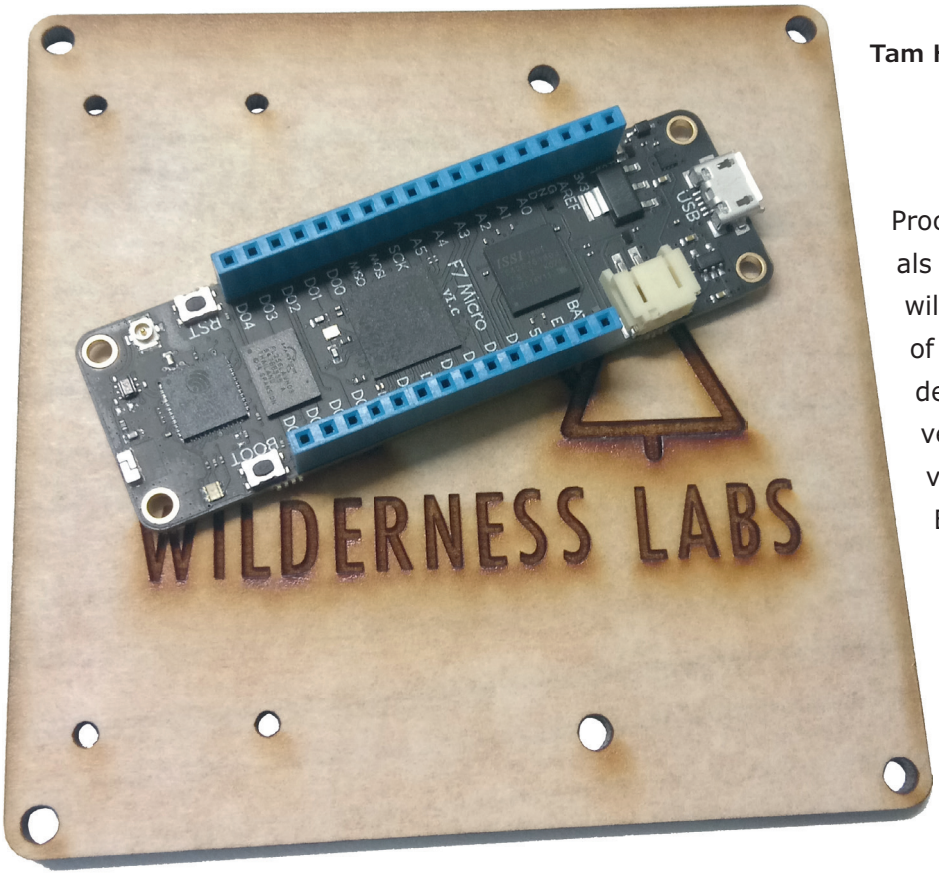
Meadow F7

een board voor .NET-ontwikkelaars



Tam Hanna (Slowakije)

Procescomputers zijn er genoeg, maar als u de software voor uw applicaties wilt ontwikkelen met Visual Basic of C#, blijven er niet veel over. Op de Meadow F7 is een STM32F777 verantwoordelijk voor het uitvoeren van de .NET-runtime, terwijl een ESP32-module verbinding maakt met WLAN-netwerken.

(Foto: Wilderness Labs)

Bryan Costanich van Xamarin, die verantwoordelijk is voor het porten van de .NET-omgeving naar Android en iOS, kocht enige tijd geleden het intellectuele eigendom van het bedrijf van Chris Walker. Het nieuwe product van zijn nieuwe bedrijf Wilderness Labs is de procescomputer Meadow, die op de kopfoto te zien is. Meadow is ontworpen om .NET-ontwikkelaars “eerste klas” toegang te bieden tot het IoT-ecosysteem.
Qua architectuur lijkt het systeem op wat er mogelijk is door een Raspberry Pi of een Orange Pi te combineren met een real-time processorkern. Een STM32F777, geklokt op 216 MHz, is verantwoordelijk voor het draaien van de .NET-runtime, terwijl een ESP32-module verbinding maakt met WLAN-netwerken.

