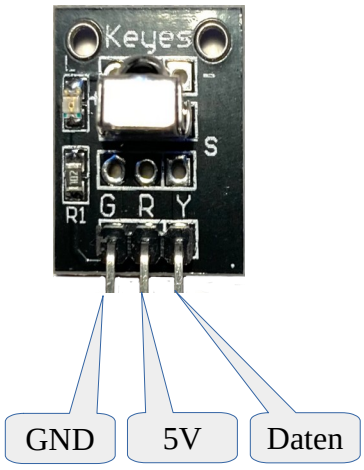
Benötigte Bauteile

- Infrarotempfänger
- Keyes Fernbedienung
- LEDs
- Widerstände 220 Ω
- Leitungsdrähte

Baue die Schaltung auf.



Achte auf die Pinbelegung der Infrarotempfänger.



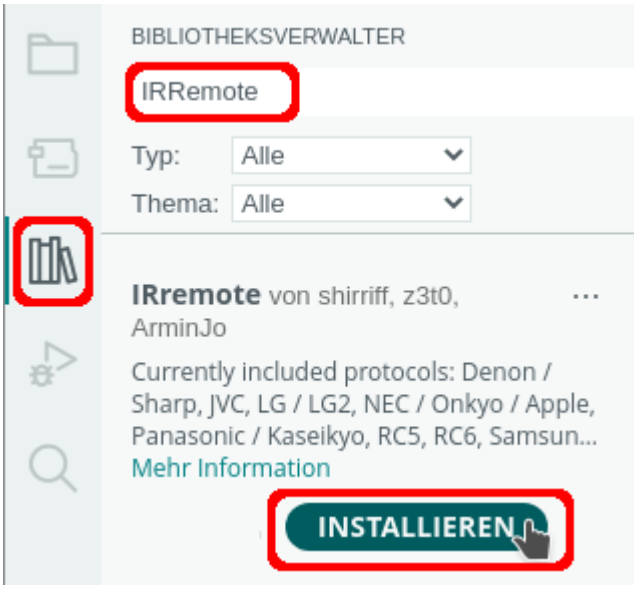
Keyes-Empfänger

VS1838B

Achte darauf, dass die Batterie richtig eingelegt wurde. Der Minus-Pol zeigt nach oben.

Benötigte Bibliothek:

Suche die Bibliothek IRremote ...



… und klicke auf installieren

Die Fernbedienung sendet beim Druck auf die Tasten einen Zahlencode.

Die Tastencodes beziehen sich auf die **Keyes-Fernbedienung**. Die Tastencodes anderer Fernbedienungen kannst du mit Hilfe des Seriellen Monitors herausfinden.

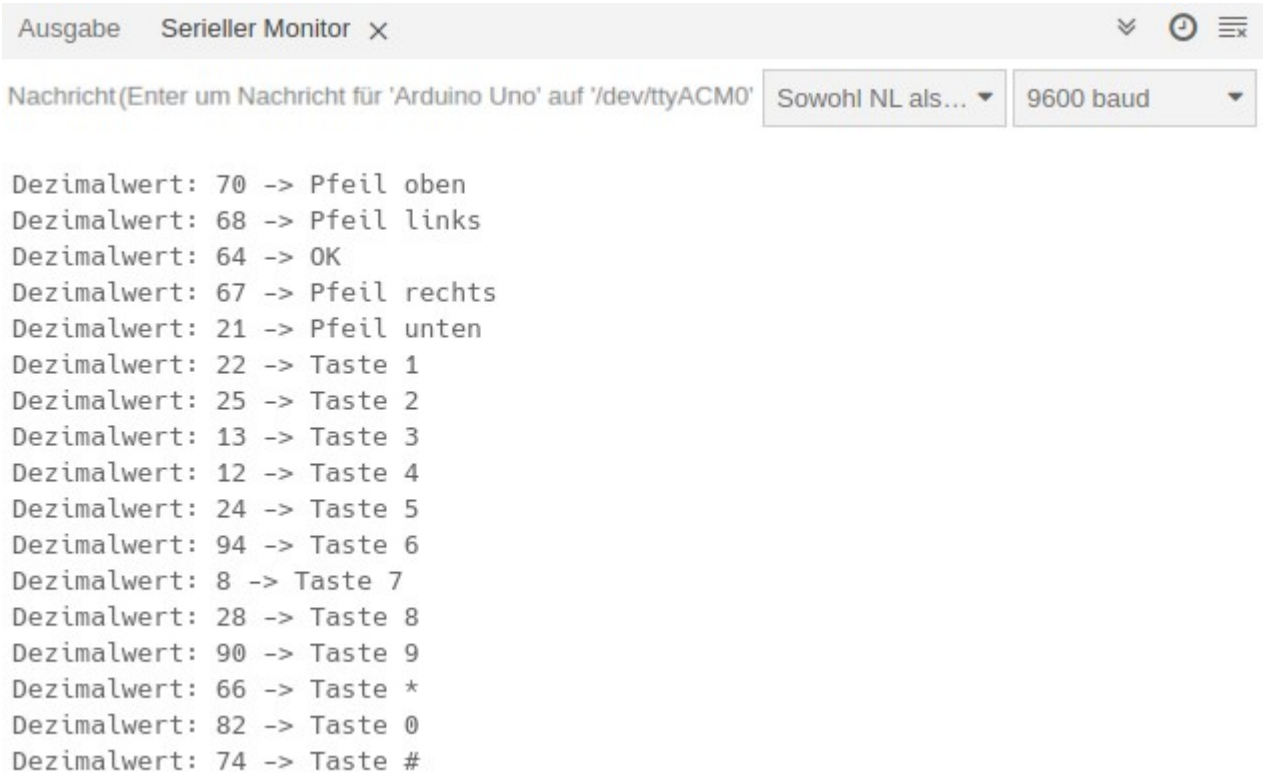
Die Tastencodes der Keyes Fernbedienung (hexadezimal und dezimal).

