Hacklace Programming

For developing your own apps or modifying the Hacklace firmware you need to install the Arduino development platform which is available for free. Detailed information about how to install and how to use Arduino can be found on the Arduino website.

Installing the Hacklace software

Download the Hacklace software as a zip archive (button „Download ZIP“ down to the right) and expand it in your sketch folder.

You will find the following folders:

- **Hacklace_Main** contains the firmware, that is pre-programmed on the Hacklace.
- **HL_Hello_World** is a simple „Hello World“ example.
- **Hacklace_Games** are games that were developed by pupils of the Karl-Kübel-Schule in Bensheim, Germany..
- **libraries** contains the library for controlling the Hacklace.

After starting the IDE you have to go to „Tools –> Board“ and choose „Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328“.

Writing an App

To program your own app just start with the Example-App. Copy the file „HL_ExampleApp.h“ and rename it.

First you name your app by replacing „ExampleApp“ in the following line of code.

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

An app consists mainly of three methods: setup, run and finish.

setup

Will be called once when the app is started. Used to initialise your app.

run

Will be called periodically every 10 milliseconds. This is where the main activity of your app happens.

finsh

Will be called once when your app is terminated. Used to do some clean-up if necessary.

All routine tasks for operating the Hacklace are performed by the „HL“ object which is an instance of class „Hacklace“. You can use this object and its methods in your app. E. g. to clear the display call the „clearDisplay“ method.

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

More methods are listed in the description of the class Hacklace.

If you want to use data that is kept between two calls of the run method (e. g. the state of the app) you have to use static variables. First you declare a variable

```
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 my_variable;
};
```

and then you have to define and maybe initialise it.

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

If you want to configure your app by parameters from the EEPROM (see „Datenformat“) you can read the parameters in the setup method. Take care that your setup method returns a pointer to the next EEPROM address after your parameters. For this, in the following example the pointer „ee“ is incremented after reading each byte.

```
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 );
```

```
}
```

Finally your app must be entered in the app-registry residing in the file „HL_AppRegistry.h“. The app-registry lists all apps and the position of an app defines its app-ID. The source code of your app is included and its name preceeded by „&“ goes into the app-registry.

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

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

In this example your app will be assigned the app-ID 29. You can use your app by writing its app-ID into the EEPROM (see here).

In the Hacklace library folder you will find more apps that can serve you as examples.

Class Hacklace_AppEngine

siehe Hacklace2 sourcecode

Class Hacklace

siehe Hacklace2 sourcecode

Miscellaneous

Flashing the Bootloader

The Hacklace comes pre-flashed with the Arduino bootloader. In case you want to re-flash the bootloader you can use the one for „Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328“. After flashing you have to set the fuses as follows.

```
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
```

The reason is that flashing the bootloader using the Arduino-IDE will switch on the brown-out detector. This detector will always consume a little bit of battery power (approx. 1 microamp) - even when the hacklace is switched off. To save the battery from discharging you should turn off the brown-out detector.

Hacklace, electronics, kit, programming, Arduino