Einfaches .ino - File

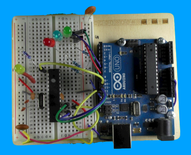
test.ino Compiler → Objectfile Linker → Ausführbarer Code mit Daten

```
int anzahl = 3;
int preis = 2;
void printBetrag() {
  Serial.println(anzahl*preis);
}

void setup() {
  printBetrag();
};
```

Symbole **anzahl** und **preis** mit je 2 Byte reserviertem Speicherplatz und dem angegebenen Inhalt.

Symbol **printBetrag** mit Code.

Der Code und die Datenbereiche werden auf den Chip kopiert und dort ausgeführt.



Einfache Includes

test.ino Compiler → Objectfile Linker → Ausführbarer Code mit Daten

```
#include "variablen.h"
#include "funktionen.h"

void setup() {
  printBetrag();
};
```

Symbole **anzahl** und **preis** mit je 2 Byte reserviertem Speicherplatz und dem angegebenen Inhalt.

Symbol **printBetrag** mit Code.

Der Code und die Datenbereiche werden auf den Chip kopiert und dort ausgeführt.

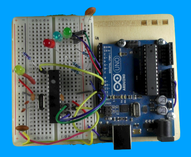
variablen.h

```
int anzahl = 3;
int preis = 2;
```

funktionen.h

```
void printBetrag() {
  Serial.println(anzahl*preis);
}
```

Alle Includes stehen im .ino - File. Dadurch wird ein grosser Quelltext erstellt, der in einem Stück verarbeitet wird. Das gibt exakt denselben Code wie in der ersten Folie.



Aufgeteilte Funktion

test.ino

```
#include "variablen.h"
#include "funktionen.h"

void setup() {
  printBetrag();
};
```

Compiler

Symbole **anzahl** und **preis** mit je 2 Byte reserviertem Speicherplatz und dem angegebenen Inhalt.

funktionen.h

```
void printBetrag();
```

variablen.h

```
int anzahl = 3;
int preis = 2;
```

funktionen.cpp

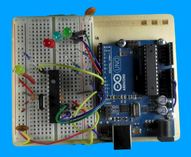
```
#include "funktionen.h"
#include "variablen.h"
#include <arduino.h>

void printBetrag() {
  Serial.println(anzahl*preis);
}
```

Compiler

Symbole **anzahl** und **preis** mit je 2 Byte reserviertem Speicherplatz und dem angegebenen Inhalt.

Symbol **printBetrag** mit Code.

Das verursacht einen **multiple definition** error



Aufgeteilte Funktion - korrekt

test.ino

```
#include "variablen.h"
#include "funktionen.h"

// Definition
int anzahl = 3;
int preis = 2;

void setup() {
  printBetrag();
};
```

Compiler

Symbole **anzahl** und **preis** mit je 2 Byte reserviertem Speicherplatz und dem angegebenen Inhalt.

funktionen.h

```
void printBetrag();
```

variablen.h

```
// Deklaration
extern int anzahl;
extern int preis;
```

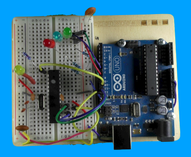
funktionen.cpp

```
#include "funktionen.h"
#include "variablen.h"
#include <arduino.h>

void printBetrag() {
  Serial.println(anzahl*preis);
}
```

Compiler

Symbole **anzahl** und **preis** werden nicht erzeugt. Die Namen sind aber bekannt.

Symbol **printBetrag** mit Code.



Includes im .h - File

test.ino

```
#include "variablen.h"
#include "funktionen.h"

// Definition
int anzahl = 3;
int preis = 2;

void setup() {
  printBetrag();
};
```

funktionen.h

```
#include "variablen.h"
void printBetrag();
```

variablen.h

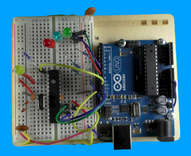
```
// Deklaration
extern int anzahl;
extern int preis;
```

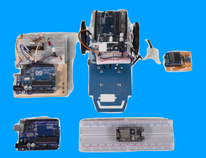
funktionen.cpp

```
#include "funktionen.h"
#include <arduino.h>

void printBetrag() {
  Serial.println(anzahl*preis);
}
```

Compiler → Das verursacht einen **redefinition** error

Mit variablen.h wird die Deklaration von anzahl und preis importiert. funktionen.h importiert diese nochmals.





Includes im .h - File - korrekt

test.ino

```
#include "variablen.h"
#include "funktionen.h"

// Definition
int anzahl = 3;
int preis = 2;

void setup() {
  printBetrag();
};
```

variablen.h

```
#ifndef VARIABLEN_H
#define VARIABLEN_H

// Deklaration
extern int anzahl;
extern int preis;

#endif
```

oder

```
#pragma once

// Deklaration
extern int anzahl;
extern int preis;
```

In diesem einfachen Fall hätte das Problem auch durch Weglassen von #include "variablen.h" im .ino - File gelöst werden können. Oft ist das aber nicht möglich und darum sieht man diese Lösung sehr oft.