Links:

Dokumentet er ikke færdigt endnu!!

Trådløs HC-12-UART

HC-12 kan bruges som en Wireless serial Port
Den kobles direkte på en UART, Arduinos pin 0 og 1 – hvis der ikke skal bruges USB-kommunikation, - eller på en SoftSerial port.

Den er bidirektionel, dvs. den kan både bruges som sender og som modtager. Dog ikke samtidig!! Kaldes Half Duplex.

Den har en rækkevidde på op til flere hundrede meter. Nogle steder er nævnt op til 2 km.

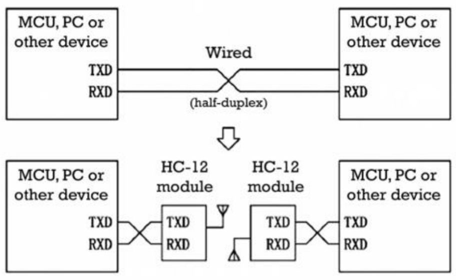
Der medfølger en lille spole-antenne, der skal loddes i. Men der kan også tilsluttes en ekstern SMA antenne via et lille stik.

Default er den sat op til en Baud-rate på 9600 bps, men der er mulighed for at ændre dette til gængse Baudrates.

HC-12 sender på ISM-båndet, - på 433 MHz. Her sendes korte Bursts, - hver gang der sendes en Byte, en pakke a´8 bit.

2 HC-12 moduler kan erstatte ledninger mellem 2 uC-er, som vist.

Sender og modtager skal iflg. datablade være mindst 1,5 meter fra hinanden for at de virker!!



HC-12 koster ca. 120 kr., - men fås meget billigere i Kinesien.

- Driftsspænding: 3.2V~5.5V
- Default address range (open field test): About 600m (maximum communication distance adjustable reach 1000m, the baud rate is 5000bps)
- Default idle current: 16MA (In different working modes operating current is different)
- Operating frequency range: 433.4-473.0MHz, up to 100 channels of communication
- The maximum transmit power: 100mW (settable)
- Default factory settings: Mode FU3, baud rate is 9600bps, communication channels CH001 (433.4M)
- Model: HC-12 433 SI4463

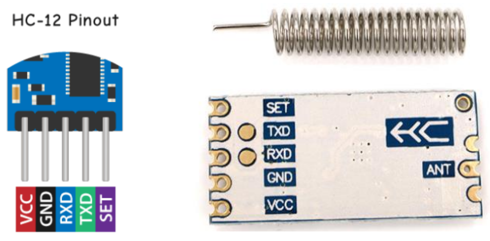
HC-12 virker ikke sammen med forgængeren, HC-11.

Tilslutning til Arduino

På boardet er der en microcontroller, der tager sig af data fra/til Arduinoen, og styrer selve radiosenderen – modtageren, dvs. transceiveren.

HC-12 har 5 pins, Vcc, Gnd, Tx, Rx, og den sidste kan bruges til at indstille den til andre operatingmodes end default.

VCC og GND er hhv. 5 Volt og 0 Volt. Der bør monteres en kondensator på 22 uF eller mere mellem + og Gnd.



TxD sender data til den tilkoblede Arduinos RxD.

RxD modtager data fra Arduino og sender dem

SoftwareSerial HC12(10, 11); // HC-12 TX Pin, HC-12 RX Pin

Og SET bruges kun hvis man vil ændre på indstillingerne Kan være Not Connected, eller +5 Volt.

