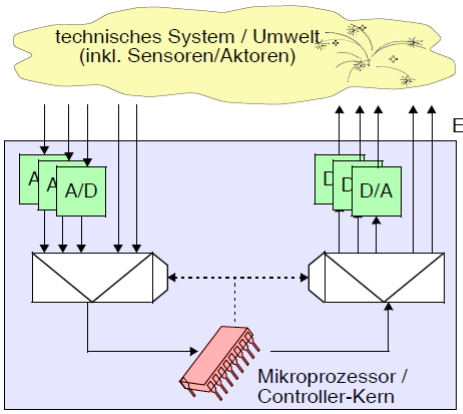
Programmieren in C

für Elektrotechniker

Einschub: Programmierung von Controllern

- **Arduino-Boards**

Steuerung technischer Systeme



MSR Wirkkreislauf
(messen, steuern, regeln)

Controller

Arduino

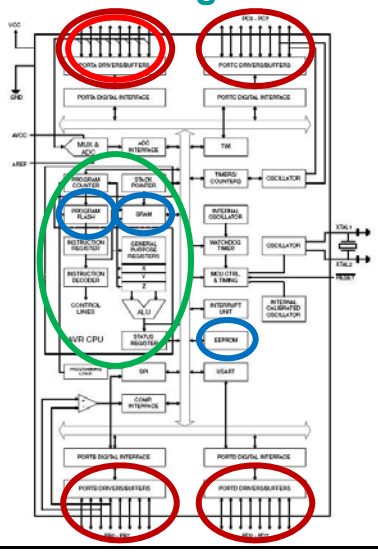
- **Quelloffene Embedded-Plattform (HW+SW)**
- **Hardware**
  - E/A-Board mit einfachem Microcontroller aus megaAVR-Serie (8 Bit, s.u.). Varianten: z.B. ARM Cortex-M3 (32 Bit, vgl. Raspberry-Plattform)
  - Programmierung über serielle Schnittstelle (heute: USB)

Arduino UNO R3:
hier: ATmega328 Microcontroller in SMD-Bauweise

Arduino

- **Quelloffene Embedded-Plattform (HW+SW)**
- **Software**
  - C/C++-ähnliche Programmiersprache
    → technische Details in Bibliotheken versteckt (→ header-Dateien)
  - Integrierte Entwicklungsumgebung IDE
    → in Java implementiert
    → Editor, gcc-Compiler, Terminal
    (s.u., vgl. dev-cpp-Umgebung)
  - Zwei zentrale Funktionen (statt **`main()`**)
    → **`setup()`**: einmal bei Projektstart
    → **`loop()`**: solange Board eingeschaltet

Atmel ATmega32 Microcontroller

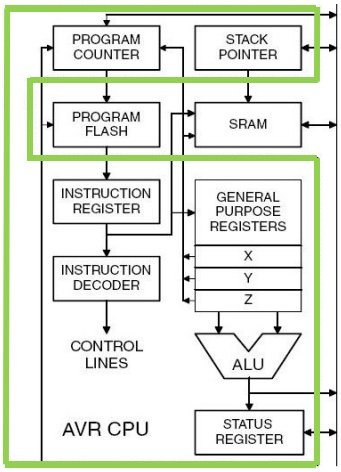


Prozessorarchitektur

- 8 Bit CPU
- 32 kByte Flash Programmspeicher
- 2 kByte RAM Datenspeicher
- 1 kByte EEPROM Datenspeicher
- 16 MHz
- 32 I/O-Pins
  - 8 Analog-Eingänge
- 4 PWM
- 3 ext. Interrupts

- DIL-40-Gehäuse

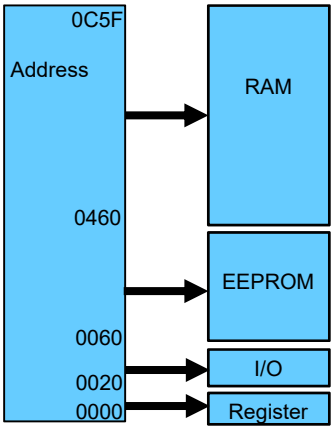
Atmel ATmega32 Microcontroller



CPU

- 32 kByte Programmspeicher
- Programmzähler und Befehlsregister: s. v. Neumann-Rechner
- 32 8-Bit-Register

