Spangaard Jan '17

The library has been running on two Photons, two Arduino Pro Mini's and one Arduino Due for about a year - but has not been tested with newer firmware versions or the Electron. I never succeeded in publishing the library to the community. May bee i try again when the new IDE version is released. GitHub and C++ are not my native languages.

I have a another more specialized version of the library in the works. The "low-level" CC1101 interface (SPI) is the same, so the old library in the link should also work with firmware 6.0.

Just do as Scruff said, and let me know if I can help or you can improve the library. Especially I have not played much with the different ways on configuring the radio.

The CC1101 is a great little two-way radio. I use it where I don't have wifi (greenhouse, mailbox etc.) or for communication between Arduino boards and Particle - or when i just need a simple and fast way to move data over short distances. I mainly use the 433 version. A roundtrip (send message - receive confirmation) takes < 20 ms.You get an interrupt when a message is ready to be read and processed. You can broadcast to every CC1101 device or let the radio filter out only messages to send to one specific radio.

The only real problem, I have not found a solid solution for, is when several CC1101 radio's are used and the packets collide in the air. The CC1101 radio can check if the "air is clean", but normally I just resend, if I don't receive a reply in xx millis without caring why.

My library is very inspired / ported from the PanStamp library used in Arduino land. See http://www.panstamp.com/

Good luck 😉

Spangaard Jan '17

Hi again,

Just tried out the library. It compiles with Photon as target, but with a few warnings (not dangerous and easy to fix).

I have uploaded the library example to two Photons with CC1101 radio's connected and it works fine. The two Photons are in my basement at home, and I am not, so I have not checked the Serial output, just the Particle.publish output.

But, the library was my first time making a C++/Particle library. And now I am a year wiser - thanks to this community 😉

The very basic library/example could be improved in many ways (using pinSetFast/pinResetFast, saving setup in eeprom, better commenting, attachInterrupt in library code, more example apps etc.).

So please let me know, if I can help you getting this great extension to the Photon/Electron to work.

~Michael

---

izmevid Jan '17

Hello,

first of all I want to thank you for your very informative reply. I tried to get your example code running, first on an Arduino Nano communicating with the Particle Electron and second with two Arduino Nanos but both setups didn’t work. I think it’s a problem of connecting the sensors the right way as I didn’t find out where you defined the pins.

My CC1101 contains the following pins:

- VCC (3.3V)
- GND
- SCK
- GD0
- GD2
- CSN
- SO
- SI

Could you please inform me about the pins to connect to on the Nano and the Electron as I really don’t know which pins are necessary?

Thanks you very much,
David

ScruffR Jan '17
Elite

Since it’s an SPI device you’d normally use the hardware SPI pins (SCK, (MO)SI, (MI)SO - CSN = SS but can be any other free pin too)
https://docs.particle.io/reference/firmware/photon/#spi

---

izmevid Jan '17

I tried using the pins as described there.

So I used the following for the Electron:
SS --> A2
SCK --> A3
MISO --> A4
MOSI --> A5

And I connected the Arduino Nano this way:
SS --> 10
MOSI --> 11
SO --> 12
SCK --> 13

GD0 --> 02

Serially monitoring the Electron shows me this from the setup loop:

```
PARTNUM 0, VERSION 20, MARCSTATE 1
CC1101 initialized
Network {19, 9}, Frequency=0x433, tx-power=84, Channel=5, Device #10, remoteDevi
GDO0 pin=11SPI: Slave Select pin=12, clock divider=24
```

… and on the Arduino Nano it keeps sending me the send package messages after the following in the setup loop:

```
Added 1 to deviceAddress becourse D9 is connected to ground
PARTNUM 0, VERSION 20, MARCSTATE 1
CC1101 initialized
Network {19, 9}, Frequency=0x433, tx-power=84, Channel=5, Device #40, remoteDevi
GDO0 pin=2SPI: Slave Select pin=10, clock divider=0


Send packet #578 - data[10]={0, 40, 66, 2, 69, 70, 71, 72, 73, 74}
```

Spangaard Jan '17

Your pin connections on the Electron looks like the same I use on the Photon.

