```
//
// Sound Activator
//
// steuert TAMS Sound Generator an, wobei über eine Diodenmatrix
// immer zwei Eingänge gleichzeitig auf Null gezogen werden
// mit 6 Schaltern S1 bis S6 werden unterschiedliche Soundgruppen oder Aktionen freigegeben
// S2 erlaubt das Programm um das Brennende Haus
// S6 erlaubt die Blasmusik
// Dr Wolfgang Kreinberg
// 2017-05-08 / update 2018-04-26 / update 2018-05-19
//
// Arduino Nano mit 14 Digitalen Ausgängen und 8 Analogen Ein/Ausgängen
// Belegung
// Pin D13 bis D6: Ausgänge schalten gegen Masse für Eingänge der Diodenmatrix U1 bis U8
// Pin D5 bis D0: Ausgänge schalten gegen Masse auf Eingänge 1 bis 6 des Darlington Arrays
// Pin A0 bis A5: Eingänge zum Abfragen der Schalter S1 bis S6 der Aktionsgruppen
// Pin A6 und A7: Ausgänge schalten gegen Masse auf Eingänge 7 und 8 des Darlington Arrays
// Darlington Array ULN2801A
// Belegung
// Pin 09: Gnd, Gleichspannungsmasse for 5 V DC und 12 V DC
// Pin 10: Referenzspannung Vcc = 12 V DC
// Pin 14 bis Pin 11: Ausgänge schalten gegen Masse auf Eingänge der Relais #1 bis #4
// Pin 18 bis Pin 15: Ausgänge schalten gegen Masse externer Gleichspannung mit Gnd Verbindung zu Pin
09
// Relais 12 V DC, je einmal um
// Belegung
// Alle Spulen auf gemeinsamen Vcc = 12 V DC plus
// Alle Spulen werden von Darlington Pin 14 bis Pin 11 auf Masse gezogen
// Alle 4 Mittenkontakte liegen auf Masse von 12 V AC Lichttrafo
// Alle 4 Arbeitskontakte schalten Verbraucher auf Masse von 12 V AC Lichttrafo

class pinHandler
{
  // Class Member Variables
  // These are initialized at startup
  int ledPin;                                    // the number of the LED pin
  long OnTime;                                   // milliseconds of on-time
  long OffTime;                                  // milliseconds of off-time


  // These maintain the current state
  int ledState;                                  // ledState used to set the LED
  unsigned long previousMillis;                  // will store last time LED was updated

  // Constructor - creates a pinHandler
  // and initializes the member variables and state
  public:
  pinHandler(int pin, long on, long off)
  {
  ledPin = pin;
  pinMode(ledPin, OUTPUT);

  OnTime = on;
  OffTime = off;

  ledState = HIGH;
  previousMillis = 0;
  }                                              // end of constructor pinHandler

  void Update()
  {
    // check to see if it's time to change the state of the LED
    unsigned long currentMillis = millis();

    if((ledState == LOW) && (currentMillis - previousMillis >= OffTime))
    {
      ledState = HIGH;                           // Turn it off, LOW for ON bei Sound
      previousMillis = currentMillis;            // Remember the time
      digitalWrite(ledPin, ledState);            // Update the actual LED
    }
    else if ((ledState == HIGH) && (currentMillis - previousMillis >= OnTime))
```

```
    {
      ledState = LOW;                              // turn it on
      previousMillis = currentMillis;              // Remember the time
      digitalWrite(ledPin, ledState);              // Update the actual LED
    }
  }                                                // on of update in CLASS
};                                                 // end of CLASS pinHandler

// Aktionen 1 - 6 auf Pin 0 bis 5                  // (Schaltdauer, Einschaltzeit)
  int schaltDauer = 300;
  pinHandler led1 (0, 2000, 2000) ;                // Darlington E6 geht auf 3/1
  pinHandler led2 (1, 1000, 1000) ;                // Darlington E5 geht auf 5/5 für Blaskapelle