Atmel ATmega32 Microcontroller



Adressräume

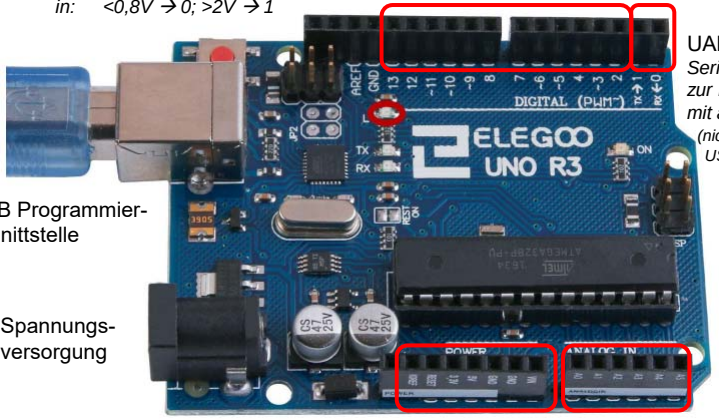
Speichermanagement

- Unterteilung des Adressraums in
  - 0x0000 .. 0x001F: 32 Byte
    Registersatz
    32 Universalregister,
    R26+27: Pointerregister X
    R28+29: Pointerregister Y
    R30+31: Pointerregister Z
  - 0x0020 .. 0x005F:
    64 Byte I/O
  - 0x0060 .. 0x045F:
    1.024 Byte EEPROM
  - 0x0460 .. 0x0C5F:
    2.048 Byte RAM

ELEGOO Arduino-Board

out: 0 → 0V; 1 → 5V
in: <0,8V → 0; >2V → 1

Digitalports ~: auch Analogausgänge (PWM)



UART
Serielle Schnittstelle zur Kommunikation mit anderen Geräten (nicht nutzbar, wenn USB verwendet wird)

USB Programmier-schnittstelle

Spannungs-versorgung

Versorgungsanschlüsse analoge Eingänge
10 Bit (1.024 Stufen)

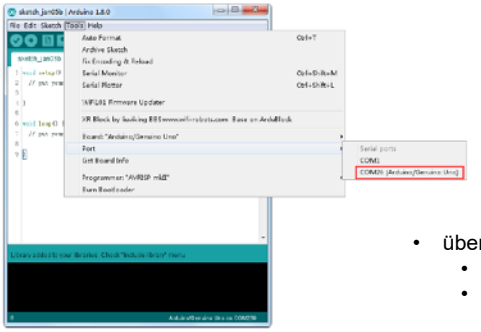
Arduino-IDE (Entwicklungsumgebung)

- Sketch: Arduino-Programm
- Codestruktur:

```
void setup () {
  …
}
void loop () {
  …
}
```

setup(): einmalig zu Programmstart aufgerufene Initialisierungs-funktion
loop(): ausgeführte Endlosschleife

Terminal (Kommandozeileninterpreter, serieller Monitor)



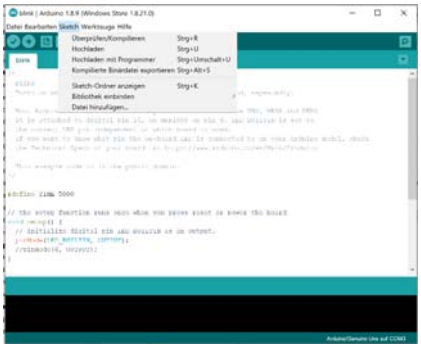
- über serielle Schnittstelle angeschlossen
  - Port einstellen
  - Übertragungsrate festlegen (typ. 9600 Baud)
- Terminal wie unter dev-cpp

