HO
GENT

Deel 7: bibliotheken

HO
GENT

Wat is een bibliotheek?

Een bibliotheek is een manier om complexe bewerkingen die toegespitst zijn op één onderwerp eenvoudiger te maken. Dit kan door het definiëren van objecten, eigenschappen en functies voor dit onderwerp.
Nemen we als voorbeeld de LCD-bibliotheek LiquidCrystal. Met deze bibliotheek kunnen we een LC display aansturen.

In het kader rechts zien we alle gedefinieerde functies. Deze functies kunnen we terugvinden in de bibliotheekdefinitie. Deze bestaat uit LiquidCrystal.h en LiquidCrystal.cpp die samen de bibliotheek vormen.

```
LiquidCrystal()
begin()
clear()
home()
setCursor()
write()
print()
cursor()
noCursor()
blink()
noBlink()
display()
noDisplay()
scrollDisplayLeft()
scrollDisplayRight()
autoscroll()
noAutoscroll()
leftToRight()
rightToLeft()
createChar()
```

```
#ifndef LiquidCrystal_h
#define LiquidCrystal_h

#include <inttypes.h>
#include "Print.h"

// commands
#define LCD_CLEARDISPLAY 0x01
#define LCD_RETURNHOME 0x02
#define LCD_ENTRYMODESET 0x04
#define LCD_DISPLAYCONTROL 0x08
#define LCD_CURSORSHIFT 0x10
#define LCD_FUNCTIONSET 0x20
#define LCD_SETCGRAMADDR 0x40
#define LCD_SETDDRAMADDR 0x80

// flags for display entry mode
#define LCD_ENTRYRIGHT 0x00
#define LCD_ENTRYLEFT 0x02
#define LCD_ENTRYSHIFTINCREMENT 0x01
#define LCD_ENTRYSHIFTDECREMENT 0x00

// flags for display on/off control
#define LCD_DISPLAYON 0x04
#define LCD_DISPLAYOFF 0x00
#define LCD_CURSORON 0x02
#define LCD_CURSOROFF 0x00
#define LCD_BLINKON 0x01
#define LCD_BLINKOFF 0x00

// flags for display/cursor shift
#define LCD_DISPLAYMOVE 0x08
#define LCD_CURSORMOVE 0x00
#define LCD_MOVERIGHT 0x04
#define LCD_MOVELEFT 0x00

// flags for function set
#define LCD_8BITMODE 0x10
#define LCD_4BITMODE 0x00
#define LCD_2LINE 0x08
#define LCD_1LINE 0x00
#define LCD_5x10DOTS 0x04
#define LCD_5x8DOTS 0x00
```

Instruction	Code										Description	Execution time**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged.	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40uS
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40uS
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged.	40uS
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40uS
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40uS
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40uS
Read busy-flag and address counter	0	1	BF	CGRAM / DDRAM address							Reads Busy-flag (BF) indicating internal operation is being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0uS
Write to CGRAM or DDRAM	1	0	write data								Writes data to CGRAM or DDRAM.	40uS
Read from CGRAM or DDRAM	1	1	read data								Reads data from CGRAM or DDRAM.	40uS

