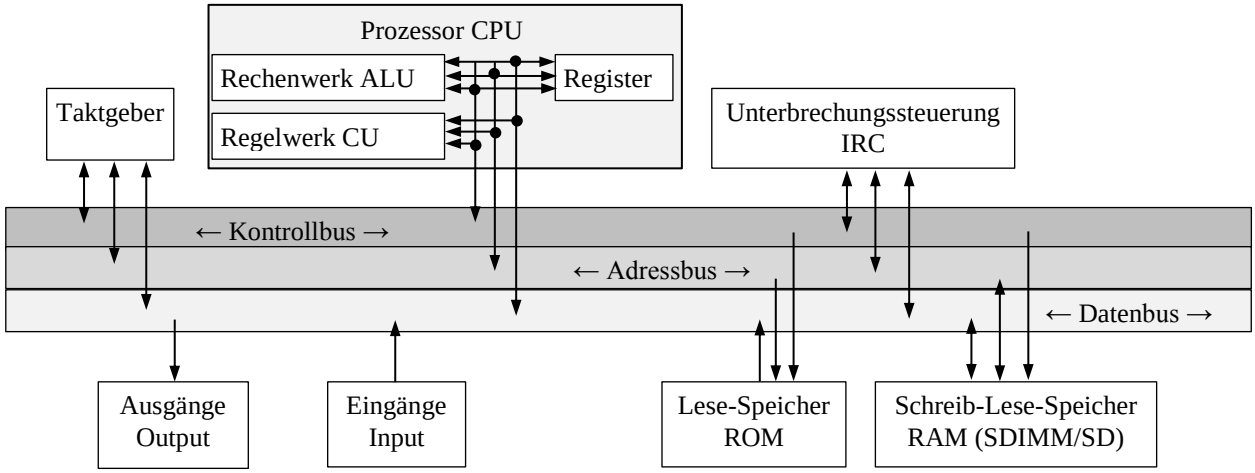
4.0.1. Mikrocontroller

4.0.1.1. Aufbau

Ein Mikrocontroller ist ein vollständiger kleiner Computer, dem nur das Netzteil und die Ein- und Ausgabegeräte fehlen. Trotz seines sehr unauffälligen Äußeren (siehe rechts) und seines bescheidenen Preises enthält er z.T. recht leistungsfähige Komponenten.

Das Gehirn des Mikrocontrollers ist der **Prozessor** (central processing unit CPU). Er ist für die Verarbeitung und Ausführung der Befehle und Programme zuständig.
Er enthält das **Rechenwerk** (arithmetic logic unit ALU), welches arithmetische Operationen wie Addition, Subtraktion, usw. sowie logische Entscheidungen wie die Überprüfung von UND, ODER, NICHT, usw. ausführt.
Den Zu-und Abfluss sowie die Übersetzung von Befehlen, Daten und Adressen kontrolliert das **Regelwerk** (control unit CU).
Das Kurzzeitgedächtnis des Prozessors ist das **Register**, in dem Befehle, Ergebnisse (Daten) und Speicherorte (Adressen) zwischengespeichert werden.



Befehle, Daten und Adressen werden über das **Bussystem** ausgetauscht, dessen Fahrplan vom **Taktgeber** geregelt wird. Neben der **Zugriffsgeschwindigkeit** der Speicher ist die Kapazität dieses Bussystems ein wichtiger Faktor für die Leistungsfähigkeit des Gesamtsystems. Ältere 32 bit-Computer können im **Datenbus** 16 bit transportieren, und im **Adress- bzw. Kontrollbus** jeweils 8 bit. Moderne Computer mit **Mehrkernprozessoren** müssen zwei oder vier Rechenwerke gleichzeitig versorgen und haben einen 64 bit-Bus. Ein **bit** ist die kleinste Speichereinheit eines Computers, der die Zustände 1 (An) oder 0 (Aus) annehmen kann. Ein bit kann also eine Stelle einer Binärzahl aufnehmen. Ein **byte** sind 8 bit und können eine 8-stellige Binärzahl aufnehmen. Ein byte kann also die Zahlen $(0000\ 0000)_2 = (00)_{16} = 0$ bis $(1111\ 1111)_2 = (FF)_{16} = 255$ speichern.

Eingänge (Inputs) werden von entsprechenden Geräten wie Maus oder Tastatur aber auch Sensoren wie z.B. Mikrofon, Kamera, Thermometer, Lagesensor, usw. belegt.
Ausgänge (Outputs) gehen an Bildschirm, Drucker, Lautsprecher, aber auch Motoren und Schalter.