Beispiel „blink“

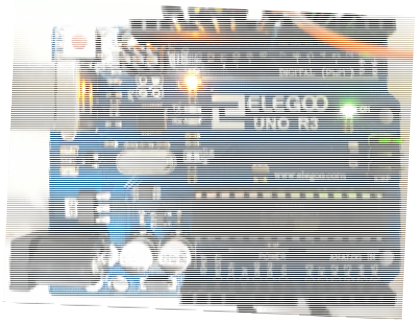
blink.ino: On-board-LED blinkt.

```
// Wartezeit: 5.000 ms
#define TIME 5000
void setup() {
   // initialize digital pin LED_BUILTIN (pin 6) as an output.
   pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
   // schreibe Digitalwert “1” an Pin 6 (LED)
   // schalte eingebaute LED ein
   digitalWrite(LED_BUILTIN, HIGH);
   // warte definierte Zeit
   delay(TIME);
   // schalte eingebaute LED aus
   digitalWrite(LED_BUILTIN, LOW);
   delay(TIME);
}
```


Beispiel „blink“



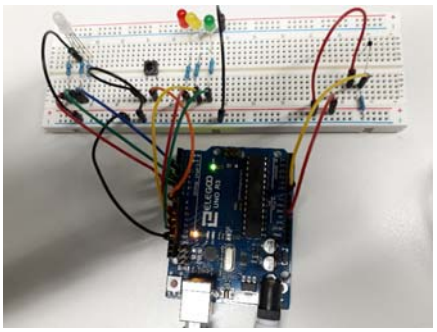
- Sketch übersetzen (Kompilieren)
- Sketch auf board hochladen



- On-Board-LED (gelb) blinkt

Drei ELEGOO-Beispiele

- **Ampel**
  - Ein- und Ausgabe
- **RGB-LED**
  - Analogausgabe (PWM)
- **Thermometer**
  - Analogeingabe



Beispiel „Ampel“ (→ Digital-E/A)

```
#define RED 10       // rote LED an Port 10
#define YELLOW 11    // gelbe LED an Port 11
#define GREEN 12     // grüne LED an Port 12

#define BUTTON 8     // Taster an Port 8

#define DELAY_TIME_LONG 2000
#define DELAY_TIME_SHORT 500

void setup() {
  pinMode(RED, OUTPUT);       // Ports 10 .. 12: Ausgabe
  pinMode(GREEN, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
        // Port 8: Eingabe, spannungslos: high (pullup)
  digitalWrite(RED, HIGH);    // rote LED leuchtet
  digitalWrite(GREEN, LOW);   // grüne LED aus
  digitalWrite(YELLOW, LOW);  // gelbe LED aus
}
```

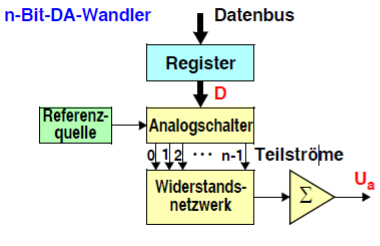
Beispiel „Ampel“ (→ Digital-E/A)

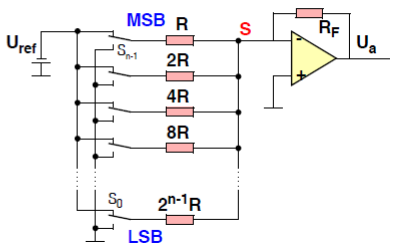
```
void loop() {

  if (digitalRead(BUTTON) == LOW) {
    // Taster gedrückt: reset, d.h. Ampel auf rot
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, LOW);
    digitalWrite(YELLOW, LOW);
  } else {  // Ampelzyklus: rot – rotgelb – grün - gelb
    digitalWrite(RED, HIGH);  // Ampel rot
    digitalWrite(GREEN, LOW);
    digitalWrite(YELLOW, LOW);
    delay(DELAY_TIME_LONG);   // lange Zeit warten
    ...
    digitalWrite(RED, LOW);   // Ampel gelb
    digitalWrite(GREEN, LOW);
    digitalWrite(YELLOW, HIGH);
    delay(DELAY_TIME_SHORT);  // kurze Zeit warten
  }

}
```