  pinHandler led3 (2, 5000, 5000) ;                // Darlington E4 Relais 4 für 12 V AC
  pinHandler led4 (3, 10000, 10000) ;              // Darlington E3 Relais 3 für 12 V AC
  pinHandler led5 (4, 3000, 3000) ;                // Darlington E2 Relais 2 für 12 V AC
  pinHandler led6 (5, 1000, 1000) ;                // Darlington E1 Relais 1 für 12 V AC   63743

// Töne 1 - 8 auf pin 6 bis 13 sowie Analog 6 und 7 // (Einschaltzeit, Schaltdauer)

  pinHandler led7 (6, 1854321, schaltDauer) ;      // U8 langes Kirchenläuten
  pinHandler led8 (7,  654321, schaltDauer) ;      // U7 Stundenläuten
  pinHandler led9 (8, 1600013, schaltDauer) ;      // U6 Blaskapelle
  pinHandler led10(9,  420543, schaltDauer) ;      // U5 Kuh
  pinHandler led11(10, 303007, schaltDauer) ;      // U4 Kuckuck
  pinHandler led12(11, 287657, schaltDauer) ;      // U3 Hahn
  pinHandler led13(12, 554321, schaltDauer) ;      // U2 Typhoon
  pinHandler led14(13, 120011, schaltDauer) ;      // U1 Hund
  pinHandler led15(A6, 4000,4000) ;                // Test 1, da noch nicht belegt
  pinHandler led16(A7, 8000,8000) ;                // Test 2, da noch nicht belegt


  int thisPin = 0;

// Analog Pins setzen

  int switchPin1 = 0 ;
  int switchPin2 = 1 ;
  int switchPin3 = 2 ;
  int switchPin4 = 3 ;
  int switchPin5 = 4 ;
  int switchPin6 = 5 ;
  int prellenAbwarten = 150 ;

// Fire and Smoke Variable
//
  unsigned long starte_fire_and_smoke = 2100001;   // Einsatzzeit der Aktion
  int fire_action = 1;                             // Zähler für die Feueraktions, wird gebraucht um den
                                                    // nächsten Startpunkt als Vielfaches der Startzeit
                                                   //zu errechnen
  int nur_Einmal = 1 ;                              // wenn Testschalter S1 geschaltet ist, wird der
                                                   //Prozess nur einmal gestartet

// BlasMusik Variable
//
  unsigned long starte_BlasMusik = 1600013;         // Einsatzzeit der Blasmusik
  int Licht_pin = 1;                                // LED2 auf Klemme 5/5
  int Blas_Musik_Pin = 8;                           // Arduino Pin 8 für Blasmusik Sound
  int blas_action = 1;                              // Zähler für die BlasMusik-aktions, wird gebraucht
                                                   // um den nächsten Startpunkt als Vielfaches der
                                                   // Startzeit zu errechnen
  int blas_nur_Einmal = 1 ;                         // wenn Testschalter S1 geschaltet ist, wird der
                                                    // Prozess nur einmal gestartet


void setup()
{
  for (thisPin = 0; thisPin < 6; thisPin++)        // 6 Pins an Darlingtons
  {
    digitalWrite (thisPin, LOW) ;                  // Darlingtons schalten bei HIGH durch
  }
  for (thisPin = 6; thisPin < 14; thisPin++)       // 8 Pins and Diodenmatrix schalten bei LOW durch
```

```
  {
    digitalWrite (thisPin, HIGH) ;                  // Nicht ändern da sonst Sound Blockiert
  }
  // Schaltereingänge setzen
  pinMode (A0, INPUT) ;                             // S1: Analog Eingang A0
  digitalWrite (A0, HIGH);                          // interner PullUp Widerstand ein
  pinMode (A1, INPUT) ;                             // S2: Analog Eingang A1
  digitalWrite (A1, HIGH);                          // interner PullUp Widerstand ein
  pinMode (A2, INPUT) ;                             // S3: Analog Eingang A2
  digitalWrite (A2, HIGH);                          // interner PullUp Widerstand ein
