Hacklace Programmierung

Um eigene Apps zu schreiben oder die Firmware des Hacklace zu verändern, benötigst du die Arduino-Entwicklungsumgebung, die du kostenlos herunterladen kannst. Ausführliche Informationen zur Installation und Benutzung sind auf der Arduino-Website zu finden.

Installation der Hacklace-Software

Am einfachsten ist es, die Hacklace-Software als ZIP-Datei herunterzuladen (Knopf „Download ZIP“ rechts unten) und in deinem Sketch-Ordner zu entpacken.

Es gibt folgende Ordner:

- **Hacklace_Main** enthält die Firmware, welche auf dem Hacklace vorprogrammiert ist.
- **HL_Hello_World** ist ein einfaches „Hello World“-Beispiel.
- **Hacklace_Games** sind Spiele, die von Schülern der Karl-Kübel-Schule in Bensheim programmiert wurden.
- **libraries** enthält die Library zum Ansteuern des Hacklace.

Nach dem Starten der Entwicklungsumgebung wählst du unter „Tools –> Board“ den Punkt „Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328“ aus.

Eine App schreiben

Um eine eigene App zu schreiben, startest du am besten mit der Example-App. Kopiere die Datei „HL_ExampleApp.h“ und benenne sie um.

Zuerst gibst du der App einen Namen, indem du in der folgenden Zeile „ExampleApp“ ersetzt.

```
// ##### enter your app name here #####
#define APP_NAME  MeineNeueApp    // <--- here
```

Eine App besteht im Wesentlichen nur aus drei Funktionen: setup, run und finish.

setup

Wird nur einmal aufgerufen, wenn die App startet. Dient zum Initialisieren deiner App.

run

Wird regelmäßig alle 10 Millisekunden aufgerufen. Hier spielt sich die eigentliche Aktivität deiner App ab.

finsh

Wird einmalig aufgerufen, wenn die App beendet wird. Hier sollten eventuell nötige Aufräumarbeiten erledigt werden.

Alle Routineaufgaben für den Betrieb des Hacklace werden von dem Objekt „HL“ der Klasse „Hacklace“ erledigt. Du kannst die Methoden dieses Objekts in deiner App verwenden. Um z. B. den Bildschirm zu löschen, wird die Methode „clearDisplay“ aufgerufen.

```
HL.clearDisplay();
```

Weitere nützliche Methoden sind in der Beschreibung der Hacklace-Klasse aufgeführt.

Möchtest du Informationen speichern, die nicht zwischen den Aufrufen verlorengehen sollen, sondern auf die deine App dauerhaft zugreifen kann (z. B. den Zustand der App), so müssen hierfür statische Variablen angelegt werden. Dazu wird die Variable zunächst deklariert

```
class APP_CLASSNAME : public Hacklace_App
{
    public:
        (...)
    private:
        // Use static variables if you want to preserve their content
between calls.
        // ##### enter your member variables here... #####
        static byte meine_variable;
};
```

und dann definiert und gegebenenfalls initialisiert.

```
/**************************
 * static class variables *
 **************************/
// Note:  Each static variable name must be prefixed with "APP_CLASSNAME::"
// ##### list your static variables here... #####
byte    APP_CLASSNAME::meine_variable = 123;
```

Wenn deine App durch Parameter aus dem EEPROM konfiguriert werden soll (siehe „Datenformat“), kannst du in der setup-Funktion weitere Bytes aus dem EEPROM lesen. Achte darauf, dass die setup-Funktion den Zeiger auf die nächste Adresse nach den Parametern zurückgibt. Im Beispiel wird dazu der EEPROM-Zeiger „ee“ nach jedem gelesenen Byte inkrementiert.

```
const unsigned char* APP_CLASSNAME::setup(const unsigned char* ee)
{
    byte param1, param2;

    // load two parameters
    param1 = eeprom_read_byte(ee++);
    param2 = eeprom_read_byte(ee++);
    (...)
```

```
    HL.clearDisplay();
    return( ee );
}
```

Schließlich muss deine neue App noch in die App-Registry aufgenommen werden. Die App-Registry in der Datei „HL_AppRegistry.h“ ist eine Liste aller Apps. Die Position in dieser Liste legt die App-ID fest. Deine App wird per Include-Befehl eingebunden und ihr Name mit einem vorangestellen „&“ in der app_registry eingetragen.

```
// include Hacklace apps
#include "HL_AnimationApp.h"
(...)
#include "HL_MeineNeueApp.h"

// list of all available apps
const Hacklace_App* app_registry[MAX_APPS] PROGMEM = {
    &AnimationApp,            // app-ID 0
    (...)
    &ExampleApp,            // app-ID 28
    &MeineNeueApp,            // app-ID 29
    &ExampleApp,            // app-ID 30
    &DownloadApp            // app-ID 31 ResetApp
};
```

In diesem Beispiel hat deine App die App-ID=29. Du kannst deine App verwenden, indem du ihre App-ID ins EEPROM schreibst (siehe hier).

Der Hacklace-Library-Ordner enthält weitere fertige Apps. Diese können dir als zusätzliche Beispiele dienen.

Klasse Hacklace_AppEngine

siehe Hacklace2 Sourcecode

Klasse Hacklace

siehe Hacklace2 Sourcecode

Verschiedenes

Flashen des Bootloaders

Das Hacklace ist bereits mit dem Arduino-Bootloader vorgeflasht. Solltest du dennoch einen neuen Bootloader aufspielen wollen, so kannst du den Bootloader für das Board „Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328“ benutzen. Anschließend müssen noch die Fuses wie folgt neu gesetzt

werden.

```
BODLEVEL  = DISABLED
RSTDISBL  = [ ]
DWEN      = [ ]
SPIEN     = [X]
WDTON     = [ ]
EESAVE    = [ ]
BOOTSZ    = 1024W_3C00
BOOTRST   = [X]
CKDIV8    = [ ]
CKOUT     = [ ]
SUT_CKSEL = EXTXOSC_8MHZ_XX_16KCK_14CK_65MS

EXTENDED = 0xFF
HIGH     = 0xDA
LOW      = 0xFF
```

Der Grund hierfür ist, dass beim Flashen des Bootloaders der Brown-out-Detektor über die Fuses eingeschaltet wird. Dieser benötigt aber immer einen kleinen Strom - auch wenn das Hacklace ausgeschaltet ist. Damit sich die Batterie nicht unnötig entlädt, sollte der Brown-out-Detektor deshalb besser ausgeschaltet werden.

Hacklace, Elektronik, Bausatz, Programmierung