



AVR[®] Home Automation Kit

Getting Started with the AVR[®] Home Automation Kit

Introduction

Author: Johan Lofstad, Microchip Technology Inc.

This is an introductory guide for the AVR[®] IoT Home Automation Kit. It describes how to connect the kit to the cloud, how to download and modify the firmware, and how to add functionality. When the kit is up and running, a stepper motor driver is introduced. General stepper motor theory is covered, how do they work, and how can you control them. The final step covers how to connect the motor to the kit, writing drivers for it and adding a custom control panel on the cloud, which controls the motor remotely.

Table of Contents

Introduction.....	1
1. Getting Started.....	3
1.1. Connecting the Board to the Host PC.....	3
1.2. The AVR-IoT Webpage.....	4
1.3. Connecting the Board to Wi-Fi® Networks.....	5
1.4. Visualizing Cloud Data in Real Time.....	7
1.5. Configuring Other Settings	9
1.6. Onboard Programmer/Debugger Interface.....	11
1.7. Generating the Demo using Atmel START.....	11
1.8. Exporting AVR-IoT WG Sensor Node START Project to Atmel Studio.....	15
2. Firmware.....	16
2.1. Implementing: Echo Single Digit.....	17
2.2. The Scheduler.....	19
3. Stepper Motors: The Basics.....	22
3.1. Stepper Motor Basics.....	22
3.2. Controlling a Stepper Motor.....	24
4. Using the Stepper 2 Click.....	27
4.1. Adding the PWM with Atmel START.....	27
4.2. Selecting the Correct PINMUX.....	30
4.3. Writing the Driver.....	32
4.4. Connecting to the Cloud.....	33
5. Revision History.....	36
The Microchip Website.....	37
Product Change Notification Service.....	37
Customer Support.....	37
Microchip Devices Code Protection Feature.....	37
Legal Notice.....	37
Trademarks.....	38
Quality Management System.....	38
Worldwide Sales and Service.....	39

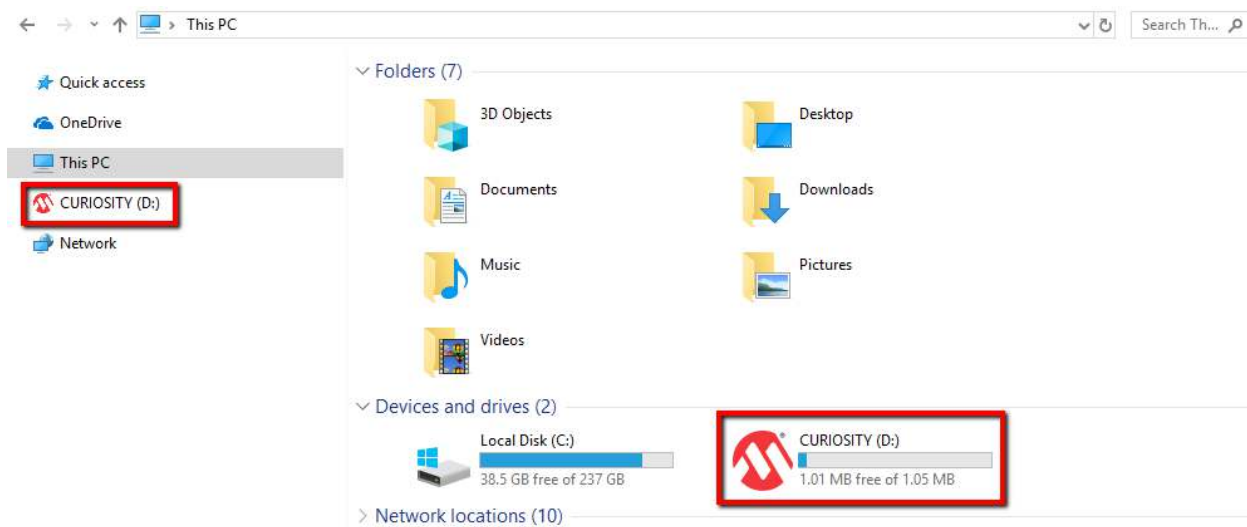
1. Getting Started

1.1 Connecting the Board to the Host PC

The AVR-IoT Development Board can be connected to a computer using a standard Micro-USB cable. Once plugged in, the LED array at the top right-hand corner of the board should flash twice in the following order: Blue→Green→Yellow→Red. When the board is not connected to Wi-Fi[®], the red LED will light up. The board will appear as a Removable Storage Device on the host PC, as shown in the figure below. Double-click the **CURIOSITY** drive to open it and get started.

Note: All procedures are the same for Windows[®], Mac OS[®], and Linux[®] environments.

Figure 1-1. Curiosity Board as Removable Storage



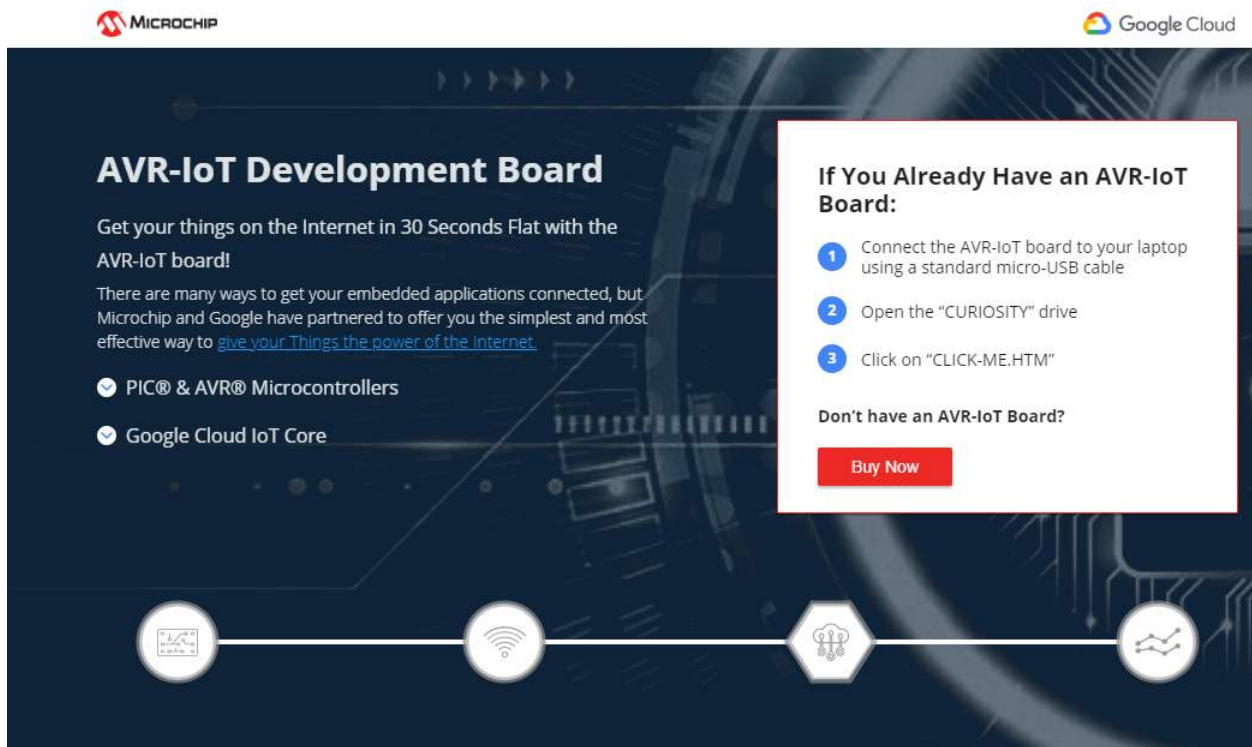
The CURIOSITY drive should contain the following five files:

- CLICK-ME.HTM - redirects the user to the AVR-IoT web demo application
- KIT-INFO.HTM- redirects the user to a site containing information and resources about the board
- KIT-INFO.TXT - a text file with details about the PKOB nano firmware and the board's serial number
- PUBKEY.TXT - a text file containing the public key used for data encryption
- STATUS.TXT - a text file containing the status condition of the board

Double-click on the **CLICK-ME.HTM** file to enter the dedicated webpage to access the Google Cloud sandbox account.

1.2 The AVR-IoT Webpage

Figure 1-2. AVR-IoT Webpage



Simplicity

The AVR microcontroller architecture has been widely recognized as one of the most effective choices for embedded control design. Makers and developers worldwide learned to appreciate its power and simplicity through the [Arduino platform](#). ([Learn more about Arduino](#))

Using the power of the [Atmel START](#) tool, a free, professional, *rapid development tool (code generator)*, you can now add Google Cloud connectivity to new and existing projects with a click of your mouse.



Figure 1-2 shows an image of the AVR-IoT WG webpage. This page displays the sensor data and allows the user to regenerate the Wi-Fi credentials as a file labeled WIFI.CFG. This can be loaded onto the board, acting as a storage device to reconfigure access point parameters.

The status markers in the middle of the page, as shown in the figure below, indicate the progress of the system setup. These markers will light up once each stage has completed successfully.

Figure 1-3. Webpage Status Indicators



The leftmost marker indicates if the board is connected to the host PC. Next to this, the Wi-Fi marker lights up once the board is connected to a Wi-Fi network, and the blue LED will turn on to indicate the board connection state. To the right of the Wi-Fi marker, the Google Cloud Message Queuing Telemetry Transport (MQTT) marker is found,

indicating the status of the TCP socket connection and MQTT connection to Google Cloud. The corresponding green LED will turn on to indicate the board connection state. Finally, the rightmost marker lights up, signifying that data is streaming from the board to the server. This is shown by the blinking of the yellow LED on the board for each successful MQTT publication of data.

1.3 Connecting the Board to Wi-Fi[®] Networks

1.3.1 Via AVR-IoT Webpage

There are several ways to connect the AVR-IoT Development Board to the Internet. The easiest way is through the AVR-IoT webpage (www.avr-iot.com/). The lower left-hand corner of the site will show a wireless network connection window where the user can choose to connect to an open (no password required) network or enter the credentials for a password protected (WPA/WPA2/WEP) Wi-Fi[®] network. The figure below shows how to enter the Wi-Fi credentials on the website.



Important: The Wi-Fi network SSID and password are limited to 19 characters. Avoid using quotation marks, names, or phrases that begin or end with spaces. The AVR-IoT Development Board supports only 2.4 GHz networks inline, thus using mobile hotspots to connect the board to the Internet is recommended.

Figure 1-4. Entering Wi-Fi[®] Credentials in AVR-IoT Webpage

Wireless Network Login

MCHP-IOT

Your WiFi information is not transmitted anywhere—the config file is generated in your browser.

Network Type

Open

WPA/WPA2

WEP

Show password

Download Configuration

Once the required details are entered, click the **Download Configuration** button. This will download the WIFI.CFG (text) file to the host PC. From the WIFI.CFG's download location, drag and drop the file to the CURIOSITY drive to update the Wi-Fi credentials of the board. The blue LED will light up once a successful connection to the Wi-Fi Access Point is made.



Important: Any information entered in the SSID and password fields is not transmitted over the web or to the Microchip or Google servers. Instead, the information is used locally (within the browser) to generate the WIFI.CFG file.

1.3.2 Via Command Line Interface (CLI)

Another way of connecting to the Wi-Fi[®] is through the Serial Command Line Interface (CLI). This interface can be accessed through any serial terminal application. Using the UART settings defined in the [1.5.2 Serial USB Interface](#) section, the user can reconfigure the board to a Wi-Fi network by entering the **wifi** command. [Figure 1-5](#) and [Figure 1-6](#) show examples of trying to connect to open, or secured networks, respectively. For more details on the **wifi** command and its parameters, refer to the [1.5.2 Serial USB Interface](#) section.

