USER MANUAL

EN RFM69HCW radio Arduino® shield

WPSH214

whadda.com

Introduction

To all residents of the European Union
Important environmental information about this product
This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.
If in doubt, contact your local waste disposal authorities.

Thank you for choosing Whadda! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

Safety Instructions

Read and understand this manual and all safety signs before using this appliance.

For indoor use only.

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorized way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman Group nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Keep this manual for future reference.

What is Arduino®

Arduino® is an open-source prototyping platform based on easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language (based on Wiring) and the Arduino® software IDE (based on Processing). Additional shields/modules/components are required for reading a twitter message or publishing online. Surf to www.arduino.cc for more information.

Product Overview

The RFM69HCW module is an inexpensive and versatile radio module. You can use it to send text or binary data between two or hundreds of modules. It's perfect for building inexpensive short-range wireless networks for home and building automation, sensor networks, automated meter readings, wireless alarm and security systems, industrial monitoring and control, and many more applications!

The RFM69HCW transceiver module operates over a wide frequency range, including the 315, 433, 868 and 915 MHz license-free ISM (Industry Scientific and Medical) frequency bands. All major RF communication parameters are programmable, and most can be set dynamically. The RFM69HCW is optimized for low power consumption while offering high RF output power and channelized operation.

This WPSH214 RFM69HCW radio shield connects the RFM69HCW module to the appropriate lines on the Arduino®. By using the RF69 Library, you can send and receive messages via standard 4-wire SPI interface. The library includes command structures for setting up anything from simple non-addressed point-to-point communication to fully addressed networks of clients and routers. Frequency can be set with 61 Hz precision to any frequency from 240.0 to 960.0 MHz.

The shield has on-board power regulation and level shifting, an external antenna via the I-Pex connector and additional jumpers to set the interface and digital communication pins on the Arduino®.

Specifications

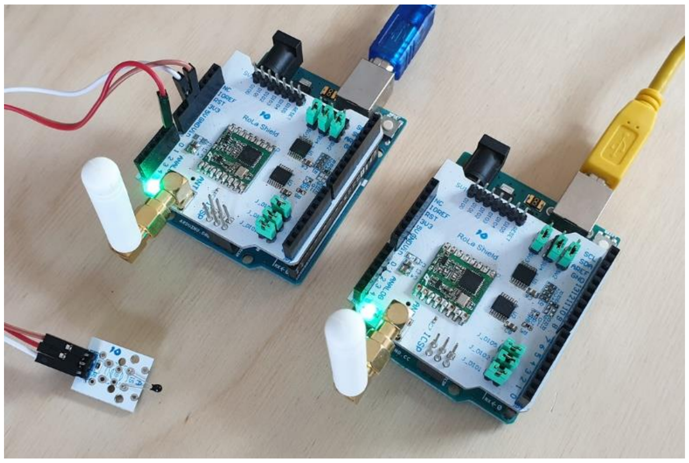
- 168 dB maximum link budget
- +20 dBm/100 mW constant RF output
- High sensitivity: down to -120 dBm @ 1.2 kbps
- High selectivity: 16-tap FIR channel filter
- Low current: Rx = 16 mA, 100 µA register retention
- Programmable Pout: -18 to +20 dBm in 1 dB step
- FSK bit rates up to 300 kb/s
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation
- Packet engine with CRC-16, AES-128, 66-byte FIF
- 115 dB+ dynamic range RSSI

Features

- Compatible with 3.3V or 5V I/O Arduino® board
- Compatible with Arduino® Leonardo, Uno, Mega and DUE development boards
- SPI interface
- Frequency band: 315, 433, 868 and 915 MHz (programmable)
- Suitable for LoRa network applications
- Low power consumption
- Included external antenna via I-Pex connector
- Built-in temperature sensor
- Built-in bit synchronizer performing clock recovery
- Incoming sync word recognition
- Automatic RF Sense and CAD with ultra-fast AFC
- Constant RF performance over voltage range of module

