ELV-Modulsystem + Arduino – die Verschmelzung zweier Welten

Die Verschmelzung zweier Welten war immer schon begeisternd. Sie ist ein Nährboden für neue Ideen und eine treibende Kraft in Bezug auf kreatives Denken. Durch die Kombination vordefinierter Elektronikbausteine lassen sich neue Geräte- bzw. Produktideen auf schnellstem Wege realisieren. Man nehme ELV Basismodul A, füge ELV Applikationsmodul B hinzu, kombiniere dies mit der Arduino IDE und versorge alles mit intelligentem Programmcode. Et voilà, ein lang gehegter Gerätewunsch erfüllt sich. Durch die Arduino IDE, die ELV Basis- und Applikationsmodule sowie die online verfügbaren Programmbeispiele und -bibliotheken wird man in die Lage versetzt, den Entwicklungsaufwand eines Geräts deutlich zu vereinfachen und zugehörige Entwicklungszeiten drastisch zu verkürzen. So sind der Vorstellungskraft keine Grenzen gesetzt.



Motivation

Mit der Integration des ELV-Modulsystems in die Arduino-Umgebung werden vielfältige Möglichkeiten geschaffen, um das ELV-Modulsystem auf persönliche Bedürfnisse zuzuschneiden. Dieser Artikel soll eine Installationsanleitung sein sowie einige Codebeispiele und Denkanstöße geben, um eine Applikation unter Zuhilfenahme der Arduino IDE zu entwerfen.

Anleitung

Um die <u>ELV LoRaWAN®-Base</u> mit der <u>Arduino IDE</u> verwenden zu können, muss zunächst das STM32duino-Paket in der Arduino-Entwicklungsumgebung installiert werden. Hierzu navigiert man im Menüband der Arduino-IDE-Applikation zunächst auf *File* und dann zu *Preferences...*, um in das Einstellungsfenster zu gelangen (Bild 1).

Innerhalb der Einstellungen befindet sich am unteren Ende des Fensters ein Feld mit der Beschriftung Additional Board Manager URLs. An dieser Stelle muss die URL eingetragen werden, damit das STM32duino-Paket installiert werden kann (Bild 2). Bei Bestätigung mit OK wird die Option der Boards in den Arduino-Board-Manager eingetragen (Bild 3).

Bei den Board-Definitionen handelt es sich um eine Auflistung verschiedener Microcontroller-Platinen und deren Eigenschaften. Einige Hersteller haben z. B. aus Platzgründen nicht alle Anschlusspins des verwendeten Microcontrollers nach außen zugänglich gemacht oder über bestimmte Beschaltungselektronik einige Funktionen von Pins bevorzugt bzw. unterbunden. Mit der Auflistung dieser Elemente wird somit gewährleistet, dass jeder Eintrag auch mit dem verwendeten Code funktionieren kann.

Um die Board-Definitionen tatsächlich herunterzuladen, muss in der Seitenleiste der Entwicklungsumgebung auf das Platinensymbol geklickt werden, um den Board-Manager zu öffnen. Hierzu wird in der Suchleiste der Suchbegriff "STM32 MCU based boards" eingegeben. Beim entsprechenden Eintrag kann im Drop-down-Menü der Versionen mindestens die Version 2.8.1 ausgewählt werden.

	duino IDE 2.3.2		
File Edit Sketch	Tools Help		
New Sketch	Strg + N	•	
New Cloud Ske	tch Alt + Strg + N		
Open	Strg + O		
Sketchbook		Image: A start and a start	
Examples		 here, to run on 	ce:
Close	Strg + W		
Save	Strg + S		
Save As	Strg + Umschalttaste + S		
Preferences	Strg + Komma	here, to run rep	eatedly:
Advanced		•	
Quit	Strg + Q		
Output			

Bild 1: Das Einstellungsmenü ist im Menüband unter "File" zu finden.

Hinweis:

Um die Bibliothek verwenden zu können wird zusätzlich noch das Programm <u>STM32CubeProgrammer</u> benötigt.



Bild 2: Im Feld "Additional boards manager URLs" muss die Adresse eingegeben werden.