Figure 1-5. Wi-Fi[®] Configuration via Serial Command Line (Open Network)

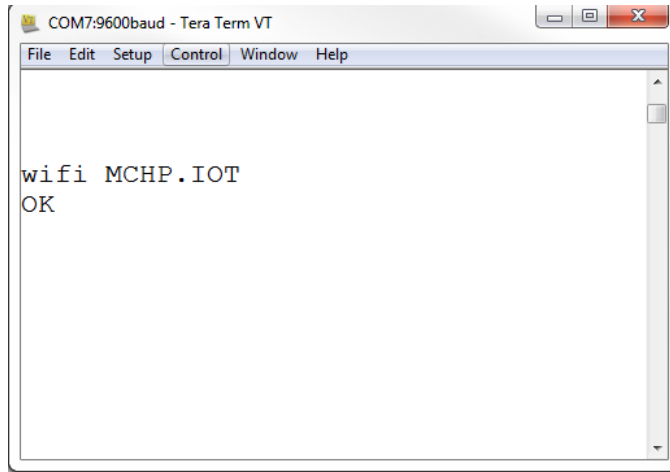
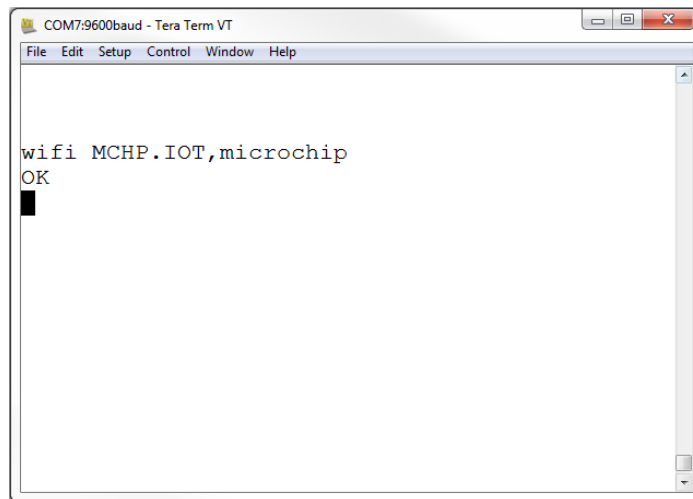


Figure 1-6. Wi-Fi[®] Configuration via Serial Command Line (Secured Network)



1.3.3 Via Soft AP

The last method to connect to the Wi-Fi is through the advanced Software Access Point (Soft AP) mode, which is a feature of the WINC module on board. This method is ideal if the user is only using a mobile device, such as a mobile phone or tablet, instead of a laptop or PC. The Soft AP mode can be entered by pressing and holding the SW0 push button for most of the start-up time between initial power-up LED cycling. When the Soft AP mode has been successfully entered, the board can be detected as a Wi-Fi access point named MCHP.IOT.ACCESSPOINT. The blue LED will start blinking when Soft AP is available. Using a mobile device such as a mobile phone or tablet, connect to the MCHP.IOT.ACCESSPOINT hotspot. It will redirect to a sign-in page where the user can enter the SSID and password of the network to which the board will connect. The Device Name will not be considered, and the authorization type will always be WPA/WPA2 (2). Once these details are entered, click the **Connect** button to connect the board to the network. Refer to the figure below to see how the sign-in page will look.

Figure 1-7. Connecting via Soft AP



1.4 Visualizing Cloud Data in Real Time

Out of the box, all AVR-IoT WG Development Boards are pre-registered to Microchip's Google Cloud sandbox account. This account is set up for demonstration purposes only. All data gathered by the sensors of the AVR-IoT WG Development Boards are published on the Microchip sandbox account and can be identified by the following details:

Table 1-1. Project Details

Project ID	avr-iot
Region	us-central1

There is no permanent storage or collection of data published by the boards connected to the Microchip sandbox account. The full storage catalog of the Google Cloud features, such as data storage/retention, can be available to the user with the use of the board once removed from the demo environment, and the associated Device ID/Public Key has been migrated to a private account.

1.4.1 Publishing Data to the Google Cloud

An MQTT PUBLISH packet is always sent to the MQTT broker using a specific topic. The AVR-IoT Development Board publishes messages using the topic '/devices/{deviceId}/events' in communication to the Google Cloud. The messages published on this topic contain the real-time data obtained from the on-board light and temperature sensors. It does not perform any averaging of data, which is done to allow instantly visible changes on the webpage. The frequency of sending a PUBLISH packet can be decided by the user application. The application is written such that the sensor data is published to the Cloud every one second.

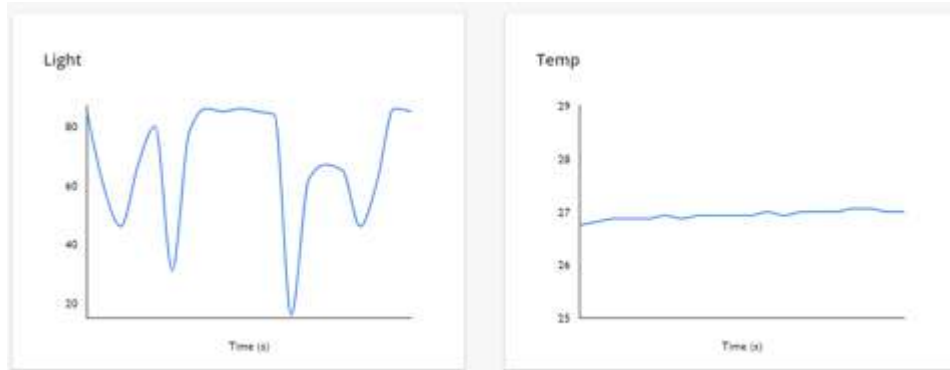
1.4.1.1 Viewing the Published Messages

Once the board is connected to a Wi-Fi access point and has established a socket connection to the Cloud, the AVR-IoT webpage will show a real-time graph of the data captured from the on-board light and temperature sensors. Data are sent as an MQTT PUBLISH packet from the board to the Cloud through a JSON object.

The ASCII string is formatted as follows:

{'Light': XXX, 'Temp': YYY }, where XXX and YYY are numerical values expressed in decimal notation. The yellow LED on the board is turned on for 250 ms every one second to indicate that the board is publishing data.

Figure 1-8. Real-Time Data on the AVR-IoT Webpage



1.4.2 Subscribing to Topics

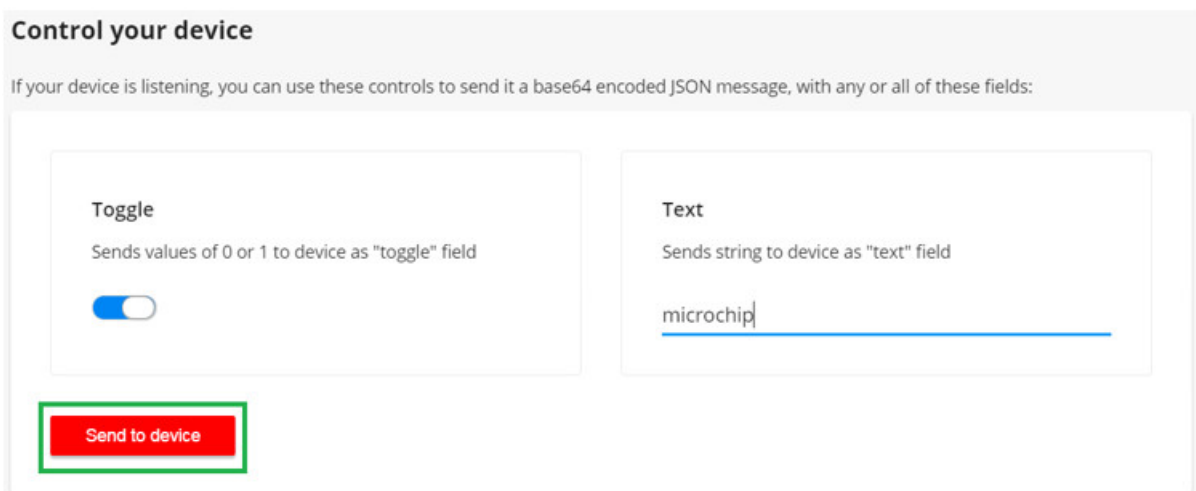
In addition to publishing its data, the AVR[®]-IoT Development Boards are capable of subscribing to a topic. When subscribing to a topic, it will receive data from the Google Cloud whenever data with that topic is published to the broker server. Subscribing to topics is desired when the receiver is interested in the information sent to the broker by other connected client devices publishing data using the subscribed topic. After sending a SUBSCRIBE packet, all the messages published on the specific topic of subscription are received by the board. As of now, the board subscribes to the '/devices/{deviceId}/config' topic. This is the only topic provided by Google Cloud for subscribing using the MQTT connection.

1.4.3 Sending the Messages

The www.avr-iot.com webpage URL displays a section “Control your device” below the Light and Temperature graphs. The **Toggle** button is used to send the switch value to the AVR-IoT board.

Similarly, the “Text” section can be used to send a text string to the board. The toggle and text fields can be modified individually or at the same time. These values are only published over the '/devices/{deviceId}/config' topic upon pressing the **Send to device** button. Since the board subscribes to this topic by default, all the published messages are received by the board.

Figure 1-9. Sending Messages on the Subscribed Topic



Control your device

If your device is listening, you can use these controls to send it a base64 encoded JSON message, with any or all of these fields:

Toggle
Sends values of 0 or 1 to device as "toggle" field

Text
Sends string to device as "text" field

microchip

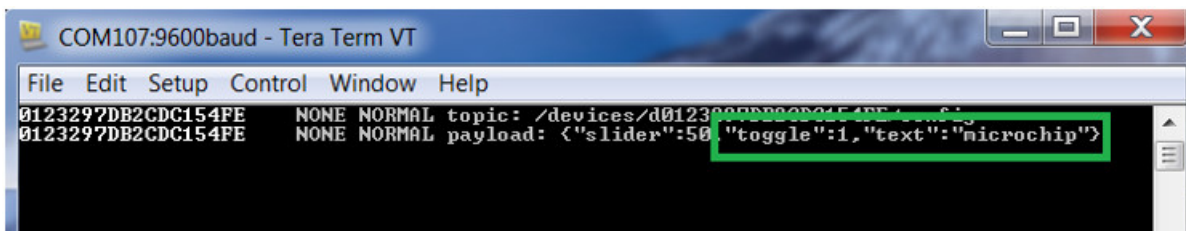
Send to device

1.4.4 Viewing Messages Received on Subscribed Topic

The toggle switch value corresponds to a short forced ON/OFF state to the yellow LED on the AVR-IoT Development Board. The LED will stay on/off for a short time depending on the position of the toggle switch before it resumes normal behavior, blinking to indicate the transmission sensor data through PUBLISH packets.

The message typed in the text field is transmitted in the form of a string to the board. In addition to the yellow LED behavior, the toggle and text field values can be viewed on a Serial Terminal application (such as Tera Term, Realterm, PuTTY, etc.)

Figure 1-10. Viewing Messages on a Serial Terminal



There is no permanent storage or collection of the data published by the boards connected through the Microchip sandbox account. The full storage features available by Google Cloud are available to the user after the board has been removed from the demo environment (Microchip sandbox) and migrated to a private account.

1.5 Configuring Other Settings

While the AVR®-IoT Development Board comes fully programmed and provisioned right out of the box, the user can still control aspects of the application firmware behavior through the USB interface. There are three methods to do this:

1. Hex file (reprogram) or WIFI.CFG (reconfigure credentials) drag and drop using the mass storage feature.
2. Commands through the Serial Command Line Interface (CLI), or using MPLAB X IDE.
3. The on-board programmer/debugger PKOB nano.