	Pfeil oben	
	70	
Pfeil links	OK	Pfeil rechts
68	64	67
1	2	3
22	25	13
4	5	6
7	8	9
8	28	90
*	Taste	#
66	82	74

Die Tastencodes kannst du mit folgendem Programm herausfinden. Sie werden im Seriellen Monitor angezeigt.

```
// benötigte Bibliothek einbinden
#include "IRremote.hpp"

// der Pin, an dem der Infrarot-Empfänger angeschlossen ist
int EmpfaengerPin = 11;

void setup()
{
  // Seriellen Monitor starten
  Serial.begin(9600);

  // Infrarot-Empfänger starten
  IrReceiver.begin(EmpfaengerPin);
}

void loop()
{
  // decode() -> Daten lesen
  if (IrReceiver.decode())
  {
    // kurzes delay, damit nur ein Tastendruck gelesen wird
    delay(200);

    // resume -> nächsten Wert lesen
    IrReceiver.resume();
```

```
    /*
      der Empfänger empfängt zwischendurch Signale,
      die nicht ausgewertet werden können
      es sollen deshalb nur die korrekt erkannten Tasten ausgewertet werden
      die Dezimalwerte der korrekten erkannten Tasten liegen zwischen > 0 und < 95
      es wird abgefragt, ob das empfangene Kommando decodedIRData.command
      zwischen 0 und (&&) 95 liegt
    */
    if (IrReceiver.decodedIRData.command > 0 && IrReceiver.decodedIRData.command < 95)
    {
      Serial.print("Dezimalwert: ");

      // IrReceiver.decodedIRData.command = Wert der gedrückten Taste
      Serial.print(IrReceiver.decodedIRData.command);
      Serial.print(" -> ");

      // Werte abfragen und anzeigen
      if (IrReceiver.decodedIRData.command == 22) Serial.println("Taste 1");
      if (IrReceiver.decodedIRData.command == 25) Serial.println("Taste 2");
      if (IrReceiver.decodedIRData.command == 13) Serial.println("Taste 3");
      if (IrReceiver.decodedIRData.command == 12) Serial.println("Taste 4");
      if (IrReceiver.decodedIRData.command == 24) Serial.println("Taste 5");
      if (IrReceiver.decodedIRData.command == 94) Serial.println("Taste 6");
      if (IrReceiver.decodedIRData.command == 8) Serial.println("Taste 7");
      if (IrReceiver.decodedIRData.command == 28) Serial.println("Taste 8");
      if (IrReceiver.decodedIRData.command == 90) Serial.println("Taste 9");
      if (IrReceiver.decodedIRData.command == 82) Serial.println("Taste 0");
      if (IrReceiver.decodedIRData.command == 66) Serial.println("Taste *");
      if (IrReceiver.decodedIRData.command == 74) Serial.println("Taste #");
      if (IrReceiver.decodedIRData.command == 68) Serial.println("Pfeil links");
      if (IrReceiver.decodedIRData.command == 67) Serial.println("Pfeil rechts");
      if (IrReceiver.decodedIRData.command == 70) Serial.println("Pfeil oben");
      if (IrReceiver.decodedIRData.command == 21) Serial.println("Pfeil unten");
      if (IrReceiver.decodedIRData.command == 64) Serial.println("OK");
    }
  }
}
```