We zien hoe in LiquidCrystal.h de verschillende low-level functies en vlaggen gedefinieerd worden.
Dit is nog maar het eerste deel van het bestand.

```
class LiquidCrystal : public Print {
public:
  LiquidCrystal(uint8_t rs, uint8_t enable,
                uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
                uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7);
  LiquidCrystal(uint8_t rs, uint8_t rw, uint8_t enable,
                uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
                uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7);
  LiquidCrystal(uint8_t rs, uint8_t rw, uint8_t enable,
                uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3);
  LiquidCrystal(uint8_t rs, uint8_t enable,
                uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3);

  void init(uint8_t fourbitmode, uint8_t rs, uint8_t rw, uint8_t enable,
            uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
            uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7);

  void begin(uint8_t cols, uint8_t rows, uint8_t charsize = LCD_5x8DOTS);

  void clear();
  void home();

  void noDisplay();
  void display();
  void noBlink();
  void blink();
  void noCursor();
  void cursor();
  void scrollDisplayLeft();
  void scrollDisplayRight();
  void leftToRight();
  void rightToLeft();
  void autoscroll();
  void noAutoscroll();

  void setRowOffsets(int row1, int row2, int row3, int row4);
  void createChar(uint8_t, uint8_t[]);
  void setCursor(uint8_t, uint8_t);
  virtual size_t write(uint8_t);
  void command(uint8_t);

  using Print::write;
private:
  void send(uint8_t, uint8_t);
  void write4bits(uint8_t);
  void write8bits(uint8_t);
  void pulseEnable();

  uint8_t _rs_pin; // LOW: command.  HIGH: character.
  uint8_t _rw_pin; // LOW: write to LCD.  HIGH: read from LCD.
  uint8_t _enable_pin; // activated by a HIGH pulse.
  uint8_t _data_pins[8];

  uint8_t _displayfunction;
  uint8_t _displaycontrol;
  uint8_t _displaymode;

  uint8_t _initialized;

  uint8_t _numlines;
  uint8_t _row_offsets[4];
};

#endif
```

In het tweede deel zien we hoe een klasse LiquidCrystal gedefinieerd wordt en hoe de high-level functies beschreven worden.

Vervolgens krijgen we nog de definitie van enkele intermediate-level functies en een aantal variabelendefinities.

De uitwerking van al deze definities vinden we terug in het LiquidCrystal.cpp bestand.

```
LiquidCrystal::LiquidCrystal(uint8_t rs, uint8_t rw, uint8_t enable,
                             uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
                             uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7)
{
  init(0, rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7);
}

LiquidCrystal::LiquidCrystal(uint8_t rs, uint8_t enable,
                             uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
                             uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7)
{
  init(0, rs, 255, enable, d0, d1, d2, d3, d4, d5, d6, d7);
}

LiquidCrystal::LiquidCrystal(uint8_t rs, uint8_t rw, uint8_t enable,
                             uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3)
{
  init(1, rs, rw, enable, d0, d1, d2, d3, 0, 0, 0, 0);
}

LiquidCrystal::LiquidCrystal(uint8_t rs,  uint8_t enable,
                             uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3)
{
  init(1, rs, 255, enable, d0, d1, d2, d3, 0, 0, 0, 0);
}
```

Zo vinden we in het begin van het cpp bestand de 4 verschillende manieren waarop het object kan aangemaakt worden.
Elk van deze manieren roept uiteindelijk de functie “init” aan, met één of meerdere parameters voorgedefinieerd.

Iets verder vinden we de uitwerking van de init-functie terug met de verschillende parameters. De eerste parameter is de keuze voor 4- of 8-bit mode. Deze wordt bij elk van de 4 aanroepen op 0 of op 1 gezet, afhankelijk van het aantal parameters. In 4-bit mode worden d4,d5,d6 en d7 op 0 gezet bij het aanroepen van de init-functie.

```
void LiquidCrystal::init(uint8_t fourbitmode, uint8_t rs, uint8_t rw, uint8_t enable,
                         uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3,
                         uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7)
```

```
  _rs_pin = rs;
  _rw_pin = rw;
  _enable_pin = enable;

  _data_pins[0] = d0;
  _data_pins[1] = d1;
  _data_pins[2] = d2;
  _data_pins[3] = d3;
  _data_pins[4] = d4;
  _data_pins[5] = d5;
  _data_pins[6] = d6;
  _data_pins[7] = d7;

  if (fourbitmode)
    _displayfunction = LCD_4BITMODE | LCD_1LINE | LCD_5x8DOTS;
  else
    _displayfunction = LCD_8BITMODE | LCD_1LINE | LCD_5x8DOTS;

  begin(16, 1);
```

In de code van de init-functie zien we hoe de verschillende aansluitpinnen worden doorgegeven aan variabelen en hoe de variabele displaymode een logische functie is van de constanten die de instelling van het display regelen.
Daarna wordt de controle doorgegeven aan de begin-functie. In deze functie wordt de initialisatieroutine voor het LCD uitgevoerd.