1.5.1 Mass Storage Drag and Drop

One way to program the embedded device is to drag and drop a .hex file into the CURIOSITY drive. The C compiler toolchain generates a .hex file for each project it builds. This .hex file contains the code of the project. The Nano Embedded Debugger (PKOB nano) also provides access to a serial port interface (serial to USB bridge). This facilitates the user to drag and drop a modified .hex file, which contains the firmware updates. This feature does not require any USB driver to be installed and works in all major OS environments. Alternative application 'example.hex' files for the board firmware will be available for selection from the downloads section at the bottom of the AVR-IoT webpage as soon as they become available.

1.5.2 Serial USB Interface

The Wi-Fi Access Point credentials can be reconfigured through a Serial Command Line Interface (CLI) on the AVR-IoT Development Boards. This interface may also be used to provide application diagnostic information. To access this interface, use any preferred serial terminal application (that is, Tera Term, CoolTerm, and PuTTY) and open the serial port labeled Curiosity Virtual COM port, with the following settings:

Table 1-2. Serial USB Interface Settings

Baud rate	9600
Data	8-bit
Parity bit	None
Stop bit	1 bit
Flow control	None
Local echo	ON

Transmit protocol	CR+LF (Carriage Return + Line Feed)
-------------------	-------------------------------------

Note: For users of the Windows[®] environment, the USB serial interface requires the installation of a USB serial port driver, included in the installation of the MPLAB X IDE.

The user can control the board by typing the command keywords, listed in the table below:

Table 1-3. Serial Command Line Commands

Command	Arguments	Description
reset	-	Reset the settings on the device
device	-	Print the unique device ID of the board
key	-	Print the public key of the board
reconnect	-	Re-establish connection to the Cloud
version	-	Print the firmware version of the serial port user interface
cli_version	-	Print the command line interface firmware version of the serial port user interface
wifi	<Network SSID>, <Password>, <Security Option*>	Enter the Wi-Fi network authentication details
debug	<Debug Options**>	Print debug messages to see status of board operation

*- Authorization Type options are available by typing one of the following three numbers to determine the network security option used:

1. Open - Password and Security option parameters are not required.
2. WPA/WPA2 - Security Option Parameter not required.
3. WEP - Network Name, Password, and Security Option (3) Parameter are required when connecting to a WEP network. For example, 'wifi MCHP.IOT,microchip,3'.

** - Debug Severity options are available to `debug_printer()` by using a severity number from zero to four:

0. Normal – Only SEVERITY_NONE messages are printed. In the application, this is a printout of the payload received over the Subscribed Topic.

1. Warning – SEVERITY_WARNING and under messages are printed*.

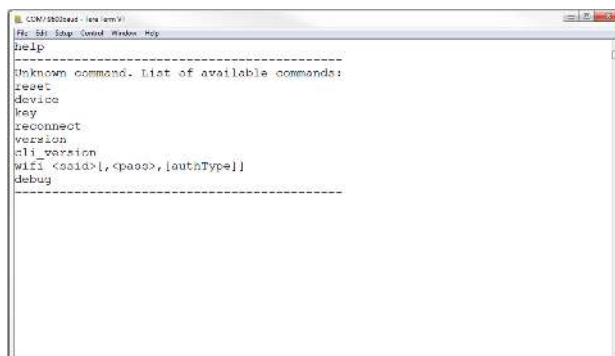
2. Notice – SEVERITY_NOTICE and under messages are printed*.

3. Info - SEVERITY_INFO and under messages are printed*.

4. Debug - SEVERITY_DEBUG and under messages are printed. There are multiple ERROR handling assist messages in place using this SEVERITY.

Note: *There are NO messages of this type in the IoT WG Development Board application.

Figure 1-11. Serial Command Line Interface



```
COM1552Serial - (avr-gcc)
File Edit Debug Console Window Help
help
-----
Unknown command. List of available commands:
reset
device
key
reconnect
version
cli_version
wifi <ssid>[, <pass>, [authType]]
debug
-----
```

1.6 Onboard Programmer/Debugger Interface

For users familiar with the MPLAB X IDE, the AVR-IoT boards can be programmed and/or debugged directly via these IDEs standard operations. The AVR-IoT Development Boards are automatically detected by the MPLAB X, enabling full programming and debugging through the on-board PKOB nano interface.

1.7 Generating the Demo using Atmel START

Atmel START, an innovative online tool, can be used to select and customize additional code examples including single-click support for 100+ Click sensor boards (out of the 450+ models available so far) for custom implementation on the AVR-IoT WG Development Board. The codes can be downloaded by clicking **Browse Examples** on the Atmel START page, as shown in Figure 1-12. Since START can only generate code for the AVR-IoT Development Board, and supporting libraries are not yet supported in MPLAB, the PIC® devices are not currently supported.

To generate the microcontroller code used on the AVR-IoT Development Board, select **Browse Examples** from the Atmel START home page (<https://start.atmel.com/>) and follow these simple steps:

1. Search and select the AVR-IoT WG Sensor Node Development Board (see Figure 1-13).
2. To download the demo code as it is, click **Download Selected Example**. To make modifications to the code, click **Open Selected Examples** (see Figure 1-14).
3. To make changes to the configuration, such as the Google Cloud project details, scroll down the page to the AVR-IoT WG Sensor Node panel, as shown in Figure 1-15. The user can find their Google Cloud Project details like Project ID at <https://console.cloud.google.com/cloud-resource-manager>.
4. Once these changes are made, the following options are available: Preview the code, save the configuration for later use, or export the project to a selected development environment. To select one of the options, click the corresponding tab on the top of the page shown in Figure 1-16.
5. If the project is to be exported, click the **EXPORT PROJECT** tab and select which IDE or tool to be used. Then click the **DOWNLOAD PACK** button. Once downloaded, follow the “Getting Started With Atmel Studio 7” guide to import the Start project to Atmel Studio (see Figure 1-17).

Figure 1-12. Atmel START Homepage

Atmel | START

This tool will help you select and configure software components and tailor your embedded application in a usable and optimized manner.

[CREATE NEW PROJECT](#) [BROWSE EXAMPLES](#)

Step 0

Getting started

To get started you can either create a new project from scratch or open an existing example. In both cases you can configure your software components and device settings such as clocks and pin layout. When you are done, you can export your project and open it using your favourite IDE for further development.

For more information on how to use Atmel START, read the [Getting Started guide](#) or watch our [video tutorials](#).

Load existing project

[LOAD PROJECT FROM FILE](#) [RESUME AUTO SAVED PROJECT](#)

Use this option if you want to restore a locally saved project. Browse and select either a project file (*.at26) or a configuration file (*.actart*.json).

Your latest project will always be stored in your web browser. Use this option to pick up where you left off.

© 2018 Microchip Technology Inc.

Figure 1-13. START Example Browser

Atmel | START ← Return To Front Page | Help

BROWSE EXISTING EXAMPLES

Atmel START examples are a great starting point for embedded programmers. Example projects will work "out of the box" but are also easily to modify using Atmel START. Example projects are tailored for each compatible board and device, resulting in high quality production ready code with a small memory footprint. Read the available user guides to get more information about each example.

Looking for older examples? [Atmel Software Framework 3](#) contains previous releases of non-configurable examples.

Search: Category: Board:

Name	Categories	Description	Board(s) supported	User guide
Step 1 AVR IoT WG Sensor Node		A sensor node for the Google IoT Core Cloud based on the ATmega4808, WINC1510 and ECC608A.	ATmega4808 AVR IoT WG	User guide
Click Examples AVR IoT WG Sensor Node with AirQuality Click		A sensor node for the Google IoT Core Cloud based on the ATmega4808, WINC1510 and ECC608A.	ATmega4808 AVR IoT WG	User guide
AVR IoT WG Sensor Node with Weather Click		A sensor node for the Google IoT Core Cloud based on the ATmega4808, WINC1510 and ECC608A.	ATmega4808 AVR IoT WG	User guide

Showing 3 of 301 examples.

Step 2A **Step 2B**

[DOWNLOAD SELECTED EXAMPLE](#) [OPEN SELECTED EXAMPLE](#)

Figure 1-14. AVR-IoT WG Firmware Map

Atmel | START ATmega4808 ← Return To Front Page | Help And Support

[VIEW CODE](#) [SAVE CONFIGURATION](#) [EXPORT PROJECT](#)

MY SOFTWARE COMPONENTS

DASHBOARD

- Application
- Middleware
- Driver
- System driver

AVR IOT WG SENSOR NODE

- SCHEDULER
- ADC_0
- USART_0
- MQTT_IOT_0
- CLI_0
- WIFI_0
- CRYPTO_AUTHENTICATION
- SPI_0
- I2C_0
- BOD
- CLKCTRL
- CPUINT
- SLEEPCTRL

AVR IOT WG SENSOR NODE
A sensor node for the Google IoT Core Cloud based on the ATmega4808, WINC1510 and ECC608A.