SS => A2
SCK => A3
MISO => A4
MOSI => A5
GDO0 => A1 (interrupt)
GDO1 is not used.

The SPI stuff are defined in CC1101Radio.h

I don’t have a Nano, but it seems to be “standard” Arduino, so this also looks ok.

The CC1101 is 3.3v. If the Nano is 5v on SPI, you need to level shift - and you could have damaged the CC1101 module.

In the example ino file, the TX-power is set to a low level, be course I often have the devices close together when testing. Try changing TX-power in setup().

```
> #define PA_MinDistance    0x60
> #define PA_ShortDistance  0x84
> #define PA_LongDistance   0xC8
> #define PA_MaxDistance    0xC0
```

Both serial outputs should show ‘Send packet #121 - data[10…’ every ~2 seconds like you see on the Nano. Don’t you see the same on the Electron ?

You succeed in reading PARTNUM and VERSION from the cc1101 modules. This indicates SPI is working.

What radio device do you have ? - this example only works with 433 Mhz modules. Some times cc1101 modules are sold on ebay as if they work on 433, 868 or 915 Mhz. But all mine (+20) only works at one specific frequence. Very often the 433 modules is green and 868/915 are blue.

You can change the frequence in setup()
cc1101.deviceData.carrierFreq = CFREQ_433; // CFREQ_868 //CFREQ_915 // CFREQ_433

If you change frequence, you must change on both devices.

On some of my modules only one of the two VCC and GND pins works - use the ones furthest back.

Is it possible for you to try with two Particle devices ?

If this doesn't work, I need to make similarly setup and try it out. I think an Arduino Nano is pin compatible with an Arduino Pro Mini or Micro ?

Hope this helps

~Michael

---

izmevid Jan '17

Hi Michael,

as you pointed out that the CC1101 should be connected the right way, I found out that I didn't connect the GDO0 pin to A1. Solving this then seemed to be the only issue preventing the connection from working as expected.

Thanks for all your helpful advice.
David

Spangaard Jan '17

Hi David,

Great, hope you get what you expect from this little juwel 😉

~Michael

---

izmevid Jan '17

Hi Michael,

as the purpose of this radio module is to send weather data over more or less long distances I wondered if and how it is possible to send Integer values of a higher number of bits than 8. This would make transferring the values (f.e. temperature), multiplied by 10 to get one decimal place, possible.

Is it possible to at least send values of the 16bit Integer type or even higher?

Thanks,
David

Spangaard Jan '17

You can send any datatype. I think the radio's buffer size is limited to 64 bytes pr. packet/message. In most of my code, I work with a limit of 32 bytes pr. message.

A message is always an array of bytes, where the first byte is the address of the radio device you want to send the message to ('0' is a broadcast message all radio's receive).

So all you have to do is convert your data into an array of bytes, copy the byte-array to the packet-variable and un-pack/convert your data back again on the receiving device.

I'm a C++ novice, so I use methods I understand – they may not be optimal.