Operation

Overview



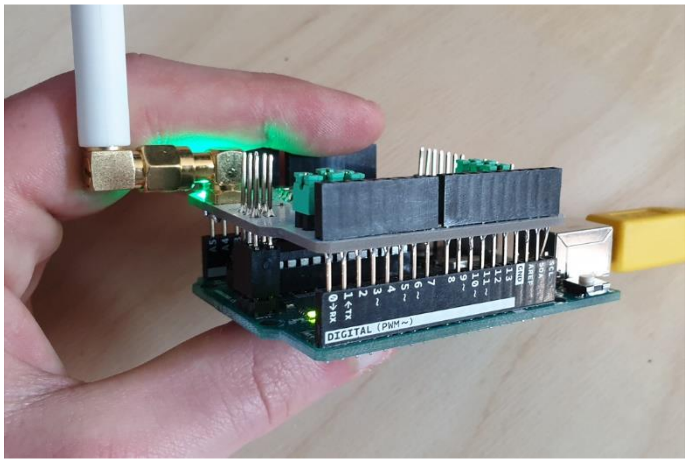
We will connect two radio shields to one another, read the temperature and send the temperature value to another module.

Requirements:

- 2x Arduino® UNO
- 2x WPSH214 RFM69HCW module
- 1x WPSE320 analogue temperature sensor module
- 2x USB-A to USB-B cable