Figure 1-15. AVR-IoT WG Configuration Section

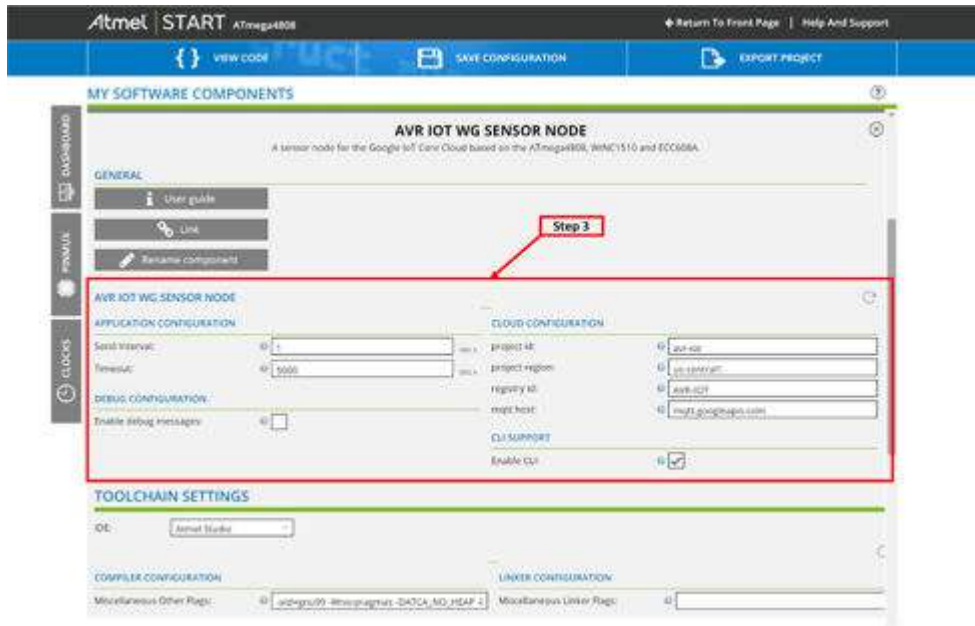


Figure 1-16. User Options Tabs

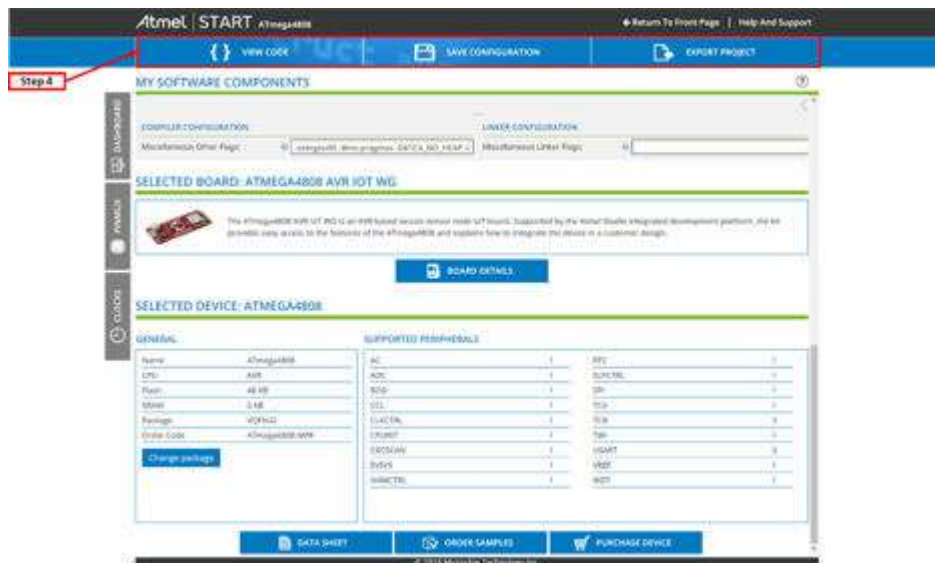
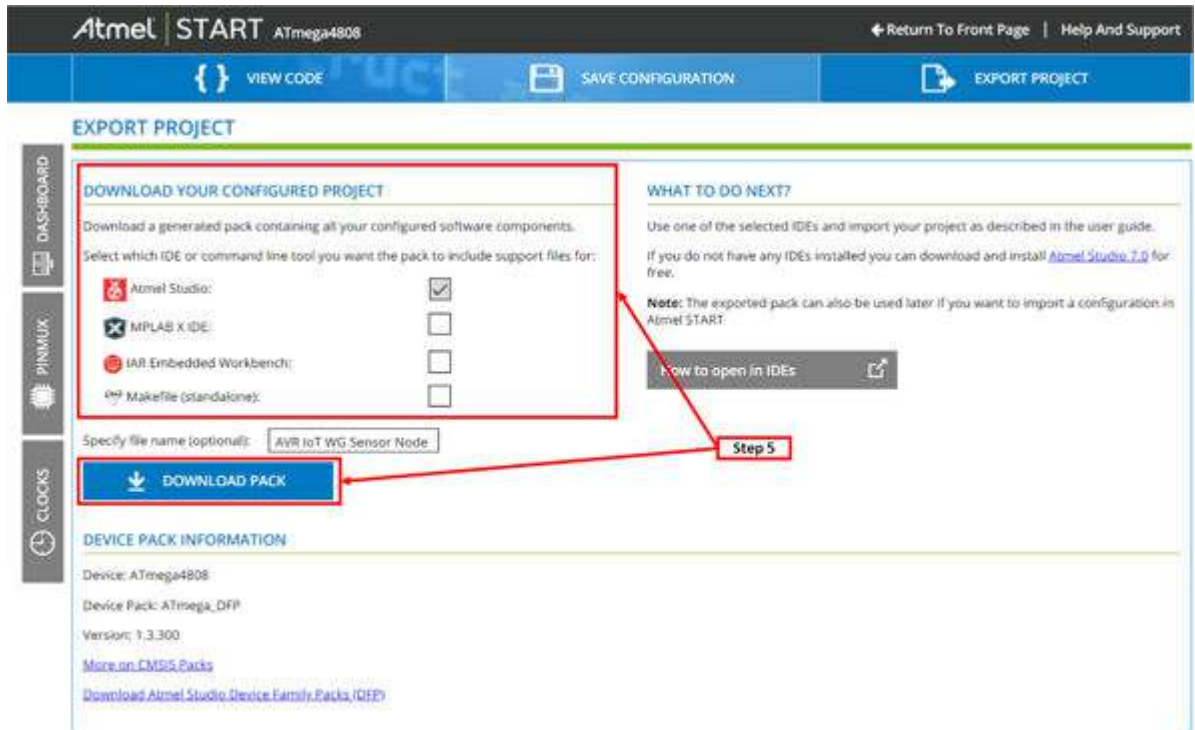


Figure 1-17. Exporting a Project



1.8 Exporting AVR-IoT WG Sensor Node START Project to Atmel Studio

After generating the project in Atmel START, export it to Atmel Studio to be compiled, linked, and eventually programmed into the AVR microcontroller. For instructions on how to import Atmel START projects into Atmel Studio and program them onto the board, refer to the Atmel START User Guide.

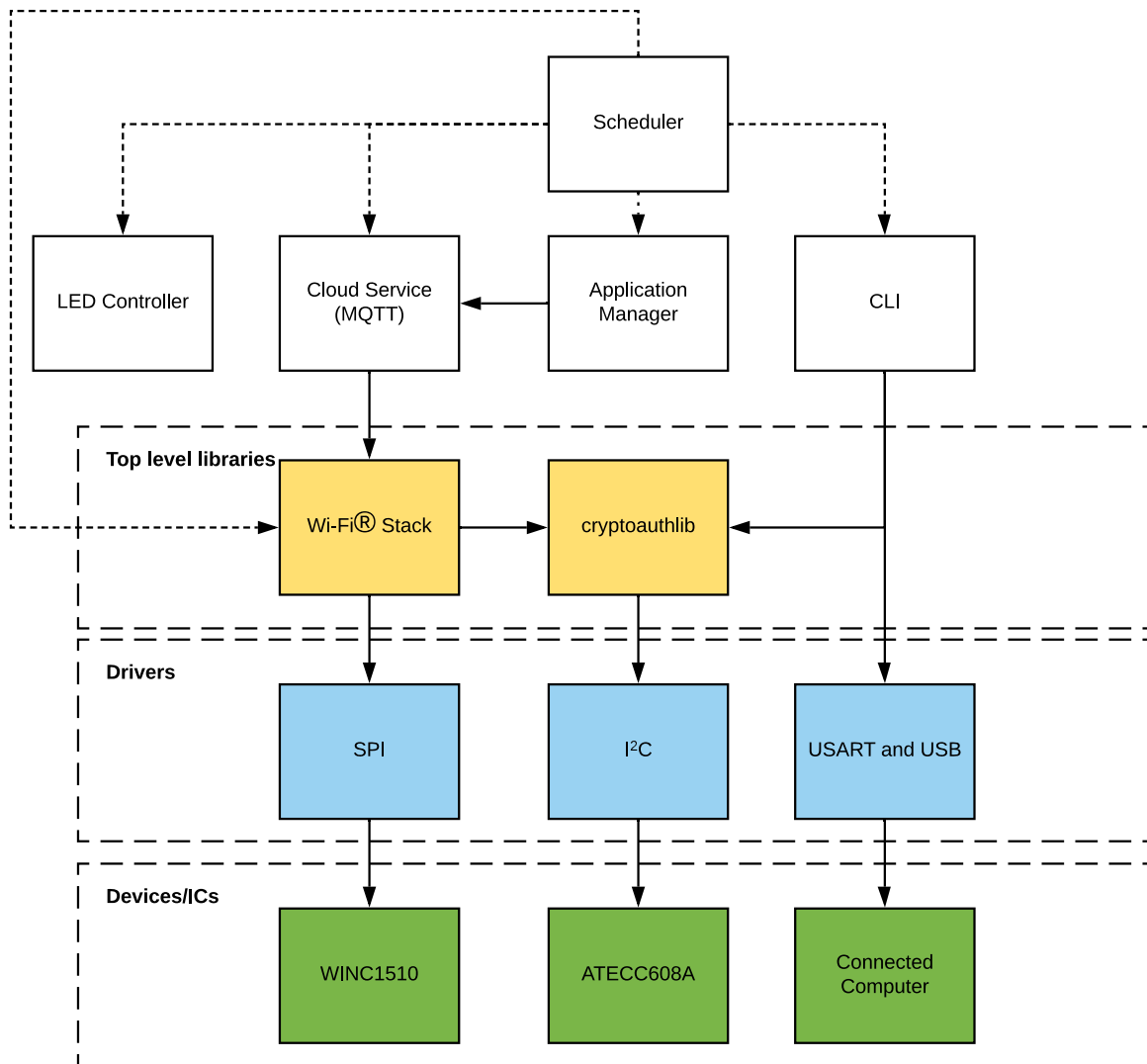
2. Firmware

After downloading the demo from Atmel START, the firmware code can be accessed. This section presents an overview of the firmware and a more thorough look at some of the more important parts. [Figure 2-1](#) shows a rough block diagram of the architecture with an accompanying description in [Table 2-1](#). If a module has a dotted line to it, the scheduler calls a function from that module at some time.

Table 2-1. Firmware Modules

Module Name	Description/Purpose	Where to Find in Source
Scheduler	The scheduler resides at the top, controlling which routine to run and at which time	src/timeout.c
LED Controller	Sets and clears LEDs to reflect the internal state of the firmware	led.c
Cloud Service (MQTT)	Handles MQTT messages for both receiving and sending	cloud/*
Application Manager	Top level manager for the entire application. Initializes the system and starts the scheduler.	application_manager.c
CLI	The command-line interface. Handles communication with the PC through USB.	cli/cli.c
Wi-Fi Stack	A complete Wi-Fi Stack. A detailed understanding is not required	cloud/wifi_service.c, winc/*
cryptoauthlib	A fully features cryptographic library. A detailed understanding not required.	cryptoauthlib/*
SPI	A basic SPI driver for the ATmega4808	src/spi_basic.c
I ² C	A simple I ² C driver for the ATmega4808	src/i2c_master.c, src/i2c_simple_master.c, i2c_types.c
USART and USB	A driver to write USART messages over USB	src/usart_basic.c

Figure 2-1. AVR IoT WG Board Firmware Architecture Block Diagram



2.1 Implementing: Echo Single Digit

The easiest way to add a custom application to the firmware is through the *sendToCloud* and *receivedFromCloud* functions found in *main.c*. The scheduler calls both these functions at some point in time, sending and receiving data from the cloud. The use case “Echo single digit” is introduced to demonstrate this. The cloud can send a single digit to the device, which is promptly returned to the cloud and logged.

Step One: Firmware Code

When a message is received from the cloud, the *receivedFromCloud* function is invoked. The received messages are in JSON format. The firmware does not support JSON parsing at this time. It must be done manually. In this example, the message is parsed by looking for the token *echo* with an accompanying number, for instance, *{“echo”:“4”}*. When the number is extracted from the message, it is assigned to a global variable *echo_num*. This is done in the following manner:

```
uint8_t echo_num = 0;
```

```
void receivedFromCloud(uint8_t *topic, uint8_t *payload)
{
    char *echoToken = "\"echo\":";
    char *subString;

    if ((subString = strstr((char *)payload, echoToken)) {
        // Extract the one digit following "echo:", remove '0' to convert to int
        echo_num = subString[strlen(echoToken) + 1] - '0';
    }
}
```

The *sendToCloud* is called every second as long as there is a valid connection to the cloud. The default firmware already logs temperature and light. By modifying the JSON message, the *echo_num* variable can be sent at the same time. The code below implements this, sending back the updated *echo_num* to the cloud every second alongside the temperature and light values.

```
void sendToCloud(void)
{
    static char json[70];

    // This part runs every CFG_SEND_INTERVAL seconds
    int rawTemperature = SENSORS_getTempValue();
    int light = SENSORS_getLightValue();
    int len = sprintf(json, "{\"Light\":%d,\"Temp\": \"%d.%02d\", \"echo\":%d}", light,
rawTemperature / 100, abs(rawTemperature) % 100, echo_num);

    if (len > 0) {
        CLOUD_publishData((uint8_t *)json, len);
        LED_flashYellow();
    }
}
```

Step Two: Sending from the cloud

Open the AVR-IoT webpage and connect the device to the internet (see [1.2 The AVR-IoT Webpage](#)). The *echo* number has already shown up as a graph with an initial value of zero. Scroll down to *Control Your Device*, and add a new slider. Set the slider to a minimum of zero, a step of one and a maximum of 9. This ensures that no two-digit numbers are sent. Rename the slider to *echo*. See [Figure 2-2](#) for the correct slider and output.