Here is an example sending/receiving structured data. It should be easy to edit the example in the library.

```
struct weatherData_s {
    uint32_t temperature; // 4 bytes
    uint16_t humidity; // 2 bytes
    uint16_t airPressure; // 2 bytes
    uint32_t readingTimestamp; // 4 bytes
};

void sendPacket() {

    weatherData_s wd;

    // replace with your sensor readings
    wd.temperature = 44; //random(0,40000);
    wd.humidity = 55; //random(10,100);
    wd.airPressure = 66; //random(900,1350);
    wd.readingTimestamp = Time.now();

    CC1101Radio::CCPACKET pkt;

    pkt.length = 4 + sizeof(wd);

    pkt.data[0] = cc1101.deviceData.remoteDeviceAddress;
    pkt.data[1] = cc1101.deviceData.deviceAddress;
    pkt.data[2] = lowByte(packetNum);
    pkt.data[3] = highByte(packetNum++);

    // copy weather data to packet variable
    memcpy(pkt.data+4, &wd, sizeof(wd));



    // do the actual transmitting
    cc1101.sendData(pkt);

    packetsSend++;
```

```
    packetsSendBytes+=pkt.length;

    Serial.print("\r\nSend packet #");
    Serial.print(packetNum);
    Serial.print(" -");
    cc1101.printCCPACKETdata(&pkt);

}

bool receivePacket() {

    if(!cc1101.packetAvailable) {
        return false;
    }

    // The cc1101 has received a package
    detachIntr();
    cc1101.packetAvailable = false;

    CC1101Radio::CCPACKET pkt;

    // read the packet
    if(cc1101.receiveData(&pkt)) {

        // do we have valid packet ?
        if((pkt.crc_ok==1) && (pkt.length > 0)) {

            packetsReceived++;
            packetsReceivedBytes+=pkt.length;

            Serial.println("\r\n--------------------------------");
            Serial.print("CC1101 have news! - ");
            Serial.print("Package len = ");
            Serial.print(pkt.length);
            Serial.print(" lgi=");
            Serial.print(pkt.lqi);
            Serial.print(" rssi=");
            Serial.print(pkt.rssi);
            Serial.print(" CRC=");
            Serial.println(pkt.crc_ok);

            weatherData_s wd;
            memcpy(&wd, pkt.data+4, sizeof(wd));

            uint16_t msgNum = pkt.data[2] + (pkt.data[3]*255);

            Serial.print("Message num (");
            Serial.print(msgNum);

            Serial.print(") from device #");
            Serial.print(pkt.data[1]);
```

```
            Serial.print(", sendt to device #");
            Serial.print(pkt.data[0]);
            Serial.print(", data length: ");
            Serial.println(pkt.length-4);

            Serial.print("\tTemperature = ");
            Serial.println(wd.temperature);
            Serial.print("\tHumidity = ");
            Serial.println(wd.humidity);
            Serial.print("\tAir pressure = ");
            Serial.println(wd.airPressure);
            Serial.print("\tTimestamp = ");
            Serial.println(wd.readingTimestamp);

            Serial.println();

        } // crc & len>0

    } // cc1101.readData

    attachIntr(); // re-attach interrupt so packetAvailable will be set true in

    return true;

}
```

It produces Serial output like this:

```
--------------------------------
CC1101 have news! - Package len = 16 lgi=47 rssi=9 CRC=1
Message num (276) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372228


Send packet #213 - data[16]={0, 192, 212, 0, 44, 0, 0, 0, 55, 0, 66, 0, 69, 251,

--------------------------------
CC1101 have news! - Package len = 16 lgi=49 rssi=10 CRC=1
Message num (277) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372230


Send packet #214 - data[16]={0, 192, 213, 0, 44, 0, 0, 0, 55, 0, 66, 0, 71, 251,

--------------------------------
```

```
CC1101 have news! - Package len = 16 lgi=47 rssi=10 CRC=1
Message num (278) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372232
```

~Michael

---

izmevid Jan '17

Thanks a lot once again for this great explenation. 😀

Spangaard Jan '17

Glad to help 😉

Without the forum and examples, many of us would’t get anywhere 😄

~Michael

Spangaard Feb '16

I have kind of ported the panStamp cc1101 library. The library is working on Photon, Arduino Uno, Mega, Mini and Due. I’am using it to reach areas where I don’t have wifi , where I need a fast connection (debug) and for building cheap remote sensors (Arduino Mini 3.3v, CC1101 radio, sensors and battery).

I have no skills regarding the CC1101 radio, so i am using the panStamp standard setup, but in 433 MHz mode.