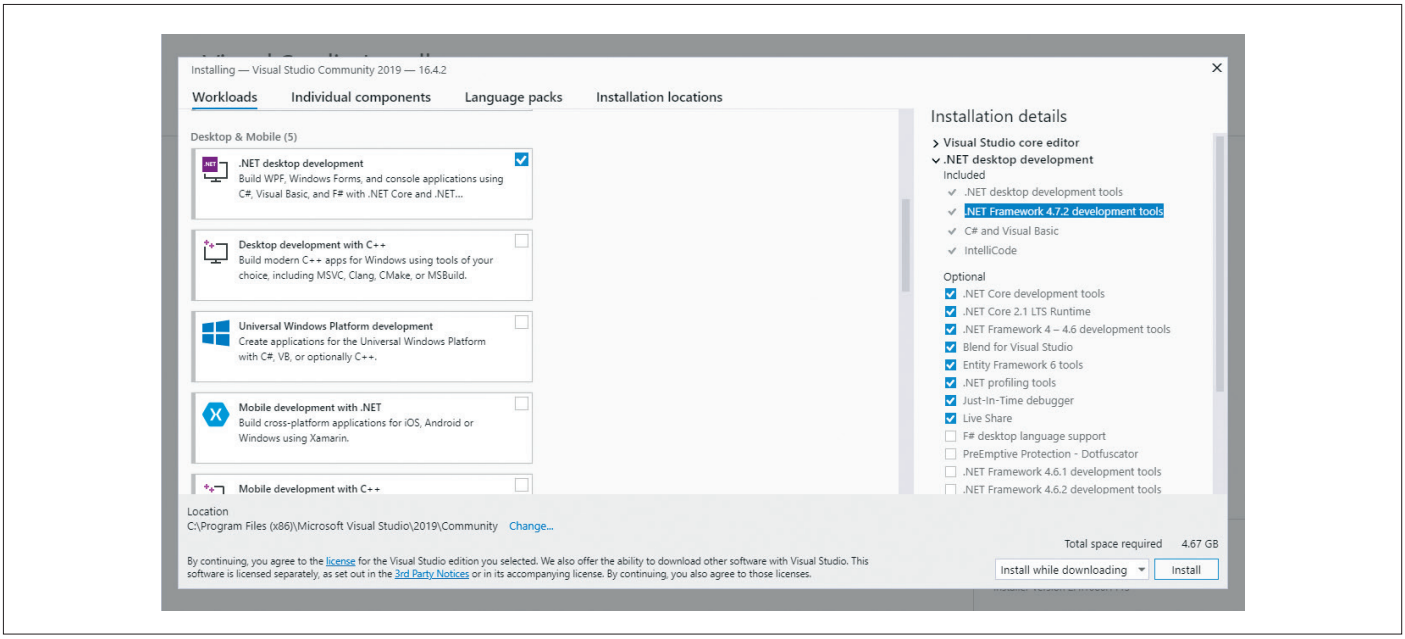
Aan de slag

De Meadow F7-boards worden geleverd met een verouderd besturingssysteem of er is (om kosten te sparen) helemaal geen besturingssysteem geïnstalleerd. We beginnen dus met de installatie van de nieuwste versie met de SGS-bootloader. Ga naar [1] en download het besturingssysteem, dat uit twee bestanden bestaat. Vervolgens hebt u de DFU-utils nodig, die onder [2] op u wachten. Voor een ervaren Elektorlezer zal dat allemaal geen probleem zijn. Volgens de documentatie wordt de bootloader gestart door op de bootknop te drukken. Houd de knop ingedrukt terwijl u het board via een micro-USB-kabel op uw Windows-werkstation aansluit. Om het serienummer van het board te bepalen gebruiken we het list-commando:

```
C:\dfu-util-0.9-win64>dfu-util --list
dfu-util 0.9

. . .
Cannot open DFU device 0483:df11
```

Oeps – wat onder oudere Windows-versies prima werkt, lukt niet onder Windows 10 vanwege de fout die in [3] wordt beschreven. Maar dat is gemakkelijk op te lossen: volg gewoon de instructies om de driver te resetten. Daarna kan het serienummer als volgt worden bepaald:

