

7. Arduino-Stammtisch: Tool-Chain und eigene Bibliotheken

Axel

Attraktor e.V.

6. August 2012

Tool-Chain und eigene Bibliotheken

- 1 Was passiert im Hintergrund, wenn man auf den Verify-Button clickt?
- 2 Wie kann die avr-libc verwendet werden?
- 3 Wie bindet man Assembler-Befehle in eigenen Code ein?
- 4 Wie bindet man Bibliotheken ein?
- 5 Wie erstellt man eigene Bibliotheken?

Was passiert im Hintergrund, wenn man auf den Verify/Upload-Button clickt?

- 1 Leichte Vorverarbeitung der Quellen (Leerzeichen, Kommentare, Zeichen-Kodierung, Funktions-Delkarationen)
- 2 Zusammenkopieren von Sketch-Teilen (Tabs) und Einfuegen von `Arduino.h` und `main.cpp`
- 3 Settings zum Arduino werden aus `../hardware/arduino/boards.txt` eingelesen.
- 4 Jede Quellcode-Einheit wird separat mit `gcc` zu Objekt-Dateien übersetzt. (Sketch, Bibliotheken, Support-Quellcode).
- 5 Alle Objekt-Dateien ausser dem Sketch werden zu einer statischen Bibliothek übersetzt
- 6 Das Sketch-Objekt wird gegen diese Bibliothek gelinkt.
- 7 Es entsteht eine `.hex` Datei
- 8 Diese wird mit dem eingestellten Programmer (`hardware/arduino/programmers.txt`) aufgespielt.

Notizen zum Build-Prozess

- Es gibt ein Tutorial unter <http://arduino.cc/en/Hacking/BuildProcess> welches aber nicht 100%ig aktuell ist.
- Preferences-Variablen:
 - ▶ `build.verbose=true` dokumentiert die Compilierung.
 - ▶ `upload.verbose=true` dokumentiert den Upload.
 - ▶ `upload.verify` verhindert leider nicht die erneute Compilierung
- Mit `build.path` kann man den Pfad setzen, in dem die Dateien zur Compilierung gespeichert werden.
- Mit `avr-objdump -d` kann man sich den Inhalt von Objekt-Dateien ansehen.
- Compiler- und Linker-Flags sind hart im arduino-Source verdrahtet
- `main.cpp` führt bei `loop()` im Hintergrund noch serielle Kommunikation durch.

Wie kann die avr-libc verwendet werden?

- In der Doku leicht zu übersehen: “It links against AVR Libc and allows the use of any of its functions”
- Die “Library Reference” bietet eine Übersicht über viele nützliche Funktionen:
<http://www.nongnu.org/avr-libc/user-manual/modules.html>
- Verwendung mit `#include` wie Arduino-Bibliotheken
- Funktionen der Standard C-Bibliothek und viele AVR-spezifische Funktionen
- Insbesondere Hardware-nahe Funktionen (Bootloader, EEPROM, CPU, I/O)
- Integration mit “Arduino-Magic” teilweise nicht möglich, z.B. `string` \leftrightarrow `String`

Wie bindet man Assembler-Befehle in eigenen Code ein?

- GCC-Befehl `asm()`: Einfachster Fall `asm(“nop”);`
- `__asm__()` geht auch
- `volatile` oder `__volatile__` verhindern Compiler-Probleme

- Mehrzeiler:

```
asm(“nop” “\n\t”  
    “nop” “\n\t”  
    );
```

- Verbindung mit C-Quellen:

```
asm(code : output operand list : input operand list [: clobber list]);  
Beispiel:
```

```
asm(“in %0, %1” : “=r” (value) : “l” %(_SFR_IO_ADDR(PORTD)) );
```

- AVR-Libc Inline Assembler Cookbook

http://www.nongnu.org/avr-libc/user-manual/inline_asm.html

- Einfache Register-Manipulation geht auch mit Konstanten (z.B. `PORTA` und Bit-Operationen).

Wie bindet man Bibliotheken ein?

- Externe Bibliotheken können in `/sketchbook/libraries` abgelegt werden und stehen dann über die "Import Library"-Funktion zur Verfügung.
- Es wird ein `#include <bibliotheksname.h>` eingefügt.
- Dies hat "magisch" verschiedene Konsequenzen:
 - ▶ Bei "Verify" wird die Bibliothek mit ins Build-Verzeichnis kopiert.
 - ▶ Die Dateien der Bibliothek werden einzeln kompiliert
 - ▶ Die Bibliothek wird mit gelinkt.

Die Struktur einer Bibliothek “Attraktor”

- Ein Verzeichnis “Attraktor”, das enthalten muss:
 - ▶ `Attraktor.cpp`
 - ▶ `Attraktor.h`
- .. und verschiedenes weiteres enthalten kann:
 - ▶ weitere C,C++,Assembler-Quellen
 - ▶ ein Verzeichnis `utility` mit weiteren Quellen
 - ▶ ein Verzeichnis `examples` mit Beispielen
 - ▶ `keywords.txt` mit Syntax-Highlighting

Wie erstellt man eigene Bibliotheken?

Die Datei Attraktor.h

```
#ifndef ATTRAKTOR_H
#define ATTRAKTOR_H

#include "Arduino.h"

class Attraktor {

public:
    Attraktor(int wochentag);
    int getMate();

private:
    int benutzeAutomat();
};
#endif
```

Wie erstellt man eigene Bibliotheken?

Die Datei Attraktor.c

```
Attraktor::Attraktor(int wochentag) {  
    ...  
}  
  
void Attraktor::benutzeAutomat() {  
  
}  
  
void Attraktor::getMate() {  
    return benutzeAutomat();  
}
```

Wie erstellt man eigene Bibliotheken?

Verwendung der Bibliothek:

```
Attraktor attraktor(1);  
  
int mate;  
  
void loop() {  
    mate = attraktor.getMate();  
}
```

Notizen zu Bibliotheken

- Ein volles Tutorial gibt es unter <http://arduino.cc/en/Hacking/LibraryTutorial>
- Es ist leider nicht (einfach) möglich, Arduino-Bibliotheken aus eigenen Bibliotheken zu benutzen.