I’am on vacation, but will be happy to share my code, when I get back. But be warned - this is my first Photon and C++ project, so the coding quality is probably bad.

Let me know if you are interested.

~Michael

Edward Feb '16

Hi Michael

I’m new to Photon as well and it’s too tempting to make some interesting prototype devices to see what potential it can bring. yes, I’m interested in the library, thanks in front!

Edward.

Spangaard **Feb '16**

Hi

You can find my CC1101Radio library her: https://github.com/Gnags/CC1101Radio/tree/master

It's my first time publishing on GitHub, and my first time building a "stand-alone-library" – hope it works 😉

The CC1101RadioSniffer sketch will:

```
Send a broadcast message at 2000 ms intervals
Receive all messages on this freqency, network/sync word and channel
Print send/receive events on serial
Publish (if Photon board) at 5000 ms intervals -  How many packages send and rec
```

The output in the Serial Monitor looks like this:

CC1101#41 send package#176 - data[10]={0, 41, 176, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#40 msgId #164 - length 10 lgi=47 rssi=103 CRC=1 data[10]={0, 40, 164, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#30 msgId #173 - length 10 lgi=46 rssi=91 CRC=1 data[10]={0, 30, 173, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#10 msgId #1240 - length 10 lgi=48 rssi=46 CRC=1 data[10]={0, 10, 216, 4, 69, 70, 71, 72, 73, 74}
CC1101#41 send package#177 - data[10]={0, 41, 177, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#40 msgId #165 - length 10 lgi=47 rssi=103 CRC=1 data[10]={0, 40, 165, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#30 msgId #174 - length 10 lgi=46 rssi=91 CRC=1 data[10]={0, 30, 174, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#10 msgId #1241 - length 10 lgi=44 rssi=46 CRC=1 data[10]={0, 10, 217, 4, 69, 70, 71, 72, 73, 74}
CC1101#41 send package#178 - data[10]={0, 41, 178, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#40 msgId #166 - length 10 lgi=48 rssi=103 CRC=1 data[10]={0, 40, 166, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#30 msgId #175 - length 10 lgi=47 rssi=92 CRC=1 data[10]={0, 30, 175, 0, 69, 70, 71, 72, 73, 74}
CC1101#41 got a packet from CC1101#10 msgId #1242 - length 10 lgi=45 rssi=41 CRC=1 data[10]={0, 10, 218, 4, 69, 70, 71, 72, 73, 74}
CC1101#41 send package#179 - data[10]={0, 41, 179, 0, 69, 70, 71, 72, 73, 74}

I am looking at the MCU connected to CC1101 with address 41. It is receiving packets from 3 other devices (30, 40 & 10). I get similar outputs from the other 3 CC1101/MCU devices I have running.

In the Particle Dash board I can see how many packets the Photon board has send and received.

I think the CC1101 is a great little device. Don't use much power, ~300 meters line of sight, RX/TX buffer, small, affordable etc. Hope you get it working.

Please let me know if this is useful or how I can improve the example (the library needs some work

before its ready for review).

~Michael

Edward Feb '16

Hi Michael

I'll try this out after I receiving the RF module, thank you for sharing this !

Edward

---

izmevid Jan '17

Hi Michael,

I was searching for a CC1101 library to use with my Electron. Now that your library showed up I wondered if it's already possible to import this library to the Web IDE. I tried to import your Github directory but it unfortunately didn't work.

How did you include this library to your projects? Did you contribute this library to the community libraries and if so which name does it have?

Appreciate your reply,

David

ScruffR Jan '17
Elite

You can add extra file tabs in Build (`(+)`) icon top right and then copy-paste the contents of the `.h`/`.cpp` files in the repo to the respective file tabs.

If you then get any error messages, just post them here.

433Mhz Library [COMPLETED]

---

izmevid Jan '17

Thanks a lot for your quick reply.

