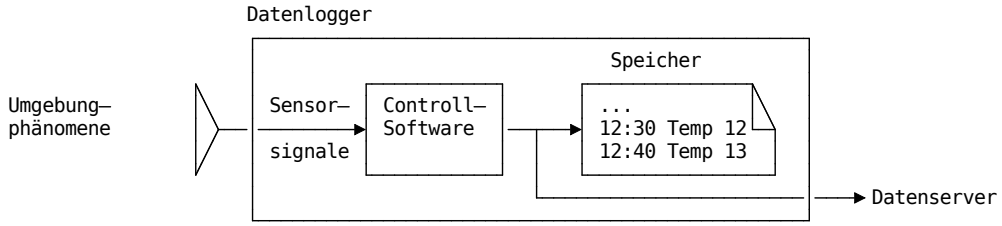
Softwareentwicklung für Microcontroller

Parameter	Kursinformationen
Veranstaltung:	`Prozedurale Programmierung / Einführung in die Informatik`
Semester	`Wintersemester 2022/23`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Anwendung von C++ auf bei der Mikrocontrollerprogrammierung`
Link auf Repository:	https://github.com/TUBAF-IfI-LiaScript/VL_ProzeduraleProgrammierung/blob/master/07_Microcontroller.md
Autoren	Sebastian Zug & André Dietrich & Galina Rudolf

Fragen an die heutige Veranstaltung ...

- Was ist die Besonderheit bei der Softwareentwicklung für eingebettete Systeme?
- Welche Randbedingungen bestimmen einen Messprozess?
- Warum brauchen wir eine neue Implementierung für die textuelle Ausgabe und können nicht unsere `cout` Konzepte weiter nutzen?
- Auf welchen "Standardisierungen" baut das Arduino Projekt auf?

Microcontroller als Datensammler



Ein Datenlogger ist eine prozessorgesteuerte Speichereinheit, welche Daten in einem bestimmten Rhythmus über eine Schnittstelle aufnimmt und auf einem Speichermedium ablegt. Dafür integriert die Datenloggerhardware Sensoren, Speicher und einen eingebetteten Rechner.

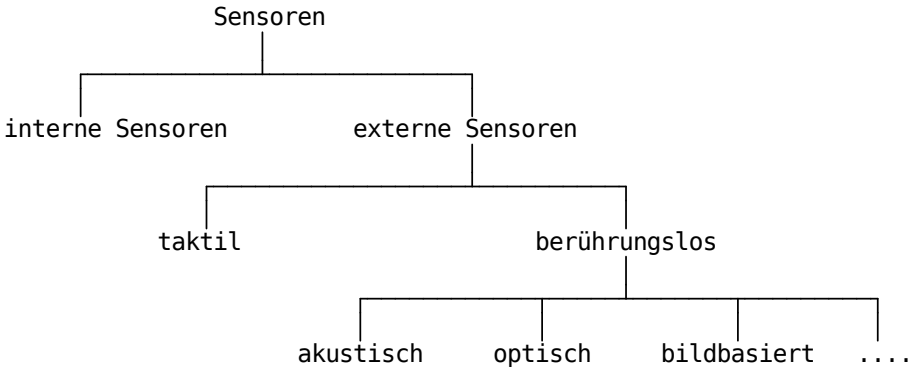
- Welches Umgebungsphänomen/e sollen erfasst werden?
- Wie lässt sich ein entsprechender Sensor mit dem Controller verknüpfen?
- Welche Qualität ist bei der Erfassung notwendig?
- Soll die Messdatenaufnahme periodisch oder eventgetrieben erfolgen?
- Wie lang ist der intendierte Messzeitraum - reichen Datenspeicher und Batteriekapazität?
- Braucht unser Logger ein Display oder eine andere Möglichkeit um seinen Zustand anzuzeigen?

Sensoren / Aktoren

Ein Sensor (von lateinisch *sentire*, „fühlen“ oder „empfinden“), auch als Detektor, (Messgrößen- oder Mess-)Aufnehmer oder (Mess-)Fühler bezeichnet, ist ein technisches Bauteil, das Umgebungsparameter als Messgröße quantitativ erfassen kann. Dabei werden physikalische Eigenschaften wie Temperatur, Feuchtigkeit, Druck, Helligkeit, Beschleunigung oder chemische z. B. pH-Wert, elektrochemisches Potential usw. als weiterverarbeitbares elektrisches Signal bereitgestellt.