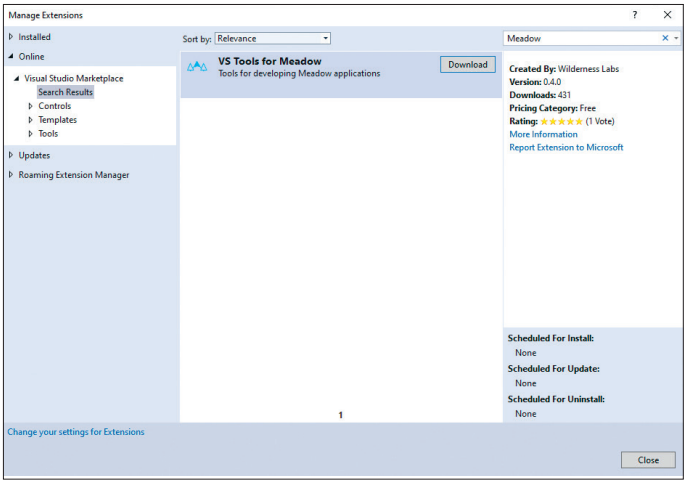
```
C:\dfu-util-0.9-win64>dfu-util --list
dfu-util 0.9
. . .
Found DFU: [0483:df11] ver=2200, devnum=4,
   cfg=1, intf=0, path="5-3", alt=3, name="@
   Device Feature/0xFFFF0000/01*004 e",
   serial="346B38733536"
```



Figuur 1: Zonder .NET-framework 4.7.2-ontwikkeltools is de Meadow niet prettig om mee te werken.

Succesvol gedetecteerde Meadow-boards verschijnen in Windows-apparaatbeheer niet als één, maar als vier eindpunten. Maar voor ons is alleen het serienummer van belang, dat we in de volgende stap gebruiken om de kernel en runtime te installeren. Natuurlijk moet u de hier getoonde commando's aanpassen aan uw eigen situatie! Zorg ervoor dat u de hexadecimale adressen correct intypt en plaats de twee bestanden `Meadow.OS_Kernel.bin` en `Meadow.OS_Runtime.bin` in de juiste map:

```
C:\dfu-util-0.9-win64>   dfu-util -a 0 -S
   346B38733536 -D Meadow.OS_Kernel.bin -s 0x08000000
dfu-util 0.9
C:\dfu-util-0.9-win64>   dfu-util -a 0 -S
   346B38733536 -D Meadow.OS_Runtime.bin -s
   0x08040000
dfu-util 0.9
```



Figuur 2: Visual Studio haalt desgewenst automatisch extensies van het internet.

Om zeker te zijn van een goede start, drukt u op de RST-knop: de RGB-LED begint dan vrij zenuwachtig te knipperen. Als ontwikkelomgeving op de desktop heb ik gebruik gemaakt van Visual Studio in de gratis community-versie 2019.8. U moet daarbij de in **figuur 1** getoonde componenten selecteren en downloaden in de installatiewizard die beschikbaar is in het Start-menu onder *Visual Studio Installer.*

Start dan Visual Studio zoals gewoonlijk. U kunt de nieuwe startwizard overslaan door op de link *Doorgaan zonder code* te klikken, als u de IDE start zonder een bestaand project te openen. Het bedrijf Wilderness Labs levert de eigenlijke SDK in de vorm van een Visual Studio-plugin. Klik op *Extensies beheren* om de Plugin Wizard te laden. Ga dan, zoals in **figuur 2,** naar de rubriek *Online* en zoek naar de string *Meadow*.

Na het installeren van de plugin start u Visual Studio opnieuw op. U ziet dan een nieuw sjabloon genaamd *Meadow Application*, dat als de basis vormt voor eigen experimenten. Maak een nieuw programma met de naam "*ElektorSample*" om te genieten van de code van het voorbeeld met de knipperende LED dat bij de bootloader wordt geleverd. De code in het bestand `MeadowApp.cs` zou voor zichzelf moeten spreken.