I previously thought that I can only use libraries when importing them with the libraries tool in the IDE. So is it also possible to use any Arduino library code?

Thanks,
David

ScruffR Jan '17
Elite

You can use any code as long it’s compatible 😉
But that library you linked is said to be compatible, so it should work the way I said.

Spangaard Jan '17

The library has been running on two Photons, two Arduino Pro Mini’s and one Arduino Due for about a year - but has not been tested with newer firmware versions or the Electron. I never succeeded in publishing the library to the community. May bee i try again when the new IDE version is released. GitHub and C++ are not my native languages.

I have a another more specialized version of the library in the works. The “low-level” CC1101 interface (SPI) is the same, so the old library in the link should also work with firmware 6.0.

Just do as Scruff said, and let me know if I can help or you can improve the library. Especially I have not played much with the different ways on configuring the radio.

The CC1101 is a great little two-way radio. I use it where I don’t have wifi (greenhouse, mailbox etc.) or for communication between Arduino boards and Particle - or when i just need a simple and fast way to move data over short distances. I mainly use the 433 version. A roundtrip (send message - receive confirmation) takes < 20 ms.You get an interrupt when a message is ready to be read and processed. You can broadcast to every CC1101 device or let the radio filter out only messages to send to one specific radio.

The only real problem, I have not found a solid solution for, is when several CC1101 radio’s are used and the packets collide in the air. The CC1101 radio can check if the “air is clean”, but normally I just resend, if I don’t receive a reply in xx millis without caring why.

My library is very inspired / ported from the PanStamp library used in Arduino land. See http://www.panstamp.com/

Good luck 😉

Spangaard Jan '17

Hi again,

Just tried out the library. It compiles with Photon as target, but with a few warnings (not dangerous and easy to fix).

I have uploaded the library example to two Photons with CC1101 radio’s connected and it works fine. The two Photons are in my basement at home, and I am not, so I have not checked the Serial output, just the Particle.publish output.

But, the library was my first time making a C++/Particle library. And now I am a year wiser - thanks to this community 😉

The very basic library/example could be improved in many ways (using pinSetFast/pinResetFast, saving setup in eeprom, better commenting, attachInterrupt in library code, more example apps etc.).

So please let me know, if I can help you getting this great extension to the Photon/Electron to work.

~Michael

izmevid Jan '17

Hello,

first of all I want to thank you for your very informative reply. I tried to get your example code running, first on an Arduino Nano communicating with the Particle Electron and second with two Arduino Nanos but both setups didn’t work. I think it’s a problem of connecting the sensors the right way as I didn’t find out where you defined the pins.

My CC1101 contains the following pins:

- VCC (3.3V)
- GND
- SCK
- GD0
- GD2
- CSN
- SO
- SI

Could you please inform me about the pins to connect to on the Nano and the Electron as I really don’t know which pins are necessary?

Thanks you very much,
David

ScruffR Jan '17
Elite

Since it’s an SPI device you’d normally use the hardware SPI pins (SCK, (MO)SI, (MI)SO - CSN = SS but can be any other free pin too)
https://docs.particle.io/reference/firmware/photon/#spi

---

izmevid Jan '17

I tried using the pins as described there.

So I used the following for the Electron:
SS --> A2
SCK --> A3
MISO --> A4
MOSI --> A5

And I connected the Arduino Nano this way:
SS --> 10
MOSI --> 11
SO --> 12
SCK --> 13
GD0 --> 02

Serially monitoring the Electron shows me this from the setup loop:

```
PARTNUM 0, VERSION 20, MARCSTATE 1
CC1101 initialized
Network {19, 9}, Frequency=0x433, tx-power=84, Channel=5, Device #10, remoteDevi
GDO0 pin=11SPI: Slave Select pin=12, clock divider=24
```

… and on the Arduino Nano it keeps sending me the send package messages after the following in the setup loop:

```
Added 1 to deviceAddress becourse D9 is connected to ground
PARTNUM 0, VERSION 20, MARCSTATE 1
CC1101 initialized
Network {19, 9}, Frequency=0x433, tx-power=84, Channel=5, Device #40, remoteDevi
GDO0 pin=2SPI: Slave Select pin=10, clock divider=0


Send packet #578 - data[10]={0, 40, 66, 2, 69, 70, 71, 72, 73, 74}
```

Spangaard Jan '17

Your pin connections on the Electron looks like the same I use on the Photon.