	Clone this wiki locally
Add STM32 boards support to Arduino	https://github.com/stm32duino/Ardu
This is the needed step to get STM32 targets added to Arduino. So carefully follow the following steps.	
Installing STM32 Cores	
1- Launch Arduino.cc IDE. Click on "File" menu and then "Preferences".	
Preferences X	
Settings Network	
Sketchbook location:	
sketchbook Drowse	
Editor language: System Default v (requires restart of Arduino)	
Editor font size: 12	
Interface scale: Automatic 100 0 % (requires restart of Arduino)	
Theme: Default theme 🗸 (requires restart of Arduino)	
Show verbose output during: 🗹 compilation 📝 upload	
Compiler warnings: All 🗸	
Display line numbers Enable Code Folding	
✓ Verify code after upload Use external editor	
Check for updates on startup	
Use accessibility features	
Additional Doards Manager URLs: https://github.com/stm32duino/DoardManagerFiles/raw/dev/package_stmicroelectronics_index.json	
More preferences can be edited directly in the file	
C:\STM32\prduino-\arduino-1.0.13\portable\preferences.txt	
(edit only when Arduino is not running)	
OK Cancel	

Bild 3: Die Board-Manager-Liste unter <u>dieser URL-Adresse</u> wird von STMicroelectronics stets aktuell gehalten.

Mit einem Klick auf *Install* werden die Board-Definitionen in die Entwicklungsumgebung übernommen (Bild 4).

Nach der erfolgreichen Installation kann die ELV LoRaWAN[®]-Base über das Menü ausgewählt werden. Dies geschieht wie in Bild 5 dargestellt über Tools \rightarrow Board \rightarrow STM32 MCU based boards \rightarrow ELV Modular System boards.

Wenn das System ausgewählt wurde, kann unter Tools \rightarrow Board part number die ELV LoRaWAN®-Base ausgewählt werden. Spätestens jetzt sollte sie auch per USB-C Kabel angeschlossen werden. Nun kann im Windows-Gerätemanager der COM-Port der Base ermittelt werden, wie in Bild 6 zu sehen ist. Dieser Port wird nachfolgend in der Arduino IDE angegeben.

Zuletzt muss unter Upload method der Eintrag SMT32CubeProgrammer (Serial) with Bootloader angewählt werden.

🔤 sket	ch_may2a Arduino IDE 2.3.2	
File E	dit Sketch Tools Help	
	Select Board	
	BOARDS MANAGER	sketch_may2a.ino
	S1M32	// put setup() {
힘	Typo: All 🗸	3 4 }
<u>tik</u>	STM32 MCU based boards by STMicroelectronics	5 6 void loop() {
0	Boards included in this package: Nucleo-144, Nucleo-64, Nucleo-32, Discovery, Eval, STM32MP1 series More info	<pre>7 // put your main code here, to run repeatedly: 8 9 }</pre>
Q	2.7.1 V INSTALL	10
	STM8 MCU based boards by STMicroelectronics	Output
	Boards included in this package: Nucleo_85208RD, Nucleo_8L152R8 More Info	LiquidCrystal@1.0.7 LiquidCrystal@1.0.7 wird installiert LiquidCrystal@1.0.7 installiert
	1.0.0 ¥ INSTALL	SD@1.2.4 wird heruntergeladen SD@1.2.4
8		SD@1.2.4 wird installiert SD@1.2.4 installiert

Bild 4: Nach erfolgreicher Installation sind die STM32-Board-Definitionen im Board-Manager der Arduino IDE zu finden.

Edit Skotch	Tools Help		
Euit Sketch	Auto Format		Nucleo-144
∕ → ⊵	Auto Format Strg + 1		Nucleo-64
sketch m	Manage Libraries Stra + Limechalttaste + L		Nucleo-32
3Ketch_III	Sarial Manitar Stra + Umschalttasta + M		Discovery
1	Serial Monitor Sug + Orischartaste + M		Eval
- 3	Senar Piotter		STM32MP1 series coprocessor
4	Firmware Updater		Generic STM32C0 series
5	Upload SSL Root Certificates		Generic STM32F0 series
M 6	Board 🕨	Boards Manager Strg + Umschalttaste + B	Generic STM32F1 series
/	Port 🕨	Arduino AVR Boards	Generic STM32F2 series
	Get Board Info	STM32 MCII based boards	Generic STM32F3 series
10	Rurn Rootloader		Generic STM32F4 series
) 10	buin bootoader]	Generic STM32F7 series
			Generic STM32G0 series
			Generic STM32G4 series
			Generic STM32H5 series
			Generic STM32H7 Series
			Generic STM32L0 series
			Generic STM32L1 series
			Generic STM32L4 series
			Generic STM32L5 series
			Generic STM32U5 series
			Generic STM32WB series
			Generic STM32WBA series
			Generic STM32WL series
			3D printer boards
			Blues Wireless boards
			Elecgator boards
			Electronic speed controllers
			Garatronic-McHobby
			Generic Flight Controllers
			IoT continuum boards
			LoRa boards
			Midatronics boards
3)			ELV Modular System boards
			SparkEup Boards