Beispiel RGB-LED (→ Analog-Ausgang)

- Ausgabe von Analogwerten zur Helligkeitssteuerung von LEDs
  - D/A-Wandlung

Parallele Umsetzung

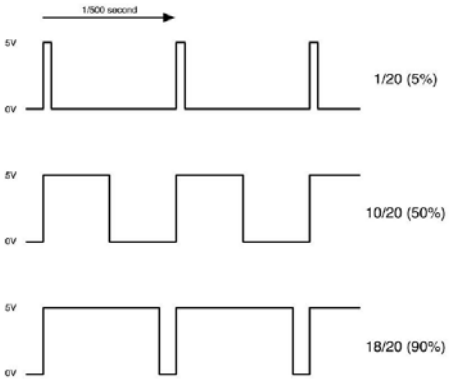




- ➢ schnell
- ➢ einfaches Verfahren
- ➢ abhängig von Genauigkeit der Widerstände
- ➢ selten verwendet

Beispiel RGB-LED (→ Analog-Ausgang)

- Ausgabe von Analogwerten zur Helligkeitssteuerung von LEDs
  - D/A-Wandlung durch Pulsweitenmodulation (PWM)



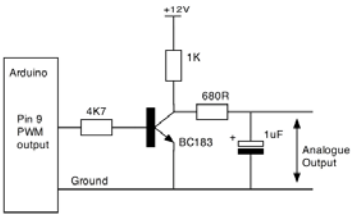
`analogWrite(n)`, 0 ≤ n ≤ 255

- Ausgabe eines Pulses von 1/500 s mit
  - ➢ n/255: Anteil high
  - ➢ 1 – n/255: Anteil low
- Wechsel für Auge zu schnell
  - ➢ Intensität n/255

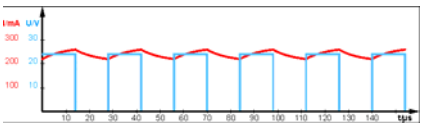
Programmieren in C Programmieren von Controllern

Beispiel RGB-LED (→ Analog-Ausgang)

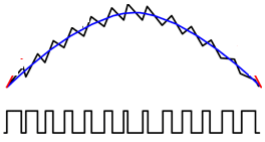
- Ausgabe von Analogwerten zur Helligkeitssteuerung von LEDs
  - D/A-Wandlung durch Pulsweitenmodulation (PWM)
    - ➢ Glättung des Ausgangs mittels RC-Glied



Kondensator glättet Digitalsignal



Ergebnis: fast Gleichspannung



Ergebnis: angenähertes Analogsignal

Beispiel RGB-LED (→ Analog-Ausgang)

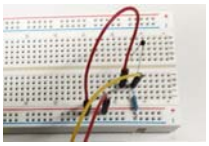
```
#define BLUE 3     // blaue LED an PWM-Port 3
#define GREEN 5    // grüne LED an PWM-Port 5
#define RED 6      // rote LED an PWM-Port 6

void setup() {
  pinMode(RED, OUTPUT);       // alle Ports: Ausgabe
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
  digitalWrite(RED, HIGH);    // RGB-Wert: rot
  digitalWrite(GREEN, LOW);
  digitalWrite(BLUE, LOW);
}

// Definition von drei Variablen für die drei Farben
int redValue;
int greenValue;
int blueValue;
```


Beispiel RGB-LED (→ Analog-Ausgang)