SS => A2
SCK => A3
MISO => A4
MOSI => A5
GDO0 => A1 (interrupt)
GDO1 is not used.

The SPI stuff are defined in CC1101Radio.h

I don’t have a Nano, but it seems to be “standard” Arduino, so this also looks ok.

The CC1101 is 3.3v. If the Nano is 5v on SPI, you need to level shift - and you could have damaged the CC1101 module.

In the example ino file, the TX-power is set to a low level, be course I often have the devices close together when testing. Try changing TX-power in setup().

```
> #define PA_MinDistance    0x60
> #define PA_ShortDistance  0x84
> #define PA_LongDistance   0xC8
> #define PA_MaxDistance    0xC0
```

Both serial outputs should show ‘Send packet #121 - data[10…’ every ~2 seconds like you see on the Nano. Don’t you see the same on the Electron ?

You succeed in reading PARTNUM and VERSION from the cc1101 modules. This indicates SPI is working.

What radio device do you have ? - this example only works with 433 Mhz modules. Some times

cc1101 modules are sold on ebay as if they work on 433, 868 or 915 Mhz. But all mine (+20) only works at one specific frequence. Very often the 433 modules is green and 868/915 are blue.

You can change the frequence in setup()
cc1101.deviceData.carrierFreq = CFREQ_433; // CFREQ_868 //CFREQ_915 // CFREQ_433

If you change frequence, you must change on both devices.

On some of my modules only one of the two VCC and GND pins works - use the ones furthest back.

Is it possible for you to try with two Particle devices ?

If this doesn't work, I need to make similarly setup and try it out. I think an Arduino Nano is pin compatible with an Arduino Pro Mini or Micro ?

Hope this helps

~Michael

---

izmevid Jan '17

Hi Michael,

as you pointed out that the CC1101 should be connected the right way, I found out that I didn't connect the GDO0 pin to A1. Solving this then seemed to be the only issue preventing the connection from working as expected.

Thanks for all your helpful advice.
David

Spangaard Jan '17

Hi David,

Great, hope you get what you expect from this little juwel 😉

~Michael

---

izmevid Jan '17

Hi Michael,

as the purpose of this radio module is to send weather data over more or less long distances I wondered if and how it is possible to send Integer values of a higher number of bits than 8. This would make transferring the values (f.e. temperature), multiplied by 10 to get one decimal place, possible.

Is it possible to at least send values of the 16bit Integer type or even higher?

Thanks,
David

Spangaard Jan '17

You can send any datatype. I think the radio's buffer size is limited to 64 bytes pr. packet/message. In most of my code, I work with a limit of 32 bytes pr. message.

A message is always an array of bytes, where the first byte is the address of the radio device you want to send the message to ('0' is a broadcast message all radio's receive).

So all you have to do is convert your data into an array of bytes, copy the byte-array to the packet-variable and un-pack/convert your data back again on the receiving device.

I'm a C++ novice, so I use methods I understand – they may not be optimal.