Bild 5: Nach erfolgreicher Installation kann nun der Eintrag der "ELV-modular-system"-Boards im Board-Menü gefunden werden.

Steht ein STLink-Programmer zur Verfügung, kann der Eintrag *SMT32CubeProgrammer* (*SWD*) with *Bootloader* verwendet werden. Hierfür muss dann SWDIO vom STLink mit Pin 18 und SWDCLK mit Pin 17 sowie GND mit Pin 16 und VDD mit Pin 15 verbunden werden.

Die korrekten Einstellungen zum Bespielen der ELV LoRaWAN[®]-Base sind in Bild 7 zusammenfassend dargestellt.

Nach dieser Einrichtung kann der erste Programmcode geschrieben werden, um die Installation zu prüfen. Hierzu kann das Blink.ino-Beispiel mit minimaler Änderung verwendet werden.

Um einen leichten Einstieg zu vollziehen, bietet es sich an, zunächst eine LED zum Blinken zu bringen. Das zugehörige Programm ist verhältnismäßig einfach. Die LED ist an einem bestimmten Pin angeschlossen. Dieser Pin muss zunächst als Ausgang initialisiert werden. Danach muss der Pin zwischen dem Logiklevel "HIGH" und "LOW" wechseln. Damit der Einschaltvorgang für das menschliche Auge sichtbar wird, muss die Einschaltdauer auf einen wahrnehmbaren Zeitwert eingestellt werden.

Das Beispielprogramm ist im Menü unter File \rightarrow Examples \rightarrow 01. Basics \rightarrow Blink zu finden. Die Anpassungen im Programmcode sind im Bild Codeblock 1 zu finden.



Bild 6: Der COM-Port kann im Windows-Gerätemanager nachgeschaut werden.

Alle Programmcodes finden Sie auch als Source-Datei im ELVshop bei diesem Fachbeitrag unter Downloads.

Auto Format	Strg + T	
Archive Sketch		
Manage Libraries	Strg + Umschalttaste + I	
Serial Monitor	Strg + Umschalttaste + M	
Serial Plotter		
Firmware Updater		
Upload SSL Root Certificates		
Board: "ELV Modular System boards"		Þ
Port: "COM5"		Þ
Get Board Info		
Debug symbols and core logs: "None"		Þ
Optimize: "Smallest (-Os default)"		۲
Board part number: "ELV-LW-Base ELV_BM_TRX1"		۲
C Runtime Library: "Newlib Nano (default)"		۲
Upload method: "STM32CubeProgrammer (Serial) with Bootloader"		۲
U(S)ART support: "Enabled (generic 'Serial')"		٨
Burn Bootloader		

Codeblock 1

// die setup-Funktion wird einmal zu Beginn ausgeführt
void setup() {
// Die Board-LED der ELV-LoRaWAN® Base wird initialisiert (Grüne LED der
DUO-LED DS2
<pre>pinMode(LED_BUILTIN, OUTPUT);</pre>
}
// Die loop()-Funktion wird immer wieder ausgeführt
<pre>void loop() {</pre>
<pre>digitalWrite(LED_BUILTIN, LOW); // LED einschalten (LOW ist das Logiklevel)</pre>
delay(1000);
<pre>digitalWrite(LED_BUILTIN, HIGH); // LED wieder ausschalten, in dem die</pre>
Spannung auf HIGH gezogen wird
<pre>delay(1000); // 1000 Millisekunden warten</pre>
}



Bild 8: Die Schaltflächen zum Verifizieren, Übertragen und Debuggen der Firmware.