Figure 2-2. Adding the Echo Slider



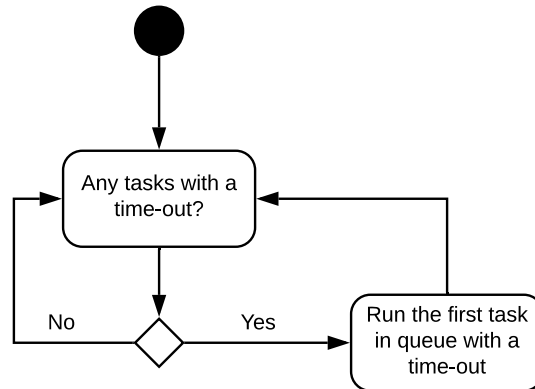
2.2 The Scheduler

The scheduler has previously been introduced as the module which controls what is to run at which time. Each task has a time-out, which is the given time which must elapse before the task is run. When the time for a task expires, the scheduler executes that task whenever the CPU is available. A simplified state diagram of the scheduler is shown in [Figure 2-3](#).



Tip: Notice there is no notion of priority, meaning all tasks have equal priority. This type of scheduling is a **Round Robin Scheduler**.

Figure 2-3. UML State Diagram of the Scheduler



2.2.1 Creating and Scheduling a Task

For some applications, it is desirable to have a task run on another interval than the *sendToCloud* routine. This can be solved by adding a task directly to the scheduler. A task is scheduled with a call to the *scheduler_timeout_create* function. It requires two arguments, a pointer to a *timer_struct_t* and a time-out. The *timer_struct_t* holds internal state information to the scheduler, in addition to a function pointer. When the scheduler deems the task ready to run, this function pointer is called. This function pointer must thus point to the task.

Step One: Creating the tasks function

The goal is a task that runs every five seconds, blinking one of the onboard LEDs.

```

static absolutetime_t led_task()
{
    LED_YELLOW_toggle_level();
    return 5000L;
}
    
```

Notice the *return 5000L*;. This is the number of ticks before the task is scheduled to run again.

Step Two: Creating the *timer_struct_t* and scheduling it

```

int main(void)
{
    application_init();

    ENABLE_INTERRUPTS();
    timer_struct_t led_timer = {led_task, NULL};
    scheduler_timeout_create(&led_timer, 5000L);

    while (1) {
        runScheduler();
    }

    return 0;
}
    
```

2.2.2 Considerations

The scheduler has no notion of priority, making it impossible to guarantee that a given task is going to run before a deadline. In other words, the scheduler is **NOT a real-time scheduler**, and can thus not be used to run time-sensitive systems. For instance, controlling the rotors of a quadcopter. If a quadcopter hovers, the rotors must run at the same speed. By measuring the speed of the rotors, the power to the different rotors can be adjusted, *in real-time*, to make sure their speed is equal. If the time between each of these adjustments is too high, the quadcopter becomes

unstable and crashes. This *control loop* has thus a real-time requirement, and cannot be implemented with the given scheduler.

3. Stepper Motors: The Basics

The following section introduces the basic theory required to understand a simple stepper motor and how to control it. The basics of a stepper motor are discussed, how to rotate it, and how to control rotation properties such as speed. An example control pulse is presented with an accompanying discussion of how to generate such a pulse.

3.1 Stepper Motor Basics

A stepper motor is a simple type of DC motors which provides high torque and precision. The motor described here is a **Bipolar Permanent Magnet Stepper Motor**, which is the simplest type. It operates with a permanent magnet as the rotor and several electromagnets as the stator. By magnetizing the different electromagnets, the rotor is thrown into alignment. The principal operation is illustrated in [Figure 3-1](#), showing a motor rotating a full 360 degrees.

The first step has the motor in a 0° position by magnetizing the upper and lower electromagnet. To rotate the motor 90°, magnetization is moved to the right and left, pulling the rotor into position. This continuous swapping of magnetization makes the motor rotate.

The operation described above is named **full step operation**, as the motor moves to complete alignment with one pair of active electromagnets. By magnetizing more than one pair, it is possible to achieve **half step operation**, as illustrated in [Figure 3-2](#). The resulting force points towards the middle of the two magnets, allowing for more precise stepping.

Figure 3-1. Illustration of a Stepper Motor's Basic Operation

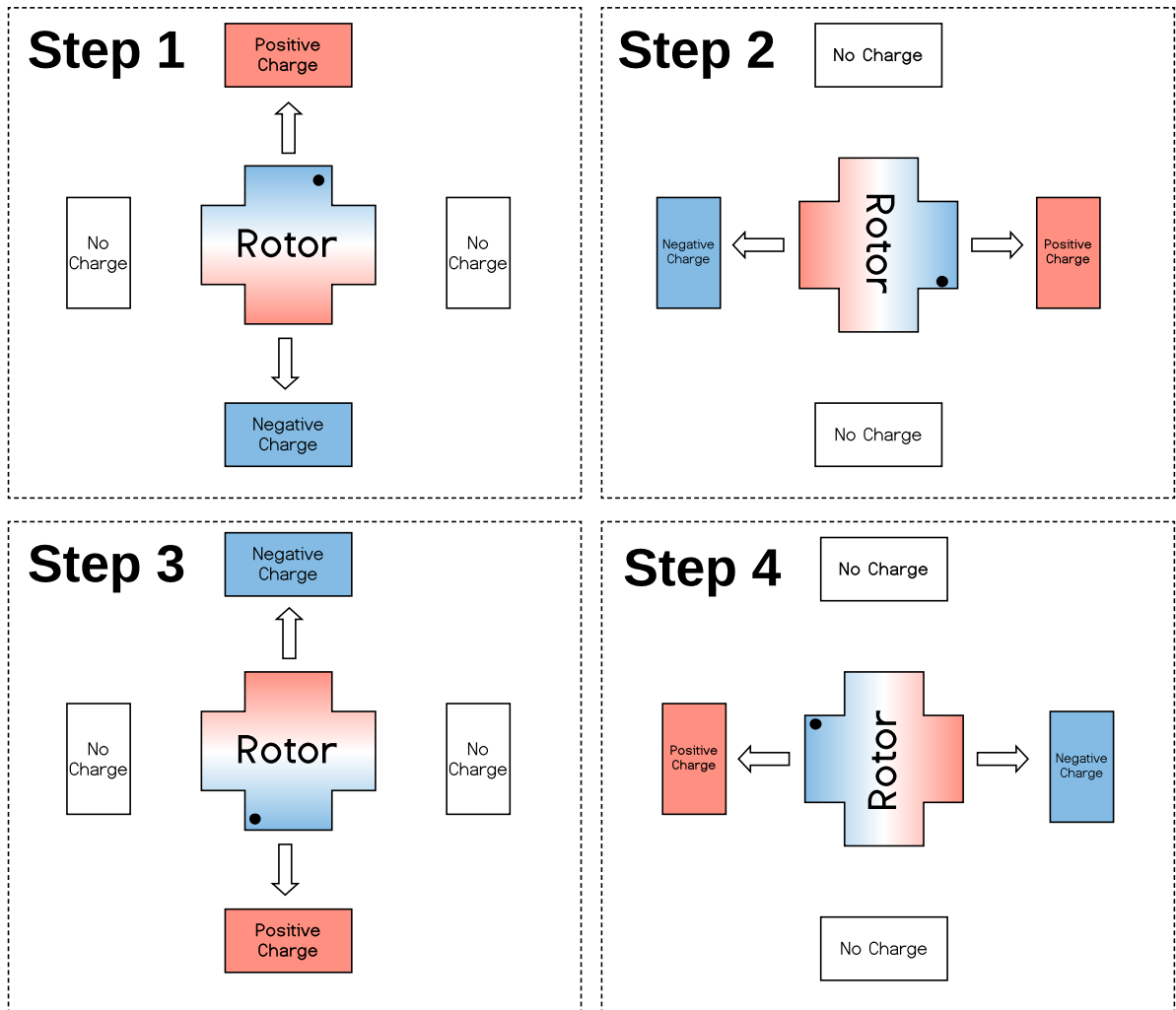
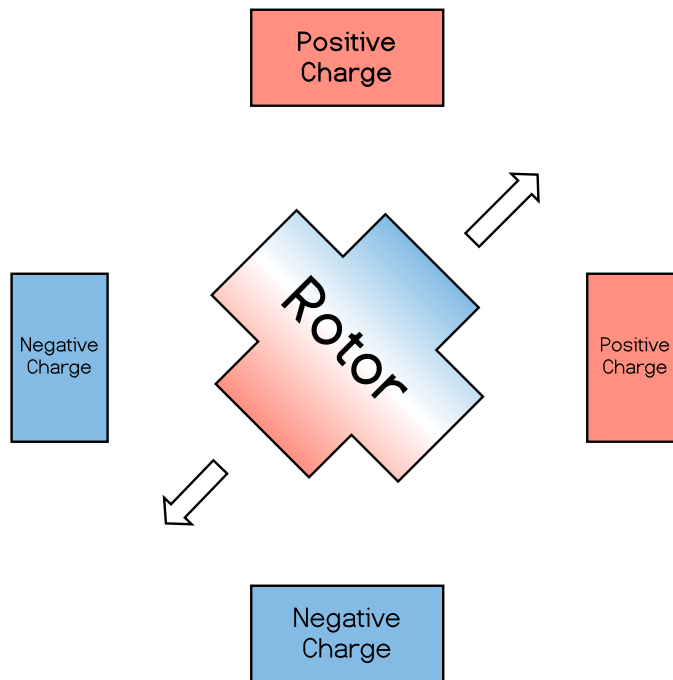


Figure 3-2. Stepper Motor in a Half Step



The number of degrees the motor moves with each step is called the **step angle**. The motor discussed so far only has four magnets, allowing for a minimum step angle of 45° in half step operation. Most applications would require a much smoother rotation with a much lower step angle. The step angle can be reduced by adding more magnets, which is why most stepper motors have many more than four.



Tip: For more information on stepper motors, see <https://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en012151>.