Eingangssignale werden von der **Unterbrechungssteuerung** (interrupt control) nach Dringlichkeit sortiert und an den Prozessor gesendet mit der Anweisung, das laufende Programm zu unterbrechen und z.B. den Mausklick des Benutzers vorrangig zu behandeln.

Im menschlichen Gehirn sind die Kapazität und die Verfügbarkeit des **Gedächtnisses** mindestens ebenso wichtig wie die eigentliche **Denkgeschwindigkeit** für die geistige Leistungsfähigkeit. Entsprechend ist für einen Rechner nicht nur die Geschwindigkeit bzw. Taktfrequenz der CPU von Bedeutung sondern die **Zugriffsgeschwindigkeit** und **Kapazität** der **Speicher**. **Je größer aber der Speicher ist, desto länger braucht man, um etwas in ihm zu finden**. Man muss also Kompromisse machen zwischen Kapazität und Zugriffsgeschwindigkeit. In der Praxis werden die benötigten Daten schrittweise aus großen langsamen und entfernt liegenden Speichern heraussortiert und in immer kleinere, schnellere und näher liegende **Zwischenspeicher** (Caches) an die CPU herangeführt.

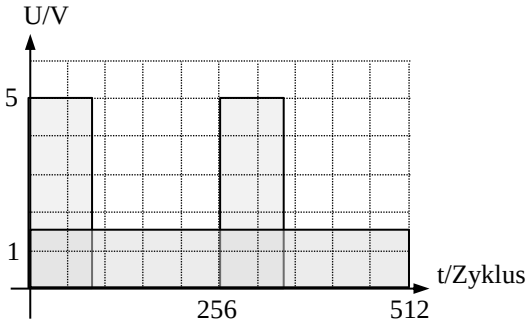
Ein gestartetes Programm wie z.B. eine Textverarbeitung wird zunächst von ihrem Dauerablageplatz auf der **Festplatte** (solid disk sd) in einen kleineren, leichter zugänglichen **Arbeitsspeicher** kopiert. Dieser befindet sich meistens auf erweiterbaren Speicherkarten wie z.B. Dual/Single Input Memory Module (SIMM/DIMM) mit meist mehreren teilweise durch einen eigenem Taktgeber synchronisierten **Schreib-Lese-Speichern** (Synchronous Dynamic Random Access Memory SDRAM), welche oft einen bedeutenden Anteil am Gesamtpreis des Computers haben. Die für die Bearbeitung des nächsten Mausklicks benötigten Daten werden wiederum in das noch kleinere und noch schnellere **Rechenregister** im Prozessor selbst kopiert.

Alle Zwischenspeicher müssen spätestens im Sekundentakt ständig aktualisiert werden und werden beim Abschalten des Computers gelöscht. Damit der Computer beim Anschalten weiß, wie er an die benötigten Daten kommt und wo die Festplatte ist, gibt es einen kleinen **Lesespeicher** (Read Only Memory ROM), dessen Inhalt dauerhaft und nicht überschreibbar ist. Er enthält das **Basic Input Output System** (BIOS), welches beim Anschalten automatisch in den Arbeitsspeicher geladen wird und dem Prozessor die Orientierung zwischen Bussystem, Speichern, Ein- und Ausgängen ermöglicht. Heute sind diese Speicher mit speziellen Programmen auch überschreibbar (Electrically Erasable Programmable ROM EEPROM oder FLASH-EEPROM). Nicht ganz ungefährlich aber üblich ist die z.B. **automatische Aktualisierung** des BIOS von **Mobiltelefonen** über das Internet.

Übungen: Aufgaben zu Mikrokontrollern Nr. 1 - 5

4.0.1.2. Analoge PWM-Ausgänge

Ein **digitales** Gerät wie ein Mikroprozessor kann keine **analogen**, d.h. stufenlos regelbaren Signale produzieren, da es nur die Zustände Ein (1) und Aus (0) kennt. Um analoge Geräte wie **Lautsprecher** und **Motoren** steuern zu können, können **digitale Ausgänge** durch **Pulsweitenmodulation** (PWM) als **analoge Ausgänge** genutzt werden.



Wenn z.B. ein PWM-Ausgang mit einer **Auflösung** von 8 bit bei 5 Volt eine Spannung von 1,73 V liefern soll, so wird er von jeweils insgesamt $2^8 = 256$ Zyklen für $\frac{1{,}73\ \text{V}}{5\ \text{V}} \cdot 256 \approx 89$ Zyklen angeschaltet und für 256 – 89 = 167 Zyklen ausgeschaltet.