Bild 7: Die korrekten Programmier-

einstellungen

Ob der Programmcode Syntaxfehler hat, kann mithilfe des Verify-Buttons anhand des Häkchen-Symbols geprüft werden (Bild 8). Dieser gibt bei erfolgreicher Kompilierung über die Konsole im unteren Fensterbereich Auskunft darüber, wie viel Speicherplatz das aktuelle Programm benötigt (Bild 9).

Flashen

Bevor der Programmcode auf die LoRaWAN®-Base übertragen werden kann, muss diese in den Bootloader-Modus gebracht werden. Dies geschieht, indem man beim Zuführen der Spannungsversorgung die Taste S1 gedrückt hält. Wenn der Bootloader aktiv ist, blinkt die rote LED an DS2 für eine Sekunde auf. Um den Flash-Vorgang zu starten, muss auf den Button mit dem Pfeil nach rechts (Bild 8) gedrückt werden. Nach dem erfolgreichen Flashen des Geräts beginnt die ELV LoRaWAN®-Base mit der Ausführung des Programms (Bild 9). Die grüne LED an DS2 sollte nun im Sekundentakt blinken gemäß den im Codeblock 1 definierten Vorgaben.

Verwendung von Spezialfunktionen

Eine Besonderheit des auf der ELV LoRaWAN®-Base verwendeten Funkmoduls ist der sehr energiesparende Low-Power-Modus. Um die darin enthaltenen Funktionen verwenden zu können, muss die Entwicklungsumgebung um eine Bibliothek erweitert werden. Eine solche Bibliothek stellt eine Sammlung von Programmcodes dar, die über programminterne Funktionen abrufbar ist. Die Bibliothek, die die Low-Power-Funktionen verwendbar macht, heißt STM32LowPower. Hier ist in der Beschreibung gesondert darauf zu achten, dass eine Abhängigkeit zu einer anderen Bibliothek besteht. Um die Low-Power-Funktionen verwenden zu können, muss die Real-Time-Clock des Controllers verwendet werden, da die anderen Timer im Low-Power-Modus abgeschaltet werden. Die Bibliothek, die diese Funktionalität bereitstellt, heißt <u>STM32RTC</u>. Das Installieren einer Bibliothek ist immer identisch und wird im Folgenden beschrieben.

Auf der Hauptseite der Bibliothek kann im rechten oberen Bereich der grüne "<> Code"-Button (siehe Bild 10) geklickt werden, wodurch sich ein Drop-down-Menü öffnet, das die Option Download ZIP verfügbar macht (Bild 11).