  pinMode (A3, INPUT) ;                             // S4: Analog Eingang A3
  digitalWrite (A3, HIGH);                          // interner PullUp Widerstand ein
  pinMode (A4, INPUT) ;                             // S5: Analog Eingang A4
  digitalWrite (A4, HIGH);                          // interner PullUp Widerstand ein
  pinMode (A5, INPUT) ;                             // S6: Analog Eingang A5
  digitalWrite (A5, HIGH);                          // interner PullUp Widerstand ein

  // Analoge auf Ausgang setzen
  pinMode (A6, OUTPUT) ;                            // Analog Ausgang geht auf Darlington E7
  digitalWrite (A6, LOW);                           // Ausgang auf 0 V schalten
  pinMode (A7, OUTPUT) ;                            // Analog Ausgang geht auf Darlington E8
  digitalWrite (A7, LOW);                           // Ausgang auf 0 V schalten
}


void loop()
{
// switchPin1 = wenn auch gedrückt, geht Action sofort los
//
// Testzwecke Anfang
  led1.Update();                                    // Darlington E6: Licht Bergwerk 12 V DC       //
noch frei Test 2000
  led15.Update();     // Analog Ausgang geht auf Darlington E7 // noch frei Test 4000
  led16.Update();     // Analog Ausgang geht auf Darlington E8 // noch frei Test 8000
// Testzwecke Ende

// Betrieb Anfang
  if(readSwitch(switchPin4)) led3.Update();         // Relais 4: 12 V AC                           //
noch frei
  if(readSwitch(switchPin4)) led4.Update();         // Relais 3: 12 V AC                           //
noch frei für Brauerei nehmen
  if(readSwitch(switchPin5)) led7.Update();         // U8 langes Läuten
  if(readSwitch(switchPin5)) led8.Update();         // U7 Glockenschlag
  if(readSwitch(switchPin6)) BlasMusik();           // switchPin6 und U6 Blasmusik
  if(readSwitch(switchPin3)) led10.Update();        // U5 Kuh
  if(readSwitch(switchPin3)) led11.Update();        // U4 Kuckuck
  if(readSwitch(switchPin3)) led12.Update();        // U3 Hahn
  if(readSwitch(switchPin4)) led13.Update();        // U2 Typhoon
  if(readSwitch(switchPin3)) led14.Update();        // U1 Hund
  if(readSwitch(switchPin2)) FireSmoke();           // switchPin2 ist für Brennendes Haus
}

int readSwitch(int derSwitch) {
  if (analogRead(derSwitch) < 500)  {               // Analogeingang abfragen
    delay (prellenAbwarten);
    if (analogRead(derSwitch) < 500)                // nochmal fragen
       return 1 ;                                   // war geschaltet
  }
  return 0 ;                                        // war nicht geschaltet
  }

void FireSmoke()
  {
    // durch die Verwendung von Delay laufen in diesem Segment nur diese Aktionen ab
    // Es wird Status U1 (Pin D13), U7(Pin D7) und U8 (Pin D6) genutzt
    // diese Pins müssen definiert ausgeschaltet werden um Doppelsteuerun zuzulassen
    //
    if (readSwitch(switchPin1) && (nur_Einmal > 0))             // Pin 1 ist auch noch gedrückt UND
nur_Einmal ist noch „1"
    {
     if (millis() > 10000) {                                    // warte 10 Sekunden
```

```
        FireSmokeAction();                                          // starte die Aktion
        nur_Einmal = 0;                                             // aber nur dieses Mal, daher auf Null
setzen
     }
    }
    else if (starte_BlasMusik*fire_action > millis()) // ist Startzeit erreicht?? Vielfaches der Start-
zeit für erneuten Start
    {
//  Mache nichts, wenn die Zeit noch nicht erreicht ist
    }
    else
    {
     FireSmokeAction();                             // regulärer Start gemäß Startzeit wenn millis()
Startzeit überschreitet
    }
    return fire_action;