Die mittlere Spannung ist dann $\frac{89 \cdot 5\ \text{V} + 167 \cdot 0\ \text{V}}{256} \approx 1{,}73\ \text{V}$.

Übungen: Aufgaben zu Mikrokontrollern Nr. 6

4.0.1.3. Analoge Eingänge

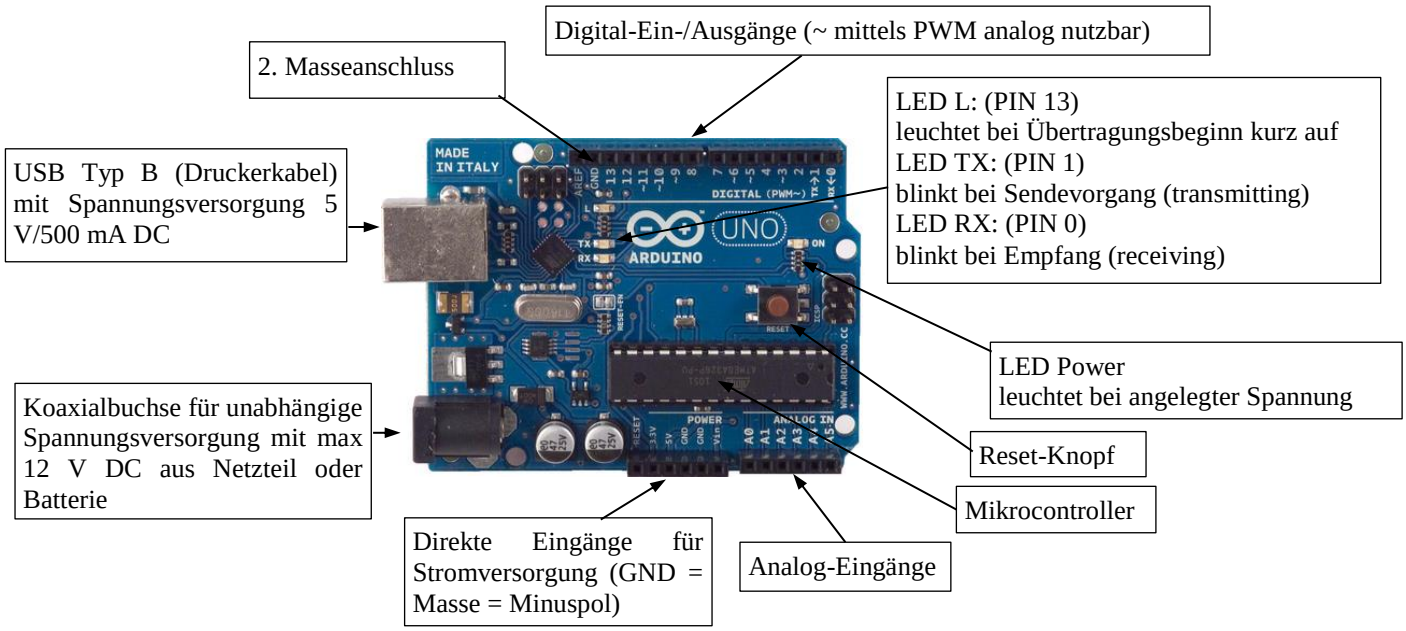
Umgekehrt kann ein digitales Gerät nicht ohne weiteres analoge Signale verstehen. **Analoge** Eingänge messen die anliegende Spannung und wandeln sie **proportional** in binäre Speicherwerte um. Wenn z.B. an einem analogen Eingang mit einer **Auflösung** von 10 bit bei maximal 5 Volt eine Spannung von 1,23 Volt anliegt, so bestimmt der Prozessor diese Spannung, indem er eine **Gegenspannung** anlegt und diese in $2^{10} - 1 = 1023$ gleich großen Schritten von $\frac{5000\ \text{mV}}{2^{10} - 1} = \frac{5000\ \text{mV}}{1023} \approx 4{,}89$ mV so oft erhöht, bis sie die zu messende Spannung von 1,23 Volt gerade ausgleicht und kein Strom mehr fließt. Die Zahl der benötigten Schritte wird dann als digitaler Messwert gespeichert. In diesem Beispiel sind es $\frac{1230\ \text{mV}}{4{,}89\ \text{mV}} = \frac{1230\ \text{mV}}{5000\ \text{mV}} \cdot 1023 =$ 251 Schritte, was in der Binärform $(00111\ 11011)_2$ gespeichert wird.