3.2 Controlling a Stepper Motor

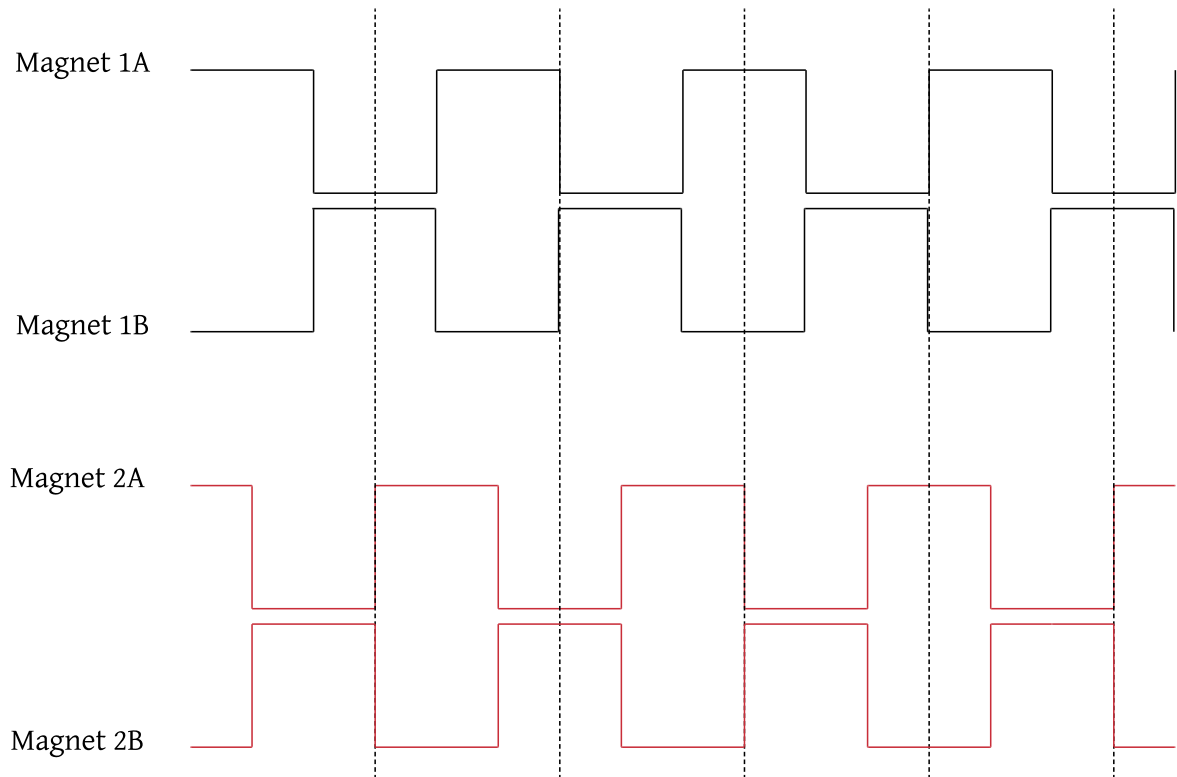


Important: This section is not necessary to understand how to make the motor run. It is, however, considered “Good to Know” material for debugging and expanding upon the design.

Controlling a stepper motor is all about magnetizing the correct magnets. Electromagnets are magnetized according to Amperes Law: *The magnetomotive force induced by a coil is given by $\mathcal{F} = NI$, where I is the current going through the coil and N the number of windings in the coil.* In short, by running current through the appropriate coils, the correct magnets are magnetized.

Figure 3-3 shows the basic waveform for a motor with two magnets pairs, one and two. The magnets in each pair are inverted to each other, creating the driving magnetomotive force. The second pair is offset by half a period compared to the first pair.

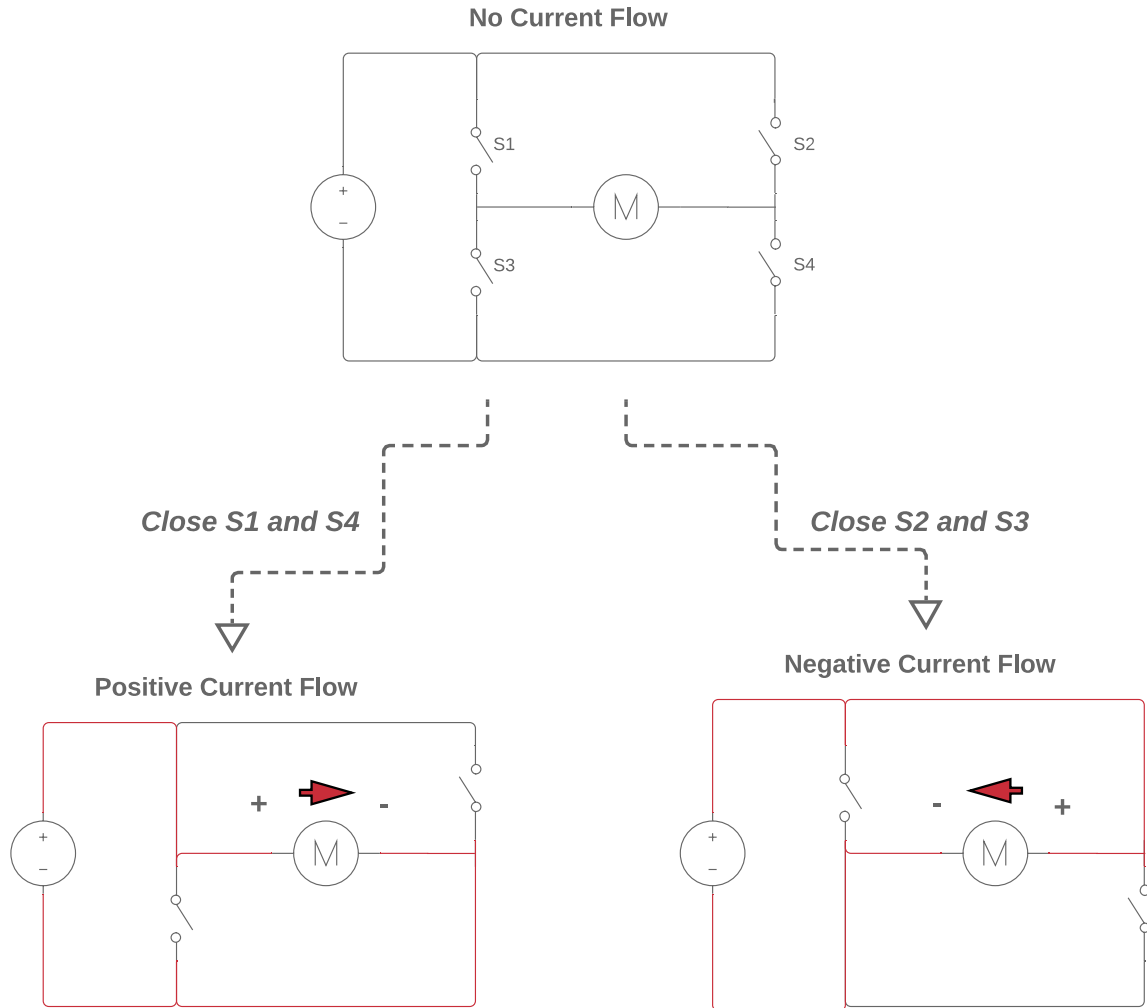
Figure 3-3. Waveform to Drive a Stepper Motor in Full Step



As the force is given by $\mathcal{F} = NI$ with a fixed number of turns N , the current I is how the amount of torque is controlled. For many applications, such as controlling a blind, a microcontroller is not able to provide the required current. To solve this problem, an **H-Bridge** is used.

The H-Bridge allows an external power supply to supply the current while allowing the microcontroller to provide the waveform by setting four switches per magnet pair. [Figure 3-4](#) illustrates this concept. When a positive current is required over a magnet pair, switch S1 and S4 is closed. Respectively, S2 and S3 can be closed to provide a negative current flow.

Figure 3-4. H-Bridge in Three States



4. Using the Stepper 2 Click



Important: This section describes the implementation of a Stepper 2 Click driver. The complete version of the driver with the examples described below can be accessed at <https://start.atmel.com/#examples> under the name *AVR IoT WG Sensor Node with Stepper 2 Click*.

This section implements the driver for the [Stepper 2 Click board](#) from MikroElektronika. It utilizes the [A4988](#) stepper motor controller, providing a basic PWM interface. There are five pins to connect with, presented in [Table 4-1](#).

Table 4-1. Stepper 2 Click Pinout

MikroBus Pin	ATmega 4808 Pin	Abbreviation	Details
1	PD7	\overline{CE}	Chip Enable - Must be low for the chip to function
2	PA0	\overline{RST}	Reset - If this pin is low, the chip is reset
3	PC3	\overline{SL}	Sleep - If this pin is low, the device enters a sleep mode
16	PD4	ST	Step Trigger - A PWM pulse which steps the motor for each pulse
15	PD6	DIR	Direction - If the pin is high, the motor turns clockwise. If the pin is low, the motor turns counter clockwise.

4.1 Adding the PWM with Atmel START



Tip: For more information on PWM generation in ATmega4808, see the [data sheet \(https://www.microchip.com/wwwproducts/en/ATMEGA4809\)](https://www.microchip.com/wwwproducts/en/ATMEGA4809), section 19.

The A4988 works by receiving a PWM signal on the **ST** pin, stepping the motor once for every pulse. According to the [AVR IoT WG Board schematics](#), this is the PD4 pin on the ATmega4808. The [device data sheet](#) states PD4 supports PWM through the fourth waveform generator output, as shown in [Figure 4-1](#).

AVR® Home Automation Kit

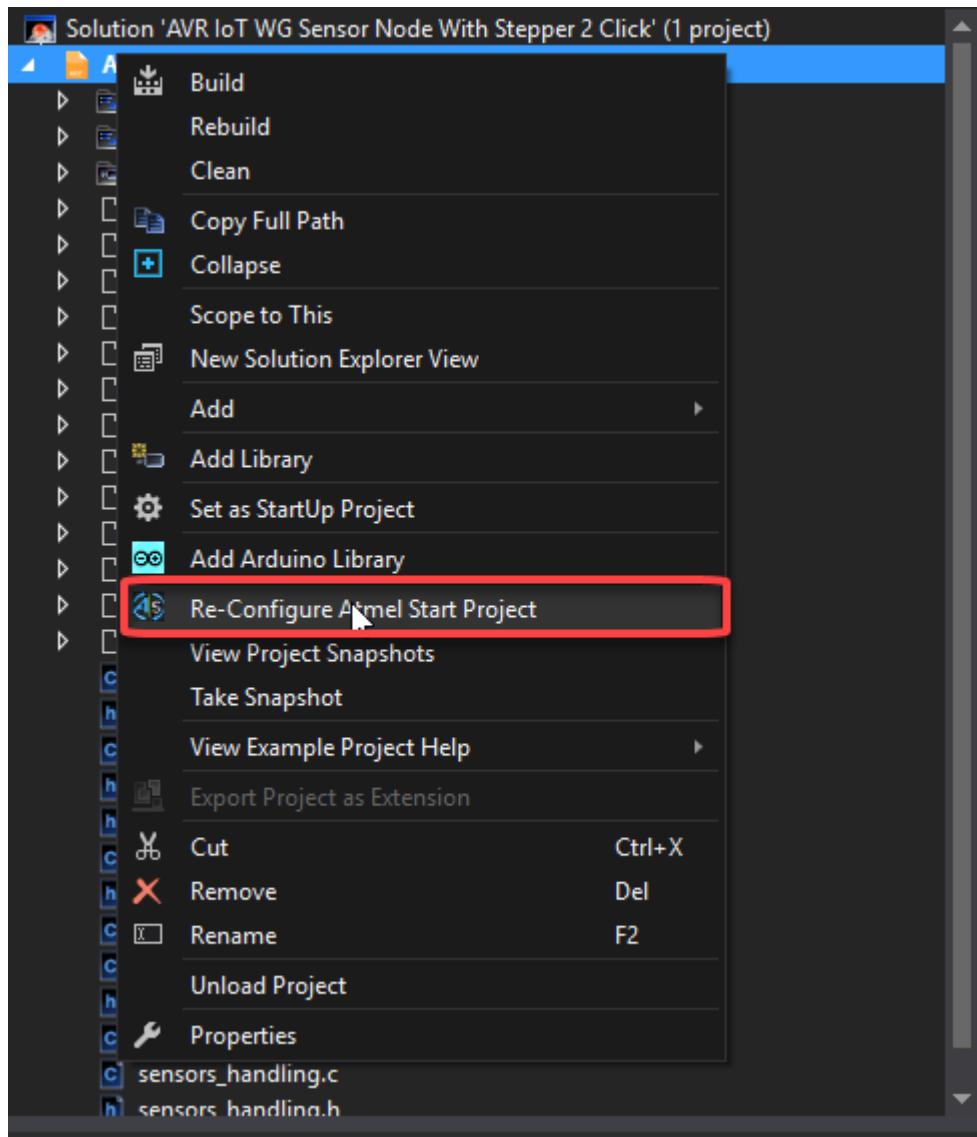
Using the Stepper 2 Click

Figure 4-1. Printout of the ATmega4808 Data Sheet - PD4 Multiplexing

10	PD0		AIN0				0-WO0(3)
11	PD1		AIN1	P3			0-WO1(3)
12	PD2		AIN2	P0			0-WO2(3)
13	PD3		AIN3	N0			0-WO3(3)
14	PD4		AIN4	P1			0-WO4(3)
15	PD5		AIN5	N1			01_05(3)
16	PD6		AIN6	P2			
17	PD7	VREFA	AIN7	N2			
18	AVDD						
19	GND						
20	PF0	TOSC1			2, TxD		0-WO0(3)

The fourth waveform generator output is generated through the Timer A (TCA) module which, is supported in Atmel START. Open the existing IoT WG Project from the previous sections in Atmel Studio, and *reconfigure* the start project as shown in [Figure 4-2](#).