Klassifikation von Sensoren

- intern/extern ... bezogen auf den Messgegenstand (Radencoder vs. Kamera)
- aktiv/passiv () ... mit und ohne Beeinflussung der Umgebung (Ultraschall vs. Kamera)
- Ergebnisdimension ... 1, 2, 2.5, 3D
- Modalitäten ... physikalische Messgröße



Beispiel: Ultraschallsensor

... Was war das noch mal, "Schallgeschwindigkeit"

Prinzip einer schallbasierten Entfernungsmessung [Wiora]

> Erkenntnis 1: Wir messen unter Umständen die eigentliche Messgröße nicht direkt.

Für eine gleichförmige Bewegung können wir den Weg als Produkt aus dem Messintervall und der halben Laufzeit abbilden.

$$s = v \cdot \frac{t}{2}$$

> Erkenntnis 2: Die Genauigkeit dieses Sensors wird über die Zeitauflösung des Mikrocontrollers definiert.

Leider gibt es ein Problem, die Schallgeschwindigkeit in Luft ist nicht konstant ist. Sie ist eine Funktion der Dichte ρ und des (adiabatischen) Kompressionsmoduls K und hängt damit vom Standort, der Temperatur usw. ab. Annäherungsweise gilt

$$v(m/s) = 331.3 + (0.606 \times t)$$

Versuchen wir eine kleine Fehlerabschätzung, wenn wir den Temperatureinfluss ignorieren.

```
1 def calcUSspeed(T):
```

```
2     return 331.3 + (0.606 * T)
3
4 print(calcUSspeed(25) / calcUSspeed(0))
```

Erkenntnis 3: Abhängigkeiten beeinflussen den Messprozess

[Wiora] By Georg Wiora (Dr. Schorsch) - Self drawn with Inkscape, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=353385

Programmiervorgang

Der Programmiervorgang für einen Mikrocontroller unterscheidet sich in einem Punkt signifikant von Ihren bisherigen C/C++ Aufgaben - die erstellten Programme sind auf dem Entwicklungssystem nicht ausführbar. Wir tauschen also den Compiler mit der Hardware aus. Dadurch "versteht" der Entwicklungsrechner die Anweisungen aber auch nicht.

Dabei zeigt sich aber auch der Vorteil der Hochsprachen C und C++, die grundsätzlichen Sprachinhalte sind prozessorunabhängig!

Besonderheiten

Die C++ Standardbibliothek ist für *kleine* eingebetteten Systeme nicht vollständig umgesetzt!

- Container
- Ausnahmen
- ...

Arduino Konzept

Das Arduino-Projekt wurde am *Interaction Design Institute Ivrea* (IDII) in Ivrea, Italien, ins Leben gerufen. Ausgangspunkt war die Suche nach einem preiswerten, einfach zu handhabenen Mikrocontroller der insbesondere für die Ausbildung von Nicht-Informatikern geeignet ist. Das anfängliche Arduino-Kernteam bestand aus Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino und David Mellis.

Nach der Fertigstellung der Plattform wurden leichtere und preiswertere Versionen in der Open-Source-Community verbreitet. Bereits Mitte 2011 wurde geschätzt, dass über 300.000 offizielle Arduinos kommerziell produziert worden waren, zwischenzeitlich wurden mehrere Millionen produziert.

Hardware

Das Arduino Projekt hat eine Vielzahl von unterschiedlichen Boards hervorgebracht, die eine unterschiedliche Leistungsfähigkeit und Ausstattung kennzeichnen. Das Spektrum reicht von einfachen 8-Bit Controllern bis hin zu leistungsstarken ARM Controllern, die ein eingebettetes Linux ausführen.

> **Merke:** Es gibt nicht **den** Arduino Controller, sondern eine Vielzahl von verschiedenen Boards.

Unser Controller, ein 32 Bit System, auf den im nachfolgenden eingegangen wird, liegt im mittleren Segment der Leistungsfähigkeit.

Erweitert werden die Boards durch zusätzliche `Shields`, die den Funktionsumfang erweitern.

Programmierung

Jedes Arduino-Programm umfasst zwei zentrale Funktionen - `setup` und `loop`. Erstgenannte wird einmalig ausgeführt und dient der Konfiguration, die zweite wird kontinuierlich umgesetzt.