Sketch uses 12832 bytes (9%) of program storage space. Maximum is 129024 bytes. Global variables use 1228 bytes (2%) of dynamic memory, leaving 47924 bytes for local variables. Maximum is 49152 bytes. 	
Serial Port COM5 is successfully opened. Port configuration: parity = even, baudrate = 115200, data-bit = 8, IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Sketch uses 12832 bytes (9%) of program storage space. Maximum is 129024 bytes. Global variables use 1228 bytes (2%) of dynamic memory, leaving 47924 bytes for local variables. Maximum is 49152 bytes.
<pre>Serial Port COM5 is successfully opened. Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: parity = even, baudrate = 115200, data-bit = 0, Port configuration: provide protocol version: 3.3 Device name : STM32WLXX Plan size : 256 KBVtes (default) Port corf configuration: port configuration: port configuration: provide prov</pre>	STM32CubeProgrammer v2.14.0
<pre>Set in For Constrainty Open. Dort configuration: party = even, baudrate = 115200, data-bit = 8, stop-bit = 1.0, flow-control = off Activating device: OK Board : Chip ID: 0x407 Boattoader protocol version: 3.3 Device name : STH32WLxx Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin file : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Frasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNWING Program Address: : 0x8000800 Start operation achieved successfully</pre>	Senial Port COME is successfully opened
<pre>For team guideline party feach, badwater = instead, data for = of, Activating device: 0K Board : Chip ID: 0x497 BootLoader protocol version: 3.3 Device name : STM32WLxx Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.66 KB Address : 0x080000000 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully</pre>	Series configuration: a_{1} and a_{2} and a_{2} and a_{2} and a_{2} and a_{2}
<pre>Activating device: 0K Board : Chip ID: 0x497 BootLoader protocol version: 3.3 Device name : STM32MLxx Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x0800800 Start operation achieved successfully</pre>	$ \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot$
Activiting terrer on Board : Chip DD: 8x497 BootLoader protocol version: 3.3 Device name : STN32WLxx Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x080000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800	Activating device: OK
Chip ID: 0x497 BotLoader protocol version: 3.3 Device name : STM32WLXX Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x00000800 Erasing memory corresponding to segment 0: Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x0000800 Start operation achieved successfully	Board :
BootLoader protocol version: 3.3 Device name : STM32WLxx Flash size : 256 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x08000800 Start operation achieved successfully	Chin TD: Ax497
Device name : STM32WLxx Flash size : 256 KBytes (default) Device type : MCU Revision 10 : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	BootLoader protocol version: 3.3
Filash size : 255 KBytes (default) Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x08000800 Start operation achieved successfully	
Device type : MCU Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Flash size : 256 KBytes (default)
Revision ID : Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x080000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Device type : MCU
Device CPU : Cortex-M4 Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Revision ID :
Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Device CPU : Cortex-M4
Memory Programming Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	
Opening and parsing file: Blink.ino.bin File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Memory Programming
File : Blink.ino.bin Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Opening and parsing file: Blink.ino.bin
Size : 12.86 KB Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	File : Blink.ino.bin
Address : 0x08000800 Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Size : 12.86 KB
Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Address : 0x08000800
Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : : 0x8000800 Start operation achieved successfully	
Erasing memory corresponding to segment 0: Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	
Erasing internal memory sectors [1 7] Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : : 0x8000800 Start operation achieved successfully	Erasing memory corresponding to segment 0:
Download in Progress: File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Erasing internal memory sectors [1 7]
File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Download in Progress:
File download complete Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	
Time elapsed during download operation: 00:00:02.014 RUNNING Program Address: : 0x8000800 Start operation achieved successfully	File developed complete
RUNNING Program Address: : 0x8000800 Start operation achieved successfully	File download complete
RUNNING Program Address: : 0x8000800 Start operation achieved successfully	Time etapsed during downtoad operation. 00.00.02.014
Address: : 0x8000800 Start operation achieved successfully	RUNNITING Program
Start operation achieved successfully	Address: : av8000800
	Start operation achieved successfully

Bild 9: Während der Programmübertragung wird man von der Arduino IDE darüber informiert, welcher Prozess in Bearbeitung ist.

💡 main 👻 🤔 1 Branch 🚫 12 Tags		Q Go to file	t	Add file 👻	<> Co	de
S fpistm feat(WBA): add STM32WBAxx support	~		eb649e0 -	3 months ago	🕚 84 Comm	nits
github/workflows	ci: split wor	kflow			10 months	s ag
axamples	chare(exam	ples): configure RTC alarm in B	IN mode to	o wakeup	5 months	s ag
extras	ci: split wor	kflow			10 months	s ag
src src	feat(WBA):	add STM32WBAxx support			3 month	s ag
README.md	chore: enha	ance shutdown mode			5 month	s ag
🗅 keywords.txt	attachInter	ruptWakeup() add parameter t	o define Lo	owPower	4 years	s ag
library.json	chore: bum	p library version to 1.2.5			5 months	s ag
library.properties	chore: bum	ap library version to 1.2.5			5 month	s ag
II README Security					0	
STM32LowPower Arduino library to support STM32 Low Power.						

Bild 10: Alle Bibliotheken, die für das Programmieren der ELV LoRaWAN®-Base notwendig sind, sind im github von <u>STMicroelectronics</u> zu finden.