  }

void FireSmokeAction()
    {
    // lasse Programm ablaufen
    //  Setze Pins
      digitalWrite(13, HIGH);                       // Pin für Sirene und Martinshorn
      digitalWrite( 6, HIGH);                       // Pin für Sirene und Martinshorn
      digitalWrite( 7, HIGH);                       // Pin für Sirene und Martinshorn
    // Schalte Feuer ein
      digitalWrite( 5, HIGH);                       // relais #1 über Darlington schaltet Feuer ein
      delay (5000) ;                                // warte 5 Sekunden
    // Schalte Rauch ein
      digitalWrite( 4, HIGH);                       // relais #2 über Darlington schaltet Rauch ein
      delay (15000) ;                               // warte 15 Sekunden
    // Starte Sirene
      digitalWrite(13, LOW);
      digitalWrite(7,  LOW);
      delay (schaltDauer) ;
      digitalWrite(13, HIGH);
      digitalWrite(7,  HIGH);
      delay (30000);
    // Starte Martinshorn
      digitalWrite(13, LOW);
      digitalWrite(6,  LOW);
      delay (schaltDauer) ;
      digitalWrite(13, HIGH);
      digitalWrite(6,  HIGH);
      delay (120000);
    // Schalte Feuer aus
      digitalWrite( 5, LOW);                         // relais #1 schaltet Feuer aus
      delay (30000);
    // Schalte Rauch aus
      digitalWrite( 4, LOW);                         // relais #2 schaltet Rauch aus
          digitalWrite( 12, LOW);        // U2 Typhoon
          delay (500);
          digitalWrite( 12, HIGH);        // U2 Typhoon
    fire_action = fire_action +1;                    // Bereite nächste Action vor
    return fire_action ;
    }

void BlasMusik()
  {
    // durch die Verwendung von Delay laufen in diesem Segment nur diese Aktionen ab
    // Es wird Status U6(Pin D8) genutzt
    // diese Pins müssen definiert ausgeschaltet werden um Doppelsteuerun zuzulassen
    //
    if (readSwitch(switchPin1) && (blas_nur_Einmal > 0))              // Pin 1 ist auch noch gedrückt
UND nur_Einmal ist noch „1"
    {
     if (millis() > 10000) {                                          // warte 10 Sekunden
        BlasMusikAction();                                            // starte die Aktion
        blas_nur_Einmal = 0;                                          // aber nur dieses Mal, daher auf
Null setzen
     }
```

```
    }
    else if (starte_BlasMusik*blas_action > millis()) // ist Startzeit erreicht?? Vielfaches der Start-
zeit für erneuten Start
    {
//  Mache nichts, wenn die Zeit noch nicht erreicht ist
    }
    else
    {
     BlasMusikAction();                          // regulärer Start gemäß Startzeit wenn millis()
Startzeit überschreitet
    }
    return blas_action;
  }

void BlasMusikAction()
    {
    // lasse Programm ablaufen
    //  Setze Pins
      digitalWrite( Blas_Musik_Pin, HIGH);          // Pin für Blasmusik
    // Schalte Licht ein und Lautsprecher um        // LED2
      digitalWrite( Licht_pin, HIGH);               // Pin über Darlington schaltet Licht ein
      delay (5000) ;                                // warte 5 Sekunden
    // Starte Blasmusik
      digitalWrite( Blas_Musik_Pin, LOW);
      delay (schaltDauer) ;
      digitalWrite( Blas_Musik_Pin, HIGH);
      delay (22000);
    // Schalte Licht aus und Lautsprecher um
      digitalWrite( Licht_pin, LOW);                // Darlington schaltet Licht aus
          digitalWrite( 12, LOW);        // U2 Typhoon
          delay (500);
          digitalWrite( 12, HIGH);        // U2 Typhoon
    blas_action = blas_action +1;                    // Bereite nächste Action vor
    return blas_action ;
    }
```