Ontwikkelaars die overstappen van Arduino zullen er even aan moeten wennen dat het sjabloon geen lus bevat. Het gegeven voorbeeld initialiseert gewoon de twee methoden van de constructor; de eindeloze lus bevindt zich in `BlinkLeds`:

```
public MeadowApp()
{
    ConfigurePorts();
    BlinkLeds();
}
```

Start- en pingtest

Als een single-board-computer zware real-time taken moet uitvoeren, moet het besturingssysteem "gegarandeerde" responstijden bieden. Systemen die gebaseerd zijn op managed

languages zoals Java of C# hebben daar over het algemeen moeite mee. Dat ligt niet alleen aan de inefficiënte runtime: als de *garbage collector* (automatische geheugenopschoning) draait, dan kan er geen andere taak worden uitgevoerd. Ons kleine testprogramma werkt daarom bijna volledig zonder dynamische geheugentoewijzing. Maar als de controletaak veel geheugen toewijst en vrijgeeft, is er een grotere kans dat de garbage collector gaat lopen.

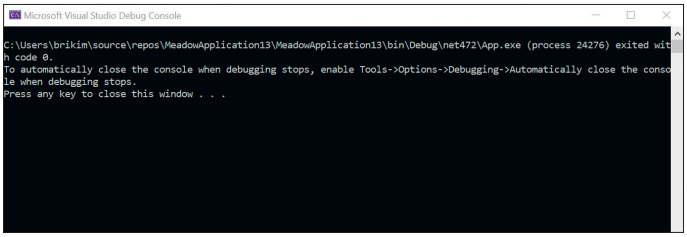
Om het gedrag van de F7 te testen, laten we hem (zoals altijd) een karakteristieke golfvorm genereren. Hiervoor passen we zowel `ConfigurePorts` als `BlinkLeds` als volgt aan:

```
public class MeadowApp : App<F7Micro, MeadowApp> {
        IDigitalOutputPort myOut;

        public MeadowApp()
. . .
        public void ConfigurePorts() {
            Console.WriteLine(“Creating Outputs...”);
            myOut= Device.
   CreateDigitalOutputPort(Device.Pins.D05);
        }

        public void BlinkLeds() {
            var state = false;

            while (true) {
                myOut.State = true;
                myOut.State = false;
                myOut.State = true;
                myOut.State = false;
            }
        }
    }
```

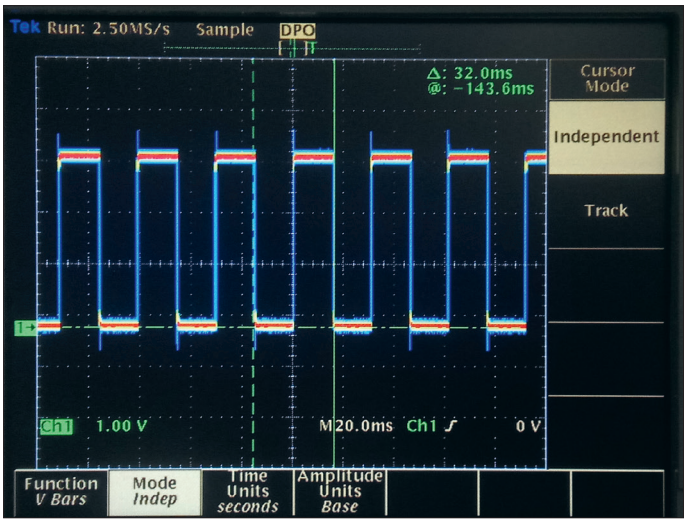
De Meadow-software maakt gebruik van abstractieklassen voor de interactie met de periferie. Onze digitale poort wordt bijvoorbeeld gecreëerd door een interface van het type `IDigitalOutputPort`. Als we een uitbreiding aanbieden die ook GPIO-pennen gebruikt, dan kunnen we (met de juiste driver) de code heen en weer schuiven tussen het “normale” en het nieuwe randapparaat
We gebruiken voor de test een digitale oscilloscoop van Tektronix, een TDS754D met een “gehackte” bandbreedte van 1 GHz. Zo’n apparaat is tweedehands vrij goedkoop verkrijgbaar in de VS, zelfs uitgerust met een optionele LCD.
Omdat de Meadow F7 er vanuit Windows uitziet als “slechts” één COM-poort, heeft Visual Studio een beetje hulp nodig om ermee te kunnen werken. Klik om te beginnen op

View Other Windows Meadow (of *Ctrl+Shift+M*)