Simulation time: 00:12.562 (58%)

HelloWorld.cpp

```
1  # define LED_PIN 13                 // Pin number attached to LED.
2  //const int led_pin_red = 13;
3
4  void setup() {
5      pinMode(LED_PIN, OUTPUT);       // Configure pin 13 to be a digit
         output.
6  }
7
8  void loop() {
9      digitalWrite(LED_PIN, HIGH);    // Turn on the LED.
10     delay(1000);                    // Wait 1 second (1000 millisecon
11     digitalWrite(LED_PIN, LOW);     // Turn off the LED.
12     delay(1000);                    // Wait 1 second.
13 }
```

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256
bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes
for local variables. Maximum is 2048 bytes.
```

Befehl	Bedeutung
`pinMode(pin_id, direction)`	Festlegung der Konfiguration eines Pins als Input / Output (`INPUT`, `OUTPUT`)
`digitalWrite(pin_id, state)`	Schreiben eines Pins, daraufhin liegen entweder (ungefähr) 3.3 V `HIGH` oder 0V `LOW` an

Eine Allgemeine Übersicht zu den Arduinobefehlen finden Sie unter folgendem Link.

Bibliotheken

Darüber hinaus existiert eine Vielzahl von Bibliotheken, die die Arbeit mit verschiedenen Sensoren/Aktoren vereinfachen und bei der Entwicklung von Anwendungslogik unterstützen.

Entwicklungsumgebung

Die Arduino-Entwicklungsumgebung fasst grundsätzliche Entwicklungswerkzeuge zusammen und richtet sich an Einsteiger.

> Zwischenzeitlich liegt eine Version 2.0 vor, die ein deutlich größeres Leistungspektrum mitbring. In den Übungen verwenden die Visual Studio Code Umgebung für die Entwicklung von Mikrocontroller Code.

Wo ist unser main()?

> Oha, wo ist denn unsere `main()` Methode geblieben? Diese sollte doch zwingender Bestandteil eines jeden C++ Programmes sein?

HelloWorld.cpp

```cpp
# define LED_PIN 13                 // Pin number attached to LED.

int main() {
    // Das ist unser Setup-Bereich
    init();   // Aufruf einer
    pinMode(LED_PIN, OUTPUT);

    // Endlosschleife als Entsprechung für Loop
    while(true){
      digitalWrite(LED_PIN, HIGH);
      delay(1000);
      digitalWrite(LED_PIN, LOW);
      delay(1000);
    }
}
```

```
Sketch uses 912 bytes (2%) of program storage space. Maximum is 32256
bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes
for local variables. Maximum is 2048 bytes.
```

Exkurs: Serielle Schnittstelle

Die serielle Schnittstelle ist eine umgangssprachliche Bezeichnung für eine Übertragungsmechanismus zur Datenübertragung zwischen zwei Geräten, bei denen einzelne Bits zeitlich nacheinander ausgetauscht werden. Die Bezeichnung bezieht sich in der umgangsprachlichen Verwendung:

- das Wirkprinzip generell, das dann verschiedenste Kommunikationsprotokoll meinen kann (CAN, I2C, usw.) oder
- die als EIA-RS-232 bezeichnete Schnittstellendefinition.

Für Mikrocontroller werden die zugehörigen Bauteile als *Universal Asynchronous Receiver Transmitter* (UART) bezeichnet.

Mögliche Anwendungen des UART:

- Debug-Schnittstelle
- Mensch-Maschine Schnittstelle - Konfiguration eines Parameters, Statusabfragen
- Übertragen von gespeicherten Werten für ein Langzeit-Logging von Daten
- Anschluss von Geräten - Sensoren, GNSS-Empfängern, Funkmodems
- Implementierung von "Feldbusse" auf RS485/RS422-Basis

Schreiben

Für das Schreiben auf der Seriellen Schnittstelle stehen in der Arduino Welt drei Funktionen bereit `println`, `print` und `write`. Diese können mit zusätzlichen Parametern versehen werden, um eine eingeschränkte Formatierung vorzunehmen.

UARTWrite.cpp

```
1  void setup() {
2    Serial.begin(9600);
3    Serial.println("Los geht's!");
4    Serial.println(5);
5    Serial.println(5, BIN);
6    Serial.println(5.34543, 2);
7    Serial.println("Fertig!");
8  }
9
10 void loop() {
11 }
```

```
Sketch uses 1828 bytes (5%) of program storage space. Maximum is 32256
bytes.
Global variables use 208 bytes (10%) of dynamic memory, leaving 1840
bytes for local variables. Maximum is 2048 bytes.


Los geht's!
5
101
5.35
Fertig!
```

Lesen

Die Umkehr der Kommunikationsrichtung ermöglicht es, Daten an den Mikrocontroller zu senden und damit bestimmte Einstellungen vorzunehmen.

ControlLED.cpp

```cpp
int incomingByte = 0; // for incoming serial data

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);   // LED Pin als Output
}

void loop() {
  // any data received?
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();

    // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```

```
Sketch uses 1884 bytes (5%) of program storage space. Maximum is 32256
bytes.
Global variables use 202 bytes (9%) of dynamic memory, leaving 1846
bytes for local variables. Maximum is 2048 bytes.
```

Aufgabe: Schalten Sie die LED die mit Pin 13 Verbunden ist mit einem A an und mit einem B aus. Dabei soll die LED mindestens 3s angeschaltet bleiben.