G GitHub - stm32duino/STM32RT ×	+					~ - 6
⇒ C @	O A https://github.com/stm32duine/STM32RTC				目 茲 ☆	◎ ± ●
Product ~ Solutions ~ 0	Open Source 👻 Pricing				Q. Search or jump to	7 Sign in Sign
stm32duino / STM32RTC ↔ Code ⊙ Issues ۩ Pull re	이 Public quests 1 ⓒ Actions 田 Projects ⓒ Security 너스 Insight	is			다 Notifications 및 F	Fork 45 î Star 117 +
	P main → P 1 Branch 🛇 16 Tags		Q Go to file	<> Code +	About	
	Francois RAMU and fpistm new sketch to demonstr	rate the RIC mode c	De Clone	0	Arduino RTC library for STM32.	
	.github/workflows	ci(config): limit numl	HTTPS GitHub CLI		Code of conduct	
	examples r	new sketch to demo	https://github.com/stm32duino/STM3	2RTC.git	Security policy Arctivity	
	extras of	ci(config): limit numl	Clone using the web URL.		E Custom properties	
	src 3	stm32RTC change th	[슈] Open with GitHub Desktop		✿ 117 stars	
	🗅 .editorconfig	chore: add git attrib	Download ZIP		♥ 45 forks	
	🗋 .gitattributes	chore: add git attribu	ites and editor config	last year	Report repository	
	C README.md s	stm32RTC can give t	he hour Format 24 or 12 hours	3 months ago	Releases 16	
	🗅 keywords.txt 🕴	feat: add SubSecond	interrupt underflow api for STM32WLxx	7 months ago	STM32RTC library v1.4.0 (Latest)	
	🗋 library.json	chore: update library	version to 1.4.0.	7 months ago	on Oct 12, 2023	
	library.properties	chore: update library	version to 1.4.0	7 months ago	+ 15 releases	
	TREADME I Code of conduct A Security				Packages No packages published	
	STM32RTC				Contributors 8	
	A RTC library for STM32.					
	Requirement				Languager	

Bild 11: Um externe Programmbibliotheken verwenden zu können, muss als erster Schritt der Programmcode als ZIP-Archiv von github heruntergeladen werden.

Um die ausgewählte Bibliothek innerhalb der Arduino IDE verwenden zu können, muss im Menü die Option Sketch \rightarrow Include Library \rightarrow Add .ZIP Library gewählt werden. Dadurch öffnet sich ein Dateiauswahldialog, in dem die heruntergeladenen ZIP-Archive ausgewählt



Bild 12: Der zweite Schritt zur Verwendung externer Bibliotheken ist das Einfügen in die Arduino-Entwicklungsumgebung.

werden können (Bild 12). Sobald die STM32RTC und die STM32Low-Power hinzugefügt worden sind, kann ein erneutes Programmbeispiel verwendet werden.

Das Programmbeispiel Codeblock 2 ist unter File \rightarrow Examples \rightarrow STM32duino Low Power \rightarrow TimedWakeup zu finden. Die erforderlichen Anpassungen sind ähnlich zu denen im Beispiel Codeblock 1.

Nach dem Kompilieren und Hochladen des Programms ändert sich nichts am sichtbaren Verhalten des Geräts. Betrachtet man allerdings den Stromverbrauch, ist erkennbar, dass in den Deep-Sleep-Phasen nur noch ein Strom von 2,5 µA fließt. Im Vergleich zu den 3,4 mA im Normalmodus ist dies eine deutliche Verbesserung in etwa um den Faktor 1000.

Codeblock 2

/* TimedWakeup
This sketch demonstrates the usage of Internal Interrupts to wakeup a chip in deep sleep mode.
In this sketch, the internal RTC will wake up the processor every second.
This example code is in the public domain.
*/ // Die Verwendete Bibliothek muss über eine include-Dirketive zum Code hinzugefügt werden
<pre>#include "STM32LowPower.h"</pre>
<pre>void setup() { pinMode(LED_BUILTIN, OUTPUT); // Die LowPower-Funktionen müssen einmal initialisiert werden LowPower.begin(); }</pre>
<pre>void loop() { digitalWrite(LED_BUILTIN, HIGH); LowPower.deepSleep(1000); digitalWrite(LED_BUILTIN, LOW); LowPower.deepSleep(1000); }</pre>

Codeblock 3



LoRaWAN®

Das wichtigste Herausstellungsmerkmal der ELV LoRaWAN®-Base ist die LoRaWAN®-Funktionalität. Um diese verwenden zu können, wird die Bibliothek <u>STM32LoRaWAN</u> benötigt. Diese ist ebenfalls von der STM32RTC-Bibliothek abhängig, die bereits im vorherigen Schritt installiert wurde.

Nach der Installation der STM32LoRaWAN®-Bibliothek kann das Programmbeispiel Codeblock 3 verwendet werden, um die LoRaWAN®-Funktion überprüfen zu können. Hierbei ist darauf zu achten, anstelle der Nullen die Keys der ELV LoRaWAN®-Base einzusetzen.