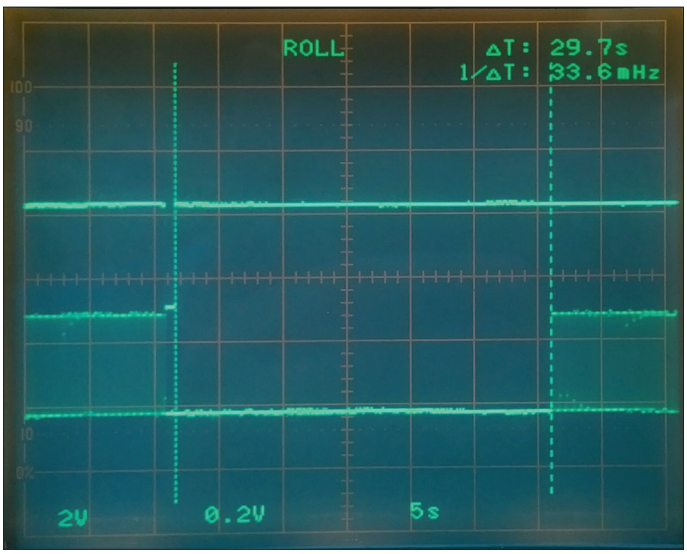
om het venster voor het selecteren van apparaten, *“Meadow Device Explorer”*, te activeren Kies uw Meadow en geef dan het commando voor een debugging-run. Bij het starten van het programma verschijnt ook het venster in **figuur 3**, dat u moet sluiten door op de Entertoets te drukken.
Sluit de oscilloscoop na de succesvolle deployment aan op pin



Figuur 3: Dit venster zit in de weg en moet verdwijnen.



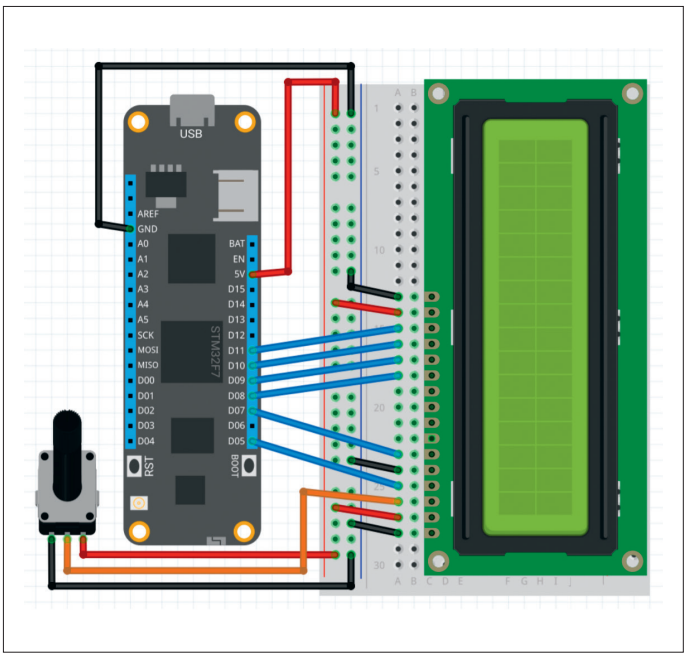
Figuur 4: Een periode van 32 ms: dat is bepaald niet snel!



Figuur 5: Opstarttijd van 30 s.

D05 en geniet van het oscillogram in **figuur 4**. Het is normaal dat de runtime enkele datumklassefouten genereert tijdens de uitvoering van het programma.

Wie bedreven is in het programmeren in assembler, krijgt van dit beeld misschien hoofdpijn, maar het verschil in de golfdalen is maar erg klein. Dat betekent dat de meeste tijd toch wel wordt besteed aan het schakelen van de pennen. Door optimalisaties is dat nog te verbeteren.



Figuur 6: Hier wordt het nogal krap op het breadboard (bron: Wilderness Labs [5]).