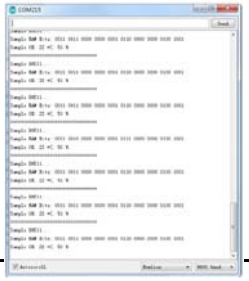
```
void loop() {
#define delayTime 20 // Geschwindigkeit des Farbübergangs

  redValue = 255; // Farbwerte zwischen 0 und 255
  greenValue = 0;
  blueValue = 0;

  for (int i = 0; i < 255; i++) {   // Übergang von Rot nach Grün
    redValue--;                     // rot-Wert verkleinern
    greenValue++;                   // grün-Wert vergrößern
    analogWrite(RED, redValue);     // neue Werte ausgeben
    analogWrite(GREEN, greenValue);
    delay(delayTime);               // kurze Zeit warten
  }

  // Übergang Grün nach Blau analog
  // Übergang von Blau nach Rot analog

}
```

Beispiel Thermometer (→ Analog-Eingang)

- Einlesen des Widerstandwertes eines Thermowiderstands
- Umwandeln des Widerstandwerts in Temperaturwerte
- Ausgabe der Temperatur auf das Terminal (serieller Monitor)



Thermischer Widerstand (NTC)
- ➢ Widerstandswert stärker von Temperatur abhängig als bei normalen Widerständen

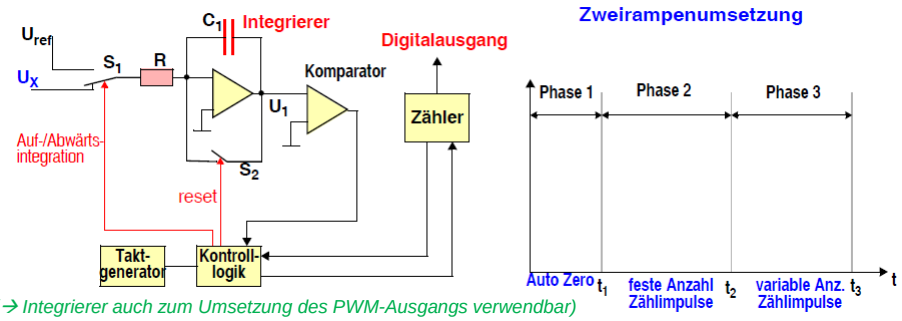


Ausgabe auf dem Terminal

Beispiel Thermometer (→ Analog-Eingang)

- Einlesen des Widerstandwertes eines Thermowiderstands
- Umwandeln des Widerstandwerts in Temperaturwerte
- Ausgabe der Temperatur auf das Terminal (serieller Monitor)

A/D-Wandlung: mehrere Verfahren möglich (hier: Dual-Slope-Verfahren)



(→ Integrierer auch zum Umsetzung des PWM-Ausgangs verwendbar)

Beispiel Thermometer (→ Analog-Eingang)

```
int tempPin = 0;

void setup() {
  Serial.begin(9600);  // Aufbau der Verbindung mit Baudrate
}

void loop() {
  double tempK; // Temperatur in Kelvin
  float tempC;  // Temperatur in Grad Celsius
  float tempF;  // Temperatur in Grad Fahrenheit

  // analogen NTC-Wert einlesen; A/D-Wandlung durch Hardware
  int tempReading = analogRead(tempPin);

  // in Temperaturwert (Kelvin)umrechnen
  tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
  tempK = 1 / (0.001129148 + (0.000234125 +
              (0.0000000876741 * tempK * tempK )) * tempK );
  tempC = tempK - 273.15;            // Kelvin -> Celcius
  tempF = (tempC * 9.0)/ 5.0 + 32.0; // Celcius -> Fahrenheit
```


Beispiel Thermometer (→ Analog-Eingang)

```
  // Temperaturwert über serielle Schnittstelle ausgeben
  Serial.print("Temperature:\n");  // auch serial.println()
  Serial.print("  ");
  Serial.print(tempK, 1);
  Serial.print(" Kelvin\n");
  Serial.print("  ");
  Serial.print(tempC, 1);
  Serial.print(" °C\n");
  Serial.print("  ");
  Serial.print(tempF, 1);
  Serial.print(" °F\n");
  Serial.print("========================================\n");

  delay(1000);  // 1 sec bis zur nächsten Messung warten
}
```