Die ELV LoRaWAN®-Base kann nun wie im Fachbeitrag zur ELV-LW-Base beschrieben mithilfe der gleichen Keys registriert werden. Um die Daten innerhalb des TTN-Netzwerks auslesen zu können, muss dort in der Oberfläche ein Payload-Parser definiert werden. Der Code für das Umwandeln von Hexadezimalwerten zurück in ASCII-Zeichen ist in Codeblock 4 dargestellt. Dieser Code wird im TTN als Uplink-Payload-Parser entweder für die gesamte Applikation oder das einzelne Endgerät eingefügt.

Codeblock 4

```
function Decoder(bytes, port) {
    // Decode plain text; for testing onLy
    return {
    myTestValue: String.fromCharCode.apply(null, bytes)
    };
}
```

Verwendung mit Applikationsmodulen des ELV-Modulsystems

Nachfolgend wird das Vorgehen zur Verwendung verschiedener ELV-Applikationsmodule erklärt, beispielhaft anhand des ELV-Applikationsmoduls Luftdruck ELV-AM-AP. An dieser Stelle erfolgt allerdings keine Schritt-für-Schritt-Anleitung, sondern lediglich ein Überblick, um ein grundlegendes Gefühl für das notwendige Vorgehen zu vermitteln. Dazu wird im Library-Manger der Arduino-Entwicklungsumgebung nach der Teilenummer des auf dem ELV-Applikationsmodul verbauten Sensors gesucht. Für das Beispiel des ELV-AM-AP ist dies der Sensor BMP581 von Bosch Sensortec. Um den Library-Manager zu verwenden, muss in der Seitenleiste der Arduino IDE auf das Bibliotheksmanager-Icon geklickt werden. In der Suchleiste kann anschließend nach der Bezeichnung des Sensors gesucht werden, wie in Bild 13 zu sehen.

Nach der Installation dieser Bibliothek stehen Programmbeispiele des BMP581 zur weiteren Verfügung. Unter File \rightarrow Example \rightarrow Sparkfun BMP581 Arduino Library \rightarrow Example1_BasicReadingsl2C kann man sich einen Überblick darüber verschaffen, wie diese Bibliothek arbeitet. Von besonderer Bedeutung sind hierbei die #include-Direktiven, Initialisierungen sowie das Auslesen der Messwerte. Verdeutlicht wird dies in Codeblock 5.

Alle Codeblocks dieses Betrags können zu einem neuen Programm zusammengeführt werden. Dies wird in Codeblock 6 veranschaulicht.

Zudem ergeben sich durch die Integration der ELV LoRaWAN®-Base einige weitere Möglichkeiten, um das ELV-Modulsystem in spannenden Projekten einzusetzen. Es können mehrere ELV-Applikationsmodule zusammengesteckt verwendet werden, wodurch der Kreativität unserer Leser kaum Grenzen gesetzt sind.

Ein interessantes Beispiel ist die Realisation einer eigenen Schaltaktorik, die man mithilfe eines Step-up-Spannungswandlers, des ELV Bausatzes <u>Relais-Schaltmodul ELV-RSM1</u> und eines zum Verbraucher passenden Relais erstellt.

Neben der Einbindung von ELV-Applikationsmodulen können auch viele individuelle Schaltungen mithilfe der ELV Prototypenadapter entwickelt werden. Hierzu wird es in den kommenden ELVjournalen weitere spannende Projekte geben.

BMP5	31	
Туре:	All	~
Topic:	All	~

SparkFun BMP581 Arduino Library by SparkFun...

1.0.1 installed

A library to drive the Bosch BMP581 pressure sensor. The BMP581 provides high resolution pressure an... More info