De reset-lijn is niet alleen verbonden met de drukknop, maar is ook beschikbaar op een pen van de lange connectorstrip (eerste pen naast de spanningsomvormer). Als u daar een oscilloscoop in roll-modus aansluit en geduld hebt, kunt u een opstarttijd van bijna 30 s zien (**figuur 5**)!
Voor de eerste experimenten is het ideaal om een display aan de F7 te koppelen, zoals in **figuur 6.** Het hierboven genoemde ontwerpparadigma met de abstractieklassen geldt ook voor het `CharacterDisplay`. Om te beginnen maken we een extra instantie van de klasse `MeadowApp`, die gaat zorgen voor de communicatie met het display.

```
public class MeadowApp : App<F7Micro, MeadowApp>  {
        CharacterDisplay myDisplay;
```

De opbouw van hardware drivers verloopt onder Meadow altijd volgens hetzelfde schema. De constructor krijgt eerst een verwijzing naar een device-object. De driver krijgt opdracht om de uitvoer-hardware van de controller te gebruiken. Net als bij de Kinect-SDK van Microsoft is dit een maatregel om de flexibiliteit te vergroten (in theorie zouden we ook een GPIO-extender kunnen implementeren).
De volgende stap is een groep van benoemde parameters die de gebruikte uitgangspennen beschrijven. Die krijgen waarden uit de `Enum Device.Pins`. Die bevat een apart bitveld voor elk hardware-randapparaat van de STM32-processor, wat de controle vergemakkelijkt:

```
public void ConfigurePorts()
{
    Console.WriteLine("Creating Outputs...");
    myDisplay = new CharacterDisplay(
        Device,
        pinRS: Device.Pins.D05,
        pinE: Device.Pins.D07,
        pinD4: Device.Pins.D08,
        pinD5: Device.Pins.D09,
        pinD6: Device.Pins.D10,
        pinD7: Device.Pins.D11,
        rows: 4, columns: 20
        );
}
```

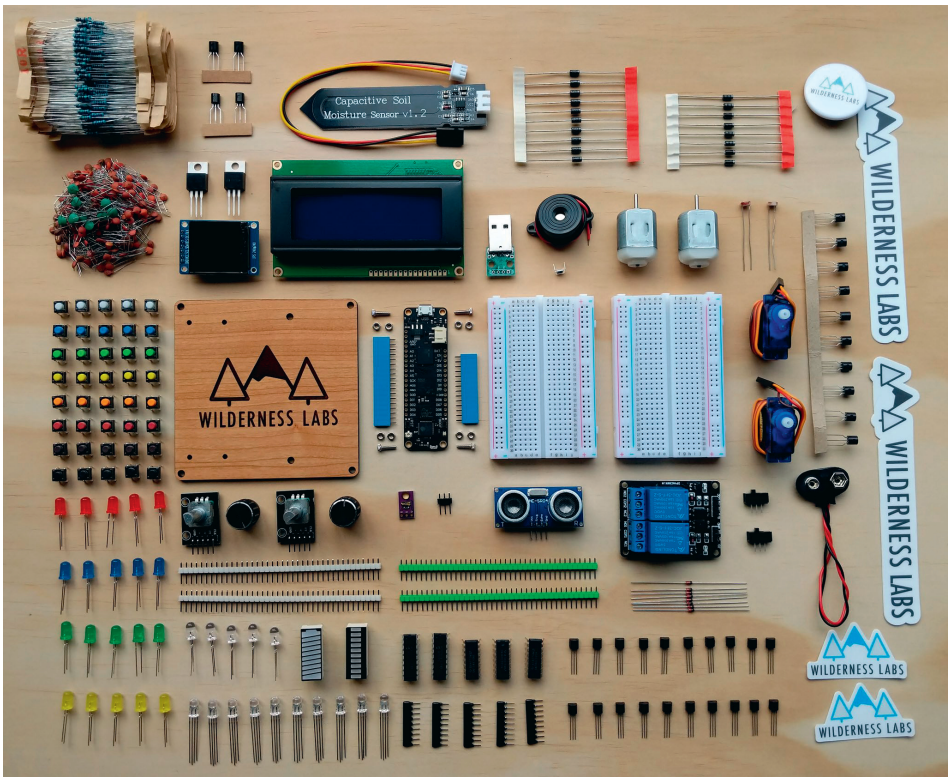
Nu blijkt dat Visual Studio de verwijzing naar de klasse `CharacterDisplay` niet kan oplossen. Dat komt door de modulaire manier van aanleveren van het framework. Klik met de rechter muisknop op het project in de Explorer aan de rechterkant en kies de NuGet package manager. Zoek dan naar de string *Meadow*Character* in de rubriek *"Zoeken"*. Het sterretje is een wildcard die staat voor een willekeurige tekst. Als het goed is, vindt u dan het pakket `Meadow.Foundation.Displays.LCD.CharacterDisplay`, dat u kunt installeren als een normaal NuGet pakket.
Dan ontbreekt alleen nog de eigenlijke uitvoerfunctie, die tekst naar het scherm stuurt. Wilderness Labs maakt hier gebruik van de infrastructuur van het .NET-framework. De syntaxis van de methode `writeLine` zou u bekend moeten voorkomen. De extra numerieke parameter bepaalt op welke regel van het display de string moet worden weergegeven. Als u de waarde "1" geeft, schrijft u in de tweede regel van bovenaf:

```
public void BlinkLeds()
{
    var state = false;

    while (true)
    {
        myDisplay.WriteLine("Hallo Elektor", 1);
        System.Threading.Thread.Sleep(1000);
    }
}
```