Maak je eigen bibliotheek

We maken zelf een bibliotheek aan. We houden het simpel en sturen 6 leds aan. De functies die we willen maken zijn

- alleLedsAan
- alleLedsUit
- evenLedsAan
- evenLedsUit
- onevenLedsaan
- onevenLedsUit

De bibliotheek zelf noemen we LedPatronen.

We beginnen met in de map libraries een nieuwe map aan te maken met als naam LedPatronen. Binnen deze map gaan we onze bibliotheekbestanden aanmaken. Dit kan met gelijk welke teksteditor, maar beter niet met de Arduino IDE, omdat deze altijd de extensie .ino toevoegt. Aanraders zijn Notepad++ en Code::Blocks.
Als eerste maken we de ledPatronen.h aan.

ledPatronen.h

```
/*
LedPatronen - voorbeeld van een bibliotheek
Created by F. Sierens.
*/
#ifndef LedPatronen_h
#define LedPatronen_h
#include "Arduino.h"
class LedPatronen
{
public:
LedPatronen(int EerstePin, int ledsAantal);
void alleLedsAan();
void alleLedsUit();
void evenLedsAan();
void evenLedsUit();
void onevenLedsAan();
void onevenLedsUit();
private:
int _EerstePin;
int _ledsAantal;
};
#endif
```

#ifndef LedPatronen_h is een precompiler instructie. Daarmee wordt gekeken of er een LedPatronen_h constante bestaat. Indien niet wordt die aangemaakt en wordt de bibliotheek ingeladen.
Indien de constante al bestaat, betekent dit dat deze bibliotheek reeds werd ingeladen (bijvoorbeeld door gebruik van een andere bibliotheek) en wordt het inladen van de bibliotheek overgeslagen.
Zo'n constructie heet een "include guard" en dient om te vermijden dat eenzelfde bibliotheek meerdere keren zou ingeladen worden.

Verder worden alle Arduino types en constanten ingeladen en vervolgens wordt de klasse gedefinieerd. Eerst worden alle publieke delen aangegeven. Deze zijn zichtbaar vanuit het programma.
Daarna worden nog twee private variabelen gedefinieerd. Deze zijn enkel binnen de bibliotheek zichtbaar en kunnen dus niet vanuit het programma gemanipuleerd worden.

ledPatronen.cpp

```
/*
LedPatronen.cpp - bibliotheek voor ledpatronen.
Created by F. Sierens
*/
#include "Arduino.h"
#include "LedPatronen.h"
LedPatronen::LedPatronen(int EerstePin, int ledsAantal) {
        for (int i = EerstePin ; i < ledsAantal + EerstePin ; i++) {
                pinMode(i, OUTPUT);
        }
        _ledsAantal = ledsAantal;
        _EerstePin = EerstePin;
}

void LedPatronen::alleLedsAan() {
for (int i = _EerstePin ; i < _ledsAantal + _EerstePin ; i++) {
  digitalWrite(i, HIGH);
  }
}

void LedPatronen::alleLedsUit() {
for (int i = _ledsAantal + _EerstePin -1 ; i >= _EerstePin ; i--) {
  digitalWrite(i, LOW);
  }
}

void LedPatronen::evenLedsAan() {
for (int i = _EerstePin ; i < _ledsAantal + _EerstePin ; i++) {
  if ( i % 2 == 0 ) digitalWrite(i, HIGH);
  }
}
```

```
void LedPatronen::evenLedsUit()
{
for (int i = _EerstePin ; i < _ledsAantal + _EerstePin ; i++) {
  if ( i % 2 == 0 ) digitalWrite(i, LOW);
  }
}

void LedPatronen::onevenLedsAan() {
for (int i = _EerstePin ; i < _ledsAantal + _EerstePin ; i++) {
  if ( i % 2 != 0 ) digitalWrite(i, HIGH);
  }
}
void LedPatronen::onevenLedsUit() {
for (int i = _EerstePin ; i < _ledsAantal + _EerstePin ; i++) {
  if ( i % 2 != 0 ) digitalWrite(i, LOW);
  }
}
```