Aufgabe: Erweitern Sie das Programm, so dass mit 'AN' und 'AUS' die Aktivierung umgesetzt werden kann. Gehen Sie davon aus, dass der Nutzer auch kleine Buchstaben verwenden kann. (Hilfen String Klasse, Arduino Reference)

Seriellen Daten in der Arduino IDE

Der in der Arduino IDE eingebettete Serial Monitor ist eine Möglichkeit die über die Serielle Schnittstelle empfangenen Daten auszulesen. Mit der richtigen Konfiguration von

- USB Port (in der Entwicklungsumgebung selbst) und
- Baudrate (im unteren Teil des Monitors)

werden die Texte sichtbar. Sie können die Informationen speichern, indem Sie diese Markieren und in eine Datei kopieren. Sofern Sie auf eine gleiche Zahl von Einträgen pro Zeile achten, können die Daten dann als csv) (*Comma-separated values*) Datei zum Beispiel mit Tabellenkalkulationsprogrammen geöffnet werden.

> **Merke:** Die Einblendung des Zeitstempels garantiert die Zuordenbarkeit der kommunizierten Daten. Der Zeitstempel wird dabei vom Entwicklungsrechner generiert.

Die Eingabe von Daten, die an den erfolgt in der Zeile unter dem Ausgabefeld. Erst mit dem Betätigen von Enter, werden die Daten tatsächlich übertragen.

Der Plotter ermöglicht die unmittelbare grafische Darstellung der Daten, die über die serielle Schnittstelle ausgetauscht werden.

Ein Datensatz besteht aus einer einzelnen Zeile, die durch einen Zeilenumbruch (`\n` oder `Serial.println`) begrenzt wird.

Neben dem einfachen plotten von Zahlen können diese auch beschriftet werden. Beschriftungen können entweder einmalig im Setup gesetzt werden ODER sie können mit jeder Zeile übergeben werden.

```
Sensor1:30, Sensor2:45\n.
```

Anmerkungen:

1. Unser Board generiert automatisch eine Ausgabe zur Version, Hersteller usw., die den Plotter "irritiert". Entsprechend zeigte sich die Darstellung der Legende in der zeilenbasierten Darstellung als stabiler.
2. Wenn Sie die zeilenbasierten Beschriftungen verwenden, fügen Sie kein Leerzeichen nach dem ":" ein!

> **Merke:** Die Serielle Schnittstelle kann aus allen Programmiersprachen heraus genutzt werden. Sie können also in C, C++, Python usw. eigene Programme für das Auslesen und die Interpretation der Daten schreiben.

Unser Controller

Kategorie	Features laut Webseite	Bedeutung
Controller	*EMW3166 Wifi module*	Eingebetteter Controller mit WLAN Schnittstelle
Peripherie	*Audio codec chip*	Hardware Audioverarbeitungseinheit
	Security encryption chip	Verschlüsselungsfeatures in einem separaten Controller
	2 user button	
Aktoren	*1 RGB light*	
	3 working status indicator	*WiFi*, *Azure*, *User* Leds auf der rechten Seite
	Infrared emitter	Infrarot Sender für die Nutzung als Fernbedienung
	OLED,128×64	Display mit einer Auflösung von 128 x 64 Pixeln
Sensoren	*Motion sensor*	Interialmessystem aus Beschleunigungssensor und Gyroskop
	Magnetometer sensor	
	Atmospheric pressure sensor	
	Temperature and humidity sensor	
	Microphone	
Anschlüsse	*Connecting finger extension interface*	

[3]

Worin unterscheidet sich der Controller von einem willkürlich ausgewählten Laptop?

Parameter	AZ3166	Laptop
Bandbreite	32Bit	64Bit
Taktrate	100Mhz	3.5Ghz
Arbeitsspeicher	256kBytes	8 GBytes
Programmspeicher	1 MByte	
Energieaufnahme	0.x Watt	100 Watt
Kosten	40 Euro	> 500 Euro
Einsatz	Spezifisches eingebettetes System	Vielfältige Einsatzmöglichkeiten

Zum Vergleich, eine Schreibmaschinenseite umfasst etwa 2KBytes.

Die Dokumentation der zugehörige *Application Programming Interface* (API) finden Sie unter Link. Hier stehen Klassen bereit, die Einbettung der Sensoren, LEDs, des Displays kapseln und die Verwendung vereinfachen.