Connecting Arduino®

1. Connect the WPSH214 RFM69HCW module to the Arduino® UNO (or Arduino® MEGA) as depicted.

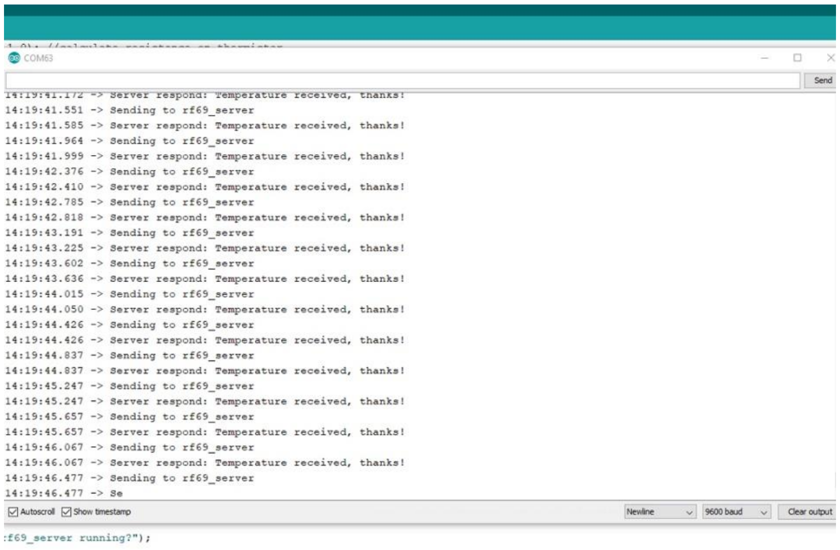


2. Connect the WPSE320 temperature module.

WPSE320		WPSH214
+	►	5 V
-	►	GND
S	►	A0

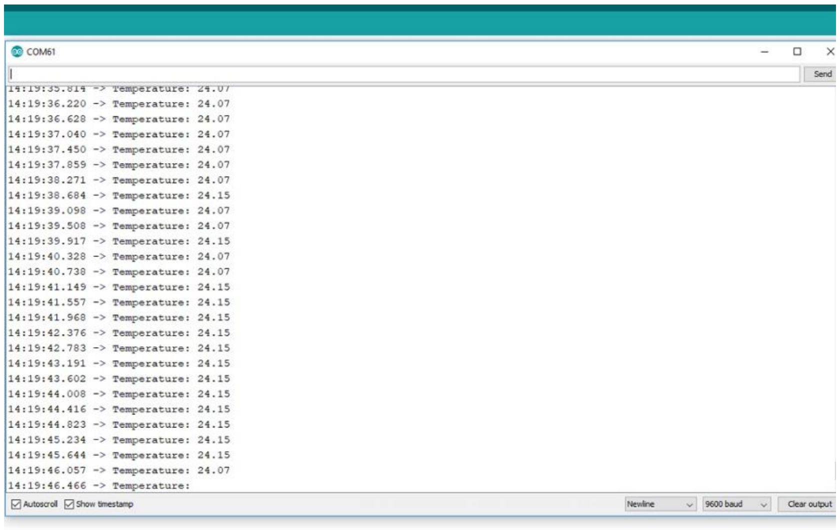
Programming

1. Download the RadioHead library here and add it to the Arduino® IDE.
2. Go to GitHub and download both codes here.
3. Open the RF69_CLIENT_SEND_DATA code and upload to the Arduino® **with** the temperature sensor. You should see following output:



```
14:19:41.172 -> Server respond: Temperature received, thanks!
14:19:41.551 -> Sending to rf69_server
14:19:41.585 -> Server respond: Temperature received, thanks!
14:19:41.964 -> Sending to rf69_server
14:19:41.999 -> Server respond: Temperature received, thanks!
14:19:42.376 -> Sending to rf69_server
14:19:42.410 -> Server respond: Temperature received, thanks!
14:19:42.785 -> Sending to rf69_server
14:19:42.818 -> Server respond: Temperature received, thanks!
14:19:43.191 -> Sending to rf69_server
14:19:43.225 -> Server respond: Temperature received, thanks!
14:19:43.602 -> Sending to rf69_server
14:19:43.636 -> Server respond: Temperature received, thanks!
14:19:44.015 -> Sending to rf69_server
14:19:44.050 -> Server respond: Temperature received, thanks!
14:19:44.426 -> Sending to rf69_server
14:19:44.426 -> Server respond: Temperature received, thanks!
14:19:44.837 -> Sending to rf69_server
14:19:44.837 -> Server respond: Temperature received, thanks!
14:19:45.247 -> Sending to rf69_server
14:19:45.247 -> Server respond: Temperature received, thanks!
14:19:45.657 -> Sending to rf69_server
14:19:45.657 -> Server respond: Temperature received, thanks!
14:19:46.067 -> Sending to rf69_server
14:19:46.067 -> Server respond: Temperature received, thanks!
14:19:46.477 -> Sending to rf69_server
14:19:46.477 -> Se
```

4. Open a new IDE with the RF69_SERVER_RECEIVE_DATA code and upload to the Arduino® **without** the temperature sensor. You should see following output:



```
14:19:35.814 -> Temperature: 24.07
14:19:36.220 -> Temperature: 24.07
14:19:36.628 -> Temperature: 24.07
14:19:37.040 -> Temperature: 24.07
14:19:37.450 -> Temperature: 24.07
14:19:37.859 -> Temperature: 24.07
14:19:38.271 -> Temperature: 24.07
14:19:38.684 -> Temperature: 24.15
14:19:39.098 -> Temperature: 24.07
14:19:39.508 -> Temperature: 24.07
14:19:39.917 -> Temperature: 24.15
14:19:40.328 -> Temperature: 24.07
14:19:40.738 -> Temperature: 24.07
14:19:41.149 -> Temperature: 24.15
14:19:41.557 -> Temperature: 24.15
14:19:41.968 -> Temperature: 24.15
14:19:42.376 -> Temperature: 24.15
14:19:42.783 -> Temperature: 24.15
14:19:43.191 -> Temperature: 24.15
14:19:43.602 -> Temperature: 24.15
14:19:44.008 -> Temperature: 24.15
14:19:44.416 -> Temperature: 24.15
14:19:44.823 -> Temperature: 24.15
14:19:45.234 -> Temperature: 24.15
14:19:45.644 -> Temperature: 24.15
14:19:46.057 -> Temperature: 24.07
14:19:46.466 -> Temperature:
```

Now, it's time to try and send data using all kinds of sensors, e.g. reading and sending the status of a PIR sensor to the Arduino® so you can make a relay switch or control LEDs...

whadda.com