Binde die benötigte Bibliothek ein und definiere die Variablen.

```
// benötigte Bibliothek einbinden
#include "IRremote.hpp"

// Pin, an dem der Infrarot-Empfänger angeschlossen ist
int EmpfaengerPin = 11;

// Anzahl der LEDs
#define AnzahlLED 7

// Pins der LEDs
int LED[AnzahlLED] = { 3, 4, 5, 6, 7, 8, 9 };

// beim Start sind alle LEDs ausgeschaltet
bool LEDStatus[AnzahlLED] = { LOW, LOW, LOW, LOW, LOW, LOW, LOW };
```

Im setup-Teil:

```
void setup()
{
  // Infrarot-Empfänger starten
  IrReceiver.begin(EmpfaengerPin);

  // pinMode der LEDs definieren
  for(int i = 0; i < AnzahlLED; i++)
  {
    pinMode(LED[i], OUTPUT);
  }
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  // decode() -> Daten lesen
  if (IrReceiver.decode())
  {
    // kurzes delay, damit nur ein Tastendruck gelesen wird
    delay(200);

    // resume -> nächsten Wert lesen
    IrReceiver.resume();

    /*
      der Empfänger empfängt zwischendurch Signale,
      die nicht ausgewertet werden können
      es sollen dehalb nur die korrekt erkannten Tasten ausgewertet werden
      die Dezimalwerte der korrekten erkannten Tasten liegen zwischen > 0 und < 95
      es wird abgefragt, ob das empfangene Kommando decodedIRData.command
      zwischen 0 und (&&) 95 liegt
    */
    if (IrReceiver.decodedIRData.command > 0 && IrReceiver.decodedIRData.command < 95)
    {
      // Werte abfragen und anzeigen
      // Tasten 1-7
      switch (IrReceiver.decodedIRData.command)
      {
        case 22:
          LEDSchalten(0);
          break;

        case 25:
          LEDSchalten(1);
          break;

        case 13:
          LEDSchalten(2);
          break;

        case 12:
          LEDSchalten(3);
          break;

        case 24:
          LEDSchalten(4);
          break;
        case 94:
          LEDSchalten(5);
          break;
        case 8:
          LEDSchalten(6);
          break;
```

```
        // *
        case 66:
          AlleAn();
          break;

        // #
        case 74:
          AlleAus();
          break;

        // Pfeil links
        case 68:
          LauflichtLinks();
          break;

        // Pfeil rechts
        case 67:
          LauflichtRechts();
          break;

        default:
          break;
      }
    }
  }
}
```

Jetzt fehlen noch die Funktionen LEDSchalten(), LauflichtLinks(), LauflichtRechts() AlleAn() und AlleAus().

```
void LEDSchalten(int LEDNummer)
{
  // LEDStatus umkehren LOW -> HIGH, HIGH -> LOW
  LEDStatus[LEDNummer] = !LEDStatus[LEDNummer];
  digitalWrite(LED[LEDNummer], LEDStatus[LEDNummer]);
}

void LauflichtLinks()
{
  for (int i = 0; i < AnzahlLED; i++)
  {
    digitalWrite(LED[i], HIGH);
    delay(100);
    digitalWrite(LED[i], LOW);
    AlleAus();
  }
}
```

```
void LauflichtRechts()
{
 for (int i = AnzahlLED; i >= 0; i--)
  {
    digitalWrite(LED[i], HIGH);
    delay(100);
    digitalWrite(LED[i], LOW);
    AlleAus();
  }
}

void AlleAus()
{
  // LEDs ausschalten
  // LEDStatus aller LEDs auf LOW setzen
  for (int i = 0; i < AnzahlLED; i++)
  {
    digitalWrite(LED[i], LOW);
    LEDStatus[i] = LOW;
  }
}

void AlleAn()
{
  // LEDs einschalten
  // LEDStatus aller LEDs auf HIGH setzen
  for (int i = 0; i < AnzahlLED; i++)
  {
    digitalWrite(LED[i], HIGH);
    LEDStatus[i] = HIGH;
  }
}
```