1.0.1 Y REMOVE

Codeblock 5

```
#include <Wire.h>
#include "SparkFun_BMP581_Arduino_Library.h"
// Ein Sensor-Objekt erstellen
BMP581 pressureSensor;
// I2C Adresse aussuchen. Das ELV-AM-AP hat 0x46 als Standard-Adresse
//uint8_t i2cAddress = BMP581_I2C_ADDRESS_DEFAULT; //
uint8_t i2cAddress = BMP581_I2C_ADDRESS_SECONDARY; // 0x46
void setup()
    // Serielle Schnittstelle initialisieren
    Serial.begin(115200);
    Serial.println("BMP581 Example1 begin!");
    // I2C Bibliothek initialisieren
   Wire.begin();
    // Sensor testen
    while(pressureSensor.beginI2C(i2cAddress) != BMP5_OK)
    {
        // Falls nicht verbunden, Ausgabe
        Serial.println("Error: BMP581 not connected, check wiring and I2C
address!"):
        // Etwas abwarten und dann noch einmal probieren
        delay(1000);
   }
    Serial.println("BMP581 connected!");
}
void loop()
    // Sensordaten auslesen
    bmp5 sensor data data = {0,0};
    int8_t err = pressureSensor.getSensorData(&data);
       Fehler auslesen, ob Auslesen erfolgreich war
    if(err == BMP5_OK)
        // Auslesen erfolgreich, Ausgabe der Messwerte
        Serial.print("Temperature (C): ");
        Serial.print(data.temperature);
Serial.print("\t\t");
        Serial.print("Pressure (Pa): ");
        Serial.println(data.pressure);
    else
        // Auslesen nicht erfolgreich, Fehler ausgeben
        Serial.print("Error getting data from sensor! Error code: ");
        Serial.println(err);
   }
    // Nur jede Sekunde ausgeben.
    delay(1000);
}
```

www.elvjournal.com

Codeblock 6

```
#include <STM32LoRaWAN.h>
#include <STM32LowPower.h>
#include <Wire.h>
#include "SparkFun BMP581 Arduino Library.h"
STM32LoRaWAN modem;
// Create a new sensor object
BMP581 pressureSensor;
// I2C address selection
uint8_t i2cAddress = BMP581_I2C_ADDRESS_SECONDARY; // 0x46
static const unsigned long TX INTERVAL = 60000; /* ms */
unsigned long last_tx = 0;
unsigned int counter = 0;
/* Get the rtc object */
STM32RTC& rtc = STM32RTC::getInstance();
void setup() {
  LowPower.begin();
  Serial.begin(115200);
  Serial.println("Start");
  modem.begin(EU868);
  // Configure join method by (un)commenting the right method
  // call, and fill in credentials in that method call.
  bool connected = modem.joinOTAA(/* AppEui */ "0000000000000", /* AppKey */
"0000000000000000000000", /* DevEui */ "000000000000");
  //bool connected = modem.joinABP(/* DevAddr */ "00000000", /* NwkSKey */
"0000000000000000000000000000", /* AppSKey */
"0000000000000000000000000000000"):
  if (connected) {
   Serial.println("Joined");
  } else {
    Serial.println("Join failed");
    while (true) /* infinite loop */
     ;
  }
  Wire.begin();
  /* set the calendar */
  rtc.begin(true);
  rtc.setTime(00, 00, 00);
  rtc.setDate(00, 00, 00);
  while(pressureSensor.beginI2C(i2cAddress) != BMP5_OK)
    {
        // Not connected, inform user
        Serial.println("Error: BMP581 not connected, check wiring and I2C
address!");
        // Wait a bit to see if connection is established
       delay(1000);
   }
    Serial.println("BMP581 connected!");
}
void send_packet() {
  counter++:
  bmp5 sensor data data = \{0,0\};
  int8_t err = pressureSensor.getSensorData(&data);
  Serial.println(data.temperature);
  Serial.println(data.pressure);
  char payload[50] = { 0 }; /* packet to be sent */
  /* prepare the Tx packet : get date and format string */
  String temp = String(data.temperature);
  String pres = String(data.pressure);
  sprintf(payload, "#%d: %02d/%02d/%04d - %02d:%02d:%02d: %s C %s Pa",
          counter.
          rtc.getMonth(), rtc.getDay(), 2000 + rtc.getYear(),
          rtc.getHours(), rtc.getMinutes(), rtc.getSeconds(), temp.c_str(),
pres.c_str());
```

```
modem.setPort(10);
 modem.beginPacket();
 modem.write(payload, strlen(payload));
 if (modem.endPacket() == (int)strlen(payload)) {
   Serial.println("Sent packet");
 } else {
   Serial.println("Failed to send packet");
 3
 if (modem.available()) {
   Serial.print("Received packet on port ");
   Serial.print(modem.getDownlinkPort());
   Serial.print(":");
   while (modem.available()) {
     uint8_t b = modem.read();
     Serial.print(" ");
     Serial.print(b >> 4, HEX);
     Serial.print(b & 0xF, HEX);
   Serial.println();
 }
 Serial.flush();
3
void loop() {
   send_packet();
    LowPower.deepSleep(TX_INTERVAL);
    Serial.begin(115200);
   Wire.begin();
}
```