Übungen: Aufgaben zu Mikrocontrollern Nr. 7

4.0.1.4. Technische Daten des AT mega 328 Mikrocontrollers:

Taktfrequenz:	20 MHz
Busbreite	8 bit
Bauart:	RISC (Reduced Instruction Set Computer): 131 einfache Befehle, die sich (fast) alle in einem Zyklus bearbeiten lassen
Register:	3 x 32 bit
ROM:	1 KByte EEPROM
Arbeitsspeicher:	2 KByte SRAM
Datenspeicher:	32 KByte EEPROM (SD)
I/O:	6 analoge Eingänge mit 10 bit-Auflösung ⇒ $2^{10} = 1024$ Stufen für max 5 V/40 mA 14 digitale Ein-/Ausgänge mit max 5 V/40 mA, davon 6 mittels 8 bit-Pulsweitenmodulation (PWM) analog schaltbar ⇒ $2^8 = 256$ Zyklen pro Intervall

Der Mikrocontroller sitzt auf einem ziemlich spartanischen **Mainboard**, dem **Arduino Uno**, welches im Wesentlichen die Stecker bzw. Stabilisierungsschaltungen für die Stromversorgung, die I/O-Ports, einen Reset-Schalter und bereitstellt:



Übungen: Aufgaben zu Mikrocontrollern Nr. 8

4.0.1.5. Arbeitsweise des Mikrocontrollers

Die meisten Computerprogramme wie z.B. die bekannten Textverarbeitungen werden zunächst in einer **Programmiersprache** wie Java (z.B. openoffice) oder C++ (z.B. microsoft word) formuliert. Dabei handelt es sich um eine Teilmenge der englischen Sprache mit wenigen klar definierten Vokabeln. Das folgende Programm lässt z.B. die an Ausgang 13 angeschlossene LED im Sekundentakt an- und ausgehen. Die **Kommentare hinter den Schrägstrichen** werden dabei vom Computer ignoriert.

```
int ledPin = 13;                  // Variable für Pin 13 initialisieren
void setup(){                     // Beginn der Variablendefinition
      pinMode(ledPin, OUTPUT);    // Pin 13 als Ausgang definieren
}                                 // Ende der Variablendefinition
void loop(){                      // Beginn der Endlosschleife
      digitalWrite(ledPin, HIGH); // LED auf High-Pegel (5V)
      delay(1000);                // Eine Sekunde warten
      digitalWrite(ledPin, LOW);  // LED auf Low-Pegel (0V)
      delay(1000);                // Eine Sekunde warten
}                                 // Ende der Endlosschleife
```

Ein solches C++-Programm kann mit einem beliebigen Schreibprogramm oder einer Textverarbeitung am PC erstellt werden. In der Praxis verwendet man spezielle **Editoren**, die das Programm automatisch auf Rechtschreibung und Grammatik (**Syntax**) untersuchen. Das C++-Programm wird gespeichert und dann einem **Compiler** übergeben, der es in **Maschinensprache** übersetzt, welche über USB an den Mikrocontroller gesendet wird und vom **Regelwerk** des Prozessors direkt umgesetzt werden kann. Es handelt sich um eine Reihe von Hexadezimalzahlen, die die Befehlsnummer, die Kennzahl eines der 131 verfügbaren Befehle (z.B. kopiere, addiere, …) und die Adressen der zu ändernden Speicher enthalten:

```
1:    100000000C9461000C947E000C947E000C947E0095
2:    100010000C947E000C947E000C947E000C947E0098
3:    100020000C947E000C947E000C947E000C947E0058
4:    100030000C947E000C947E000C947E000C947E0048
5:    100040000C949D000C947E000C947E000C947E0019
6:    100050000C949D000C947E000C947E000C947E0028
7:    100060000C949D000C947E00000000002400270009
8:    100070002A00000000000250028002B0000000000DE
```

In der Regel verwendet man zur Programmentwicklung ein Entwicklungssystem (integrated development environment **IDE**), welches neben dem **Editor** und dem **Compiler** eine Bibliothek (**library**) an fertigen Hilfsprogrammen und vor allem ein Simulationsprogramm (**debugger**) zur Fehlersuche enthält.

Übungen: Aufgaben zu Mikrocontrollern Nr. 9 und 10