[3] Produktwebseite Firma MXChip, AZ3166 Procuct Details Link

Anwendungsfall

Bestimmen Sie die Periodendauer eines Fadenpendels bzw. die Erdbeschleunigung bei einer bekannten Fadenlänge!

$$T_0 = 2\pi\sqrt{\frac{l}{g}}$$

Frage: Welche Sensoren können wir dafür nutzen?

Unser Ansatz unterscheidet sich davon, wir nutzen den Magnetkompass unseres Boards, der auf die Präsenz von magnetoresistiven Materialien in der Umgebung reagiert Link.

Der Sensor ist über den I2C Bus an unser Board angeknüpft. Dieser überträgt die Messdaten serialisiert an den Controller. Wir müssen also, um die Daten nutzen zu können sowohl eine Instanz der Sensorklasse `LIS2MDLSensor` als auch eine Instanz der Busklasse `DevI2C` erzeugen. Dem Sensor wird dabei das erzeugte Busobjekt im Konstruktor übergeben.

ReadingMagdnetometerValues.ino

```
#include "LIS2MDLSensor.h"

DevI2C ext_i2c(D14,D15);
LIS2MDLSensor lis2mdl(ext_i2c);

int axes[3];

void setup(){
    lis2mdl.init(NULL);
    Serial.begin(115200);
}

void loop(){
    lis2mdl.getMAxes(axes);
    Serial.printf("%d, %d, %d\n", axes[0], axes[1], axes[2]);
    delay(10);
}
```

Bringen wir einen magnetorresistives Objekt in die Nähe des Sensors, so kann er dessen Auswirkung auf das Magnetfeld der Erde messen.

Für unser Pendel sieht der Verlauf dann entsprechend wie folgt aus:

Eine Lösung für die Extraktion der Periodendauer finden Sie in unserem Vorlesungsverzeichnis Link. Gelingt es Ihnen eine bessere Lösung zu entwickeln?

```
import numpy as np
import matplotlib.pyplot as plt

periods = np.array([1524, 1541, 1541, 1541, 1541, 1541, 1531, 1541, 15

periods = periods / 1000  # ms -> s
length = (periods / ( 2 * np.pi))**2 * 9.81  # vgl. Gleichung Fadenpen

fig, ax = plt.subplots()
ax.hist(length)
plt.title("Gaussian Histogram")
plt.xlabel("Periodendauer in ms")
plt.ylabel("Häufigkeit")

fig  # notwendig für die Ausgabe in LiaScript sonst plt.show()
```

Quiz

Microcontroller als Datensammler

Sensoren und Aktoren

Was sind Sensoren bei einem Arduino?

- ( ) Bauteile, die Umgebungsparameter als Messgrößen erfassen
- ( ) Bauteile, die das Arduino-Board durch Anweisungen steuern
- ( ) Bauteile, die Daten vom Arduino-Board empfangen

Arduino Konzept

Hardware

Wodurch kann ein Arduino erweitert werden um den Funktionsumfang zu erhöhen?

- ◯ Sword
- ◯ Shield
- ◯ Linux

Programmierung

Was sind die Hauptbestandteile eines Arduino-Programms?

- ◯ `main`-Funktion
- ◯ `main`- und `init`-Funktion
- ◯ `setup`- und `loop`-Funktion

Serielle Schnittstelle

Schreiben

Wie lautet die Ausgabe dieses Programms?

```
void setup() {
  Serial.begin(9600);
  Serial.print("a");
  Serial.print("b");
  Serial.print(2, BIN);
  Serial.println(3.1415, 2);
}

void loop() {
}
```

Lesen

Ändern Sie folgendes Programm so ab, dass die LED nach der Eingabe eines `L` eine Sekunde lang an ist und dann wieder erlischt.

ControlLED.cpp

```
int incomingByte = 0; // for incoming serial data

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);   // LED Pin als Output
}

void loop() {
  // any data received?
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();

    // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```

```
Sketch uses 1884 bytes (5%) of program storage space. Maximum is 32256
bytes.
Global variables use 202 bytes (9%) of dynamic memory, leaving 1846
bytes for local variables. Maximum is 2048 bytes.
```

Mögliche Lösung

Mögliche Lösung der Aufgabe:

```
int incomingByte = 0; // for incoming serial data

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

```
  pinMode(13, OUTPUT);    // LED Pin als Output
}

void loop() {
  // any data received?
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();

    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);

    // if the incoming byte is 'L', turn the LED on:
    if (incomingByte == 'L') {
      digitalWrite(13, HIGH);
      delay(1000);  // wait for 1 second
      digitalWrite(13, LOW);  // turn the LED off
    }

  }
}
```