Figure 4-2. Reconfigure Atmel START Project



Add the timer module by clicking “Add Software Component → Timer” and open it. The correct configuration is shown in [Figure 4-3](#). The rest of this section describes what these settings are.

On the ATmega4808, it is only possible to generate a PWM pulse on PD4 if the timer is set in *Split Mode*. When a timer is in split mode, it effectively divides the 16-bit timer into two 8-bit timers, one of which generates the WO/4 output. In split mode, the **HPER** register defines the PWM period, and the **HCMP1** defines the duty cycle. Our desired waveform has a constant period, flipping the logical value at a constant rate. See [Figure 4-4](#) for an example. This is achieved by setting $HCMP1 = HPER/2$. The period $HPER = 0x7f$ is selected to give about 3 ms, more on this later. To enable waveform generation, output PD4 is selected as WO/4 in addition to enabling **HCMP1EN**, **HCMP1OV** and **ENABLE**.

Figure 4-3. Timer A Configured to Run PWM for Stepper Click 2

MOTOR_TIMER

TCA init driver in Split Mode

GENERAL

- [User guide](#)
- [Rename component](#)
- [Remove component](#)

COMPONENT SETTINGS

Driver: Drivers:TCA:Init

Mode: Split Mode

CLOCKS

TCA: Main Clock (CLK_MAIN) (10 MHz)

COMPONENT SIGNALS

WO/0: ----

WO/1: ----

WO/2: ----

WO/3: ----

WO/4: PD4

WO/5: ----

DRIVERS:TCA:INIT (SPLIT MODE) CONFIGURATION ON TCA0

CONFIGURATION

ENABLE: Module Enable:

CLKSEL: Clock Selection: System Clock / 256

DBG RUN: Debug Run:

LOW BYTE COMPARE CONFIGURATION

LCMP0EN: Low Compare 0 Enable:

LCMP1EN: Low Compare 1 Enable:

LCMP2EN: Low Compare 2 Enable:

LCMP0OV: Low Compare 0 Output Value:

LCMP1OV: Low Compare 1 Output Value:

LCMP2OV: Low Compare 2 Output Value:

LCMP0: Compare value Channel 0: hex v

LCMP1: Compare value Channel 1: hex v

LCMP2: Compare value Channel 2: hex v

LCNT: Low-byte Timer Counter Register: hex v

LPER: Low-byte Timer Period Register: hex v

HIGH BYTE COMPARE CONFIGURATION

HCMP0EN: High Compare 0 Enable:

HCMP1EN: High Compare 1 Enable:

HCMP2EN: High Compare 2 Enable:

HCMP0OV: High Compare 0 Output Value:

HCMP1OV: High Compare 1 Output Value:

HCMP2OV: High Compare 2 Output Value:

HCMP0: Compare value of channel 0: hex v

HCMP1: Compare value of channel 1: hex v

HCMP2: Compare value of channel 2: hex v

HCNT: High-byte Timer Counter Register: hex v

HPER: High-byte Period Register: hex v

INERRRUPT CONFIGURATION

Include ISR harness in driver_isr.c:

HUNF: High Underflow Interrupt Enable:

LCMP0: Low Compare 0 Interrupt Enable:

LCMP1: Low Compare 1 Interrupt Enable:

LCMP2: Low Compare 2 Interrupt Enable:

LUNF: Low Underflow Interrupt Enable:

Figure 4-4. The Desired Alternating Waveform for the Stepper Motor Click

4.2 Selecting the Correct PINMUX

The multiplexed mode for each pin can be configured by selecting the PINMUX section of Atmel START at the very left-hand side. The PD4 pin is already configured by the TIMER module, leaving the remaining pins from [Table 4-1](#) to

© 2020 Microchip Technology Inc.

User Guide

DS50002957A-page 30

AVR® Home Automation Kit

Using the Stepper 2 Click

be configured. Configure each pin according to [Table 4-2](#). When everything has been configured, press “Generate Project”.

Table 4-2. PINMUX Configuration for Clicker 2

Pin	Label	Pin Mode	Initial Level
PD7	MOTOR_CHIP_EN	Digital Output	Low
PA0	MOTOR_RST	Digital Output	High
PC3	MOTOR_SL	Digital Output	High
PD4	MOTOR_ST	Digital Output	High
PD6	MOTOR_DIR	Digital Output	High

Connecting the motor

The Stepper 2 click has six inputs, **1A, 1B, 2A, 2B, VCC and GND**. These are connected according to the schematic in [Figure 4-5](#). Note the need for an external power supply, as a microcontroller is not able to provide enough current. A photo of the connected board can be seen in [Figure 4-6](#).

Figure 4-5. Schematics to Connect the Stepper 2 Click Board

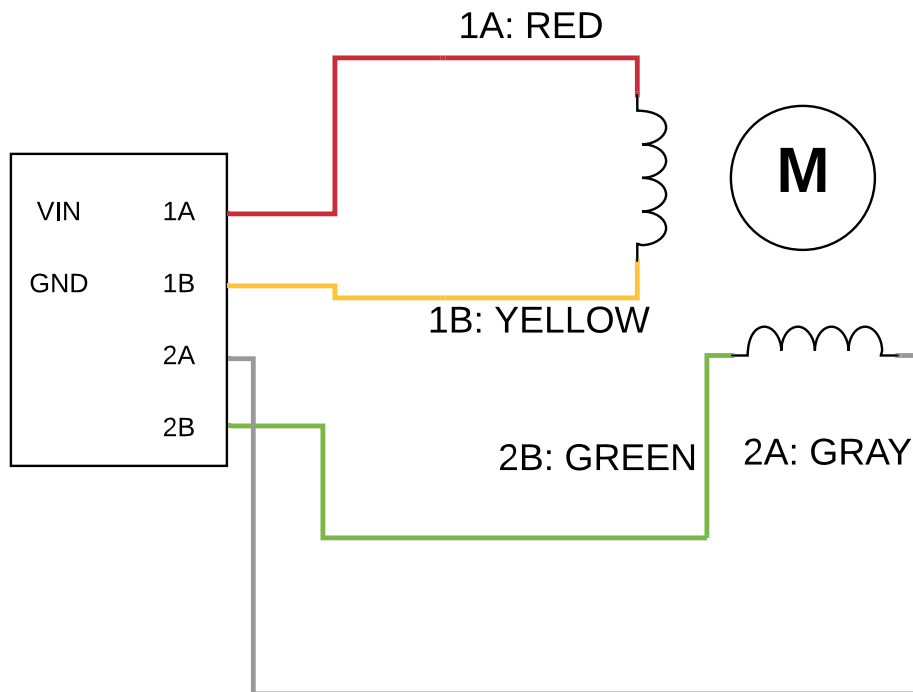
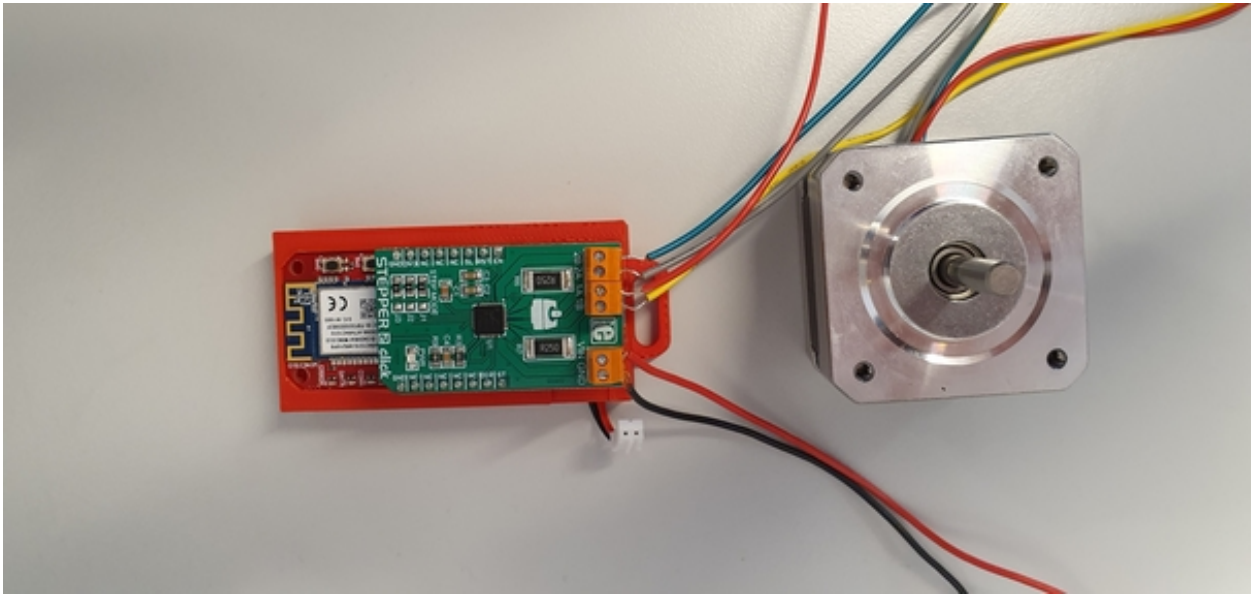


Figure 4-6. A Stepper 2 Click Connected to Both the IoT WG Board and a Stepper Motor



4.3 Writing the Driver

The motor will run at boot with the project generated through Atmel START. The next step is to add functions to start/stop the motor and set the speed and direction. The motor starts and stops by starting/pausing the PWM signal, generated through the timer A (TCA) module. The motor is started by setting the **SPLIT_ENABLE** bit in the **CTRLA** register of the TCA module. Equivalently, it is stopped by clearing that bit.

```
void motor_start(void) {
    TCA0.SPLIT.CTRLA |= 1 << TCA_SPLIT_ENABLE_bp;

    // Enable the stepper click 2
    MOTOR_CHIP_EN_set_level(false);
}

void motor_stop(void) {
    TCA0.SPLIT.CTRLA &= ~(1 << TCA_SPLIT_ENABLE_bp);

    // Reset counter
    TCA0.SPLIT.HCNT = 0;

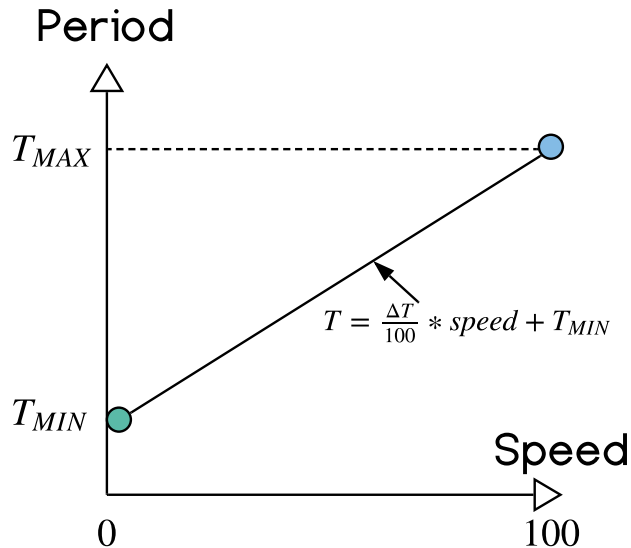
    // Disable the stepper click 2
    MOTOR_CHIP_EN_set_level(true);
}
```

The speed of the motor is determined by the period of the PWM pulse. As the period increases, the speed decreases. To have an easy-to-use interface, the speed is set as a percentage of a maximum and minimum period. The default implementation has a maximum period of $T_{MAX} = 0.005s = 5ms$ and a minimum period of $T_{MIN} = 0.001s = 1ms$. The target period T is a value between T_{MAX} and T_{MIN} based on the inputted speed percentage. This is equivalent to finding a point on a line between two points, as illustrated in [Figure 4-7](#). The scaled period is shown in [Equation 4-1](#).