Download het programma opnieuw naar de Meadow en geniet van de weergave. Als u de module uit de Hack Kit gebruikt, moet u letten op de contrastinstelling, anders kan het gebeuren dat de tekens er wel zijn, maar dat u ze niet ziet. Het valt ook op dat het scherm relatief traag werkt, ook hier is dus op het moment dat we dit schrijven nog ruimte voor optimalisatie.

Vol verwachting...

Microsofts Gadgeteer-platform, dat alweer is opgedoekt, wilde ontwikkelaars met weinig hardware-ervaring in staat stellen om snel en eenvoudig prototypes te bouwen. Voor dat doel werd een aantal geprefabriceerde modules aangeboden, die via flatcable werden verbonden met de basisboards.
Dit idee leeft nog voort bij Wilderness Labs. Het pakket "Meadow F7 Micro Development Kit w/Hack Kit Pro" (**figuur 7**), dat al op de website [6] kan worden besteld, bevat naast de Meadow F7 twee insteekkaarten, een hoogwaardig 4x20-alphanumeriek LCD, een heleboel actieve en passieve componenten, actuatoren en sensoren en als hoogtepunt een echt houten ontwikkelplatform.
Daarnaast is er een heel uitgebreide bibliotheek van drivers, die



Figuur 7: De Development-Hack-Kit bevat veel materiaal om uit te proberen (bron: Wilderness Labs [6]).

door het team rond Costanich beschikbaar wordt gesteld. Bij het ter perse gaan is de driver voor het (hoogwaardige) kleuren-LCD nog niet klaar voor gebruik, maar de kit zelf wordt ook pas in maart vrijgegeven. Kijk in plaats daarvan vol verwachting naar andere sensoren die in de Wilderness Labs Hardware List zijn opgenomen [4].

Een oud Engels spreekwoord zegt dat bedelaars niet kieskeurig moeten zijn. Als u .NET wilt gebruiken in de embedded wereld, heeft u op dit moment alleen de keuze tussen de verouderde NetDuino of de Meadow. Maar wie tevreden is met de IO-performance van de Meadow, vindt hier een zeer uitgebreide driverbibliotheek die snelle realisatie van prototypes mogelijk maakt. De Gadgeteer-belofte leeft dus voort! ⏮

(191190-04)

Weblinks

[1] Meadow F7: http://beta-developer.wildernesslabs.co/Meadow/Getting_Started/Deploying_Meadow/

[2] DFU-utils: http://dfu-util.sourceforge.net/releases/dfu-util-0.9-win64.zip

[3] Windows-Bug: www.hanselman.com/blog/HowToFixDfuutilSTMWinUSBZadigBootloadersAndOtherFirmwareFlashingIssuesOnWindows.aspx

[4] Periferie: http://developer.wildernesslabs.co/Meadow/Meadow.Foundation/Peripherals/

[5] Character Display: http://beta-developer.wildernesslabs.co/docs/api/Meadow.Foundation/Meadow.Foundation.Displays.Lcd.CharacterDisplay.html

[6] Meadow Kit /w Hack: https://store.wildernesslabs.co/collections/frontpage/products/meadow-f7-micro-development-board-w-hack-kit-pro