Here is an example sending/receiving structured data. It should be easy to edit the example in the library.

```
struct weatherData_s {
    uint32_t temperature; // 4 bytes
    uint16_t humidity; // 2 bytes
    uint16_t airPressure; // 2 bytes
    uint32_t readingTimestamp; // 4 bytes
};

void sendPacket() {

    weatherData_s wd;

    // replace with your sensor readings
    wd.temperature = 44; //random(0,40000);
    wd.humidity = 55; //random(10,100);
    wd.airPressure = 66; //random(900,1350);
    wd.readingTimestamp = Time.now();

    CC1101Radio::CCPACKET pkt;

    pkt.length = 4 + sizeof(wd);

    pkt.data[0] = cc1101.deviceData.remoteDeviceAddress;
    pkt.data[1] = cc1101.deviceData.deviceAddress;
    pkt.data[2] = lowByte(packetNum);
    pkt.data[3] = highByte(packetNum++);

    // copy weather data to packet variable
    memcpy(pkt.data+4, &wd, sizeof(wd));



    // do the actual transmitting
    cc1101.sendData(pkt);

    packetsSend++;
    packetsSendBytes+=pkt.length;
```

```
    Serial.print("\r\nSend packet #");
    Serial.print(packetNum);
    Serial.print(" -");
    cc1101.printCCPACKETdata(&pkt);

}

bool receivePacket() {

    if(!cc1101.packetAvailable) {
        return false;
    }

    // The cc1101 has received a package
    detachIntr();
    cc1101.packetAvailable = false;

    CC1101Radio::CCPACKET pkt;

    // read the packet
    if(cc1101.receiveData(&pkt)) {

        // do we have valid packet ?
        if((pkt.crc_ok==1) && (pkt.length > 0)) {

            packetsReceived++;
            packetsReceivedBytes+=pkt.length;

            Serial.println("\r\n--------------------------------");
            Serial.print("CC1101 have news! - ");
            Serial.print("Package len = ");
            Serial.print(pkt.length);
            Serial.print(" lgi=");
            Serial.print(pkt.lqi);
            Serial.print(" rssi=");
            Serial.print(pkt.rssi);
            Serial.print(" CRC=");
            Serial.println(pkt.crc_ok);

            weatherData_s wd;
            memcpy(&wd, pkt.data+4, sizeof(wd));

            uint16_t msgNum = pkt.data[2] + (pkt.data[3]*255);

            Serial.print("Message num (");
            Serial.print(msgNum);

            Serial.print(") from device #");
            Serial.print(pkt.data[1]);
            Serial.print(", sendt to device #");
```

```
            Serial.print(pkt.data[0]);
            Serial.print(", data length: ");
            Serial.println(pkt.length-4);

            Serial.print("\tTemperature = ");
            Serial.println(wd.temperature);
            Serial.print("\tHumidity = ");
            Serial.println(wd.humidity);
            Serial.print("\tAir pressure = ");
            Serial.println(wd.airPressure);
            Serial.print("\tTimestamp = ");
            Serial.println(wd.readingTimestamp);

            Serial.println();

        } // crc & len>0

    } // cc1101.readData

    attachIntr(); // re-attach interrupt so packetAvailable will be set true in

    return true;

}
```

It produces Serial output like this:

```
--------------------------------
CC1101 have news! - Package len = 16 lgi=47 rssi=9 CRC=1
Message num (276) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372228


Send packet #213 - data[16]={0, 192, 212, 0, 44, 0, 0, 0, 55, 0, 66, 0, 69, 251,

--------------------------------
CC1101 have news! - Package len = 16 lgi=49 rssi=10 CRC=1
Message num (277) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372230


Send packet #214 - data[16]={0, 192, 213, 0, 44, 0, 0, 0, 55, 0, 66, 0, 71, 251,

--------------------------------
CC1101 have news! - Package len = 16 lgi=47 rssi=10 CRC=1
```

```
Message num (278) from device #192, sendt to device #0, data length: 12
    Temperature = 44
    Humidity = 55
    Air pressure = 66
    Timestamp = 1485372232
```

~Michael

---

izmevid Jan '17

Thanks a lot once again for this great explenation. 😀

Spangaard Jan '17

Glad to help 😉

Without the forum and examples, many of us would’t get anywhere 😄

~Michael