Equation 4-1. Scaling the Period T Between T_{MAX} and T_{MIN}

$$T = \frac{T_{MAX} - T_{MIN}}{100} * speed + T_{MIN}$$

Figure 4-7. The Equivalent Line Problem for Scaling the Period T



To convert a desired period to the equivalent value in **HPER**, multiply the period with the clock speed, as shown in [Equation 4-2](#).

Equation 4-2. HPER Calculation Based on Period

$$HPER = T * F_S$$

Where $F_S = F_{CPU}/256$. The term F_S stands for *Frequency Scaled*, the CPU frequency divided by 256, as this is the chosen clock divider (see [4.1 Adding the PWM with Atmel START](#)). As discussed earlier, $HCMP1 = HPER/2$ to get an even pulse.

```
#define F_S (F_CPU/256)
#define T_MAX 0.005
#define T_MIN 0.001
#define DELTA_T (T_MAX - T_MIN)

void motor_set_speed(uint8_t speed) {
    // Make speed = 100 give minimum period
    speed = 100 - speed;

    double period = (DELTA_T / 100.0) * (double) speed + T_MIN;

    uint8_t period_hper = F_S * period;
    TCA0.SPLIT.HPER = period_hper;
    TCA0.SPLIT.HCMP1 = period_hper / 2;
}
```

4.4 Connecting to the Cloud

All that remains is to connect the motor to the cloud. The following section implements:

- A toggle to start/stop the motor
- A toggle to set the direction of the motor
- A speed slider to adjust the motor speed

On the AVR IoT website, add two toggles, *run*, and *direction*. Add one slider *speed* with a minimum value of zero, step size of one and a maximum value of 100. These attributes are sent to the device as a JSON string, for instance,

`{"run":1,"direction":0,"speed":74}`". See [Figure 4-8](#) for the correct setup. The device parses this string and acts accordingly, which can be implemented as follows:

```
void receivedFromCloud(uint8_t *topic, uint8_t *payload)
{
    debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "topic: %s", topic);
    debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "payload: %s", payload);

    char *runToggle = "\"run\":";
    char *directionToggle = "\"direction\":";
    char *speedSlider = "\"speed\":";
    char *subString;

    // Start / Stop motor
    if ((subString = strstr((char *)payload, runToggle)) {
        if (subString[strlen(runToggle)] == '1'){
            motor_start();
        }else if(subString[strlen(runToggle)] == '0'){
            motor_stop();
        }else{
            debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "Supplied run command has illegal
parameter: %s\n", subString);
        }
    }

    // Set the motor direction
    if ((subString = strstr((char *)payload, directionToggle)) {
        if (subString[strlen(directionToggle)] == '1'){
            motor_set_direction(MOTOR_DIRECTION_CLOCKWISE);
        }else if(subString[strlen(directionToggle)] == '0'){
            motor_set_direction(MOTOR_DIRECTION_COUNTER_CLOCKWISE);
        }else{
            debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "Supplied direction command has
illegal parameter: %s\n", subString);
        }
    }

    // Set speed
    if ((subString = strstr((char *)payload, speedSlider)) {
        uint8_t speedStrLength = 4;
        char speedStr[speedStrLength];

        // Start location of the speed number.
        char *currentChar = &subString[strlen(speedSlider)] + 1;

        // As long as we do not hit a ", there are more digits. Record these
        uint8_t i = 0;
        while(*currentChar != '"'){
            speedStr[i] = *currentChar;
            currentChar++;
            i++;
            if(i > speedStrLength){
                debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "Speed command has illegal
parameter: %s\n", subString);
                return;
            }
        }
        // Add the null terminator to make the string valid
        speedStr[i] = '\0';

        // Convert the speedStr to an integer
        char *endptr;
        uint8_t speed = strtol(speedStr, &endptr, 0);

        if(*endptr == '\0'){
            // Successful conversion of the entire string. Set speed
            debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "Setting speed %d \n", speed);
            motor_set_speed(speed);
        }else if(endptr == speedStr){
            // Something went wrong during the conversion. Not setting speed, raising warning.
            debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "The speed message is corrupted
(conversion error): %s \n", speedStr);
        }else{
            // The entire string was not converted. Something is wrong
            debug_printer(SEVERITY_WARNING, LEVEL_ERROR, "The speed message is corrupted
```

```
(part conversion error): %s \n", speedStr);  
    }  
  
    debug_printer(SEVERITY_NONE, LEVEL_NORMAL, "speed: %d", speed);  
}  
}
```

Figure 4-8. AVR IoT Webpage Configured for Motor Control

Control Your Device

If your device is [listening](#), you can use these controls to send it a base64 encoded JSON message. Adding a control will create a field with a customizable name. Toggle controls send numerical 0 and 1 values. Text and slider controls will send string values.

The screenshot displays the 'Control Your Device' interface. It is divided into three main sections: 'Toggles', 'Text Fields', and 'Sliders'.
1. **Toggles:** This section contains two toggle switches. The first is labeled 'run' and is currently turned on (blue). The second is labeled 'direction' and is currently turned off (grey). To the right of each toggle is a trash icon. An 'Add Toggle' button is located in the top right corner of this section.
2. **Text Fields:** This section is currently empty, with an 'Add Text Field' button in the top right corner.
3. **Sliders:** This section features a slider control for 'speed'. The slider has a blue knob positioned at 74, with the text 'Current: 74' to its right. Below the slider are three input fields: 'Min' with the value '0', 'Step' with the value '1', and 'Max' with the value '100'. A trash icon is also present to the right of the slider. An 'Add Slider' button is in the top right corner.
At the bottom of the interface is a red 'Send to device' button.



Tip: Strings in C are pointers to the first character in the sequence. The string continues by incrementing the address, until the zero character (0) is reached, indicating the end of the string. More information can be found at https://www.tutorialspoint.com/cprogramming/c_strings.htm.

The two toggles are parsed by finding the start of the tokens (for instance “run”:) extracting the single character after the token occurrence, which is either zero or one. Extracting the speed number is more tricky, as it contains an unknown number of characters (1, 2 or 3 characters). This problem is solved by finding the end location of the speed token, reading every character until a “ character is hit. When the “ character is hit, all digits have been extracted and can be converted to an integer using the `atoi()` function.

5. Revision History

Doc. Rev.	Date	Comments
A	02/2020	Initial document release

The Microchip Website

Microchip provides online support via our website at <http://www.microchip.com/>. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to <http://www.microchip.com/pcn> and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: <http://www.microchip.com/support>

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-5577-6

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit <http://www.microchip.com/quality>.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p>Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Tel: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com</p> <p>Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p>Austin, TX Tel: 512-257-3370</p> <p>Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p>Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p>Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p>Detroit Novi, MI Tel: 248-848-4000</p> <p>Houston, TX Tel: 281-894-5983</p> <p>Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p>Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p>Raleigh, NC Tel: 919-844-7510</p> <p>New York, NY Tel: 631-435-6000</p> <p>San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270</p> <p>Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078</p>	<p>Australia - Sydney Tel: 61-2-9868-6733</p> <p>China - Beijing Tel: 86-10-8569-7000</p> <p>China - Chengdu Tel: 86-28-8665-5511</p> <p>China - Chongqing Tel: 86-23-8980-9588</p> <p>China - Dongguan Tel: 86-769-8702-9880</p> <p>China - Guangzhou Tel: 86-20-8755-8029</p> <p>China - Hangzhou Tel: 86-571-8792-8115</p> <p>China - Hong Kong SAR Tel: 852-2943-5100</p> <p>China - Nanjing Tel: 86-25-8473-2460</p> <p>China - Qingdao Tel: 86-532-8502-7355</p> <p>China - Shanghai Tel: 86-21-3326-8000</p> <p>China - Shenyang Tel: 86-24-2334-2829</p> <p>China - Shenzhen Tel: 86-755-8864-2200</p> <p>China - Suzhou Tel: 86-186-6233-1526</p> <p>China - Wuhan Tel: 86-27-5980-5300</p> <p>China - Xian Tel: 86-29-8833-7252</p> <p>China - Xiamen Tel: 86-592-2388138</p> <p>China - Zhuhai Tel: 86-756-3210040</p>	<p>India - Bangalore Tel: 91-80-3090-4444</p> <p>India - New Delhi Tel: 91-11-4160-8631</p> <p>India - Pune Tel: 91-20-4121-0141</p> <p>Japan - Osaka Tel: 81-6-6152-7160</p> <p>Japan - Tokyo Tel: 81-3-6880-3770</p> <p>Korea - Daegu Tel: 82-53-744-4301</p> <p>Korea - Seoul Tel: 82-2-554-7200</p> <p>Malaysia - Kuala Lumpur Tel: 60-3-7651-7906</p> <p>Malaysia - Penang Tel: 60-4-227-8870</p> <p>Philippines - Manila Tel: 63-2-634-9065</p> <p>Singapore Tel: 65-6334-8870</p> <p>Taiwan - Hsin Chu Tel: 886-3-577-8366</p> <p>Taiwan - Kaohsiung Tel: 886-7-213-7830</p> <p>Taiwan - Taipei Tel: 886-2-2508-8600</p> <p>Thailand - Bangkok Tel: 66-2-694-1351</p> <p>Vietnam - Ho Chi Minh Tel: 84-28-5448-2100</p>	<p>Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p>Denmark - Copenhagen Tel: 45-4450-2828 Fax: 45-4485-2829</p> <p>Finland - Espoo Tel: 358-9-4520-820</p> <p>France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p>Germany - Garching Tel: 49-8931-9700</p> <p>Germany - Haan Tel: 49-2129-3766400</p> <p>Germany - Heilbronn Tel: 49-7131-72400</p> <p>Germany - Karlsruhe Tel: 49-721-625370</p> <p>Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p>Germany - Rosenheim Tel: 49-8031-354-560</p> <p>Israel - Ra'anana Tel: 972-9-744-7705</p> <p>Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p>Italy - Padova Tel: 39-049-7625286</p> <p>Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340</p> <p>Norway - Trondheim Tel: 47-72884388</p> <p>Poland - Warsaw Tel: 48-22-3325737</p> <p>Romania - Bucharest Tel: 40-21-407-87-50</p> <p>Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p>Sweden - Gothenberg Tel: 46-31-704-60-40</p> <p>Sweden - Stockholm Tel: 46-8-5090-4654</p> <p>UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>