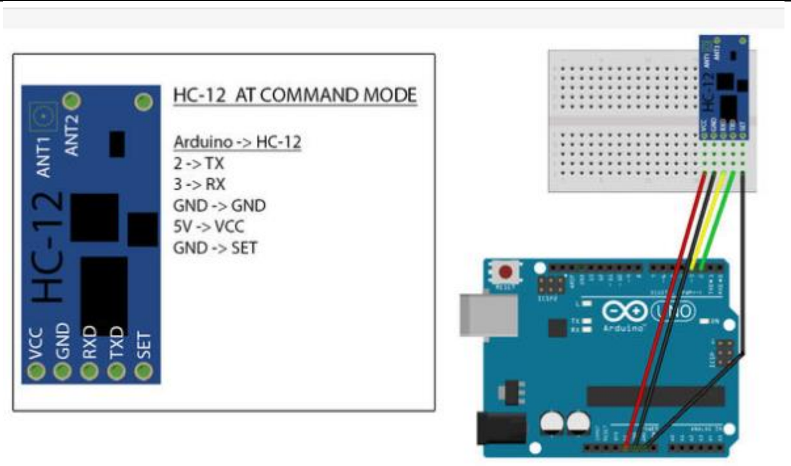
Strømforbrug i Bursts er 200 mA. Dvs. den helst skal forsynes fra en anden enhed end Arduinoen.

Nogle steder tilrådes at montere en 1N4007 diode i serie med dens Ucc-pin hvis den forsynes med mere end 4,5 Volt. Men det har jeg nu aldrig gjort!!

Arduinoen har indbygget en seriel buffer på 64 byte. Modtages flere bytes, før de læses, tabes de.

Der kan bruges flere modtagere / sendere i et system, blot de arbejder på samme Baudrate og ikke sender samtidigt.

På Fumlebrædt:

Tilslutning til Arduino Uno. Bemærk, Set kan udelades, skal blot svæve.	 HC-12 AT COMMAND MODE Arduino -> HC-12 2 -> TX 3 -> RX GND -> GND 5V -> VCC GND -> SET ANT1 ANT2 HC-12 VCC GND RXD TXD SET UNO Arduino

Kodeeksempler

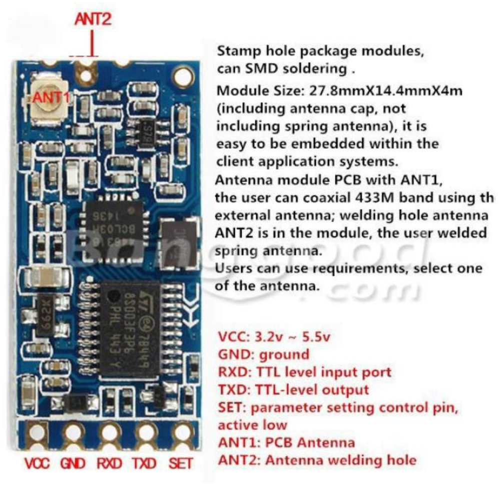
```
// Kodeeksempler

/*  HC12 Send/Receive Example Program 1
    By Mark J. Hughes
    for AllAboutCircuits.com

    Connect HC12 "RXD" pin to Arduino Digital Pin 4
    Connect HC12 "TXD" pin to Arduino Digital Pin 5
    Connect HC12 "Set" pin to Arduino Digital Pin 6

    Do not power over USB.  Per datasheet,
    power HC12 with a supply of at least 100 mA with
    a 22 uF - 1000 uF reservoir capacitor.
    Upload code to two Arduinos connected to two computers.

    Transceivers must be at least several meters apart to work.

 */
#include <SoftwareSerial.h>

const byte HC12RxdPin = 4;              // Recieve Pin on HC12
const byte HC12TxdPin = 5;              // Transmit Pin on HC12

SoftwareSerial HC12(HC12TxdPin,HC12RxdPin); // Create Software Serial Port

void setup() {
  Serial.begin(9600);                   // Open serial port to computer
  HC12.begin(9600);                     // Open serial port to HC12
}
```

```
void loop() {
 if(HC12.available()){            // If Arduino's HC12 rx buffer has data
  Serial.write(HC12.read());        // Send the data to the computer
  }
 if(Serial.available()){           // If Arduino's computer rx buffer has data
  HC12.write(Serial.read());        // Send that data to serial
 }
}
// Fra <https://www.allaboutcircuits.com/projects/understanding-and-implementing-the-hc-12-wireless-transceiver-module/>
```

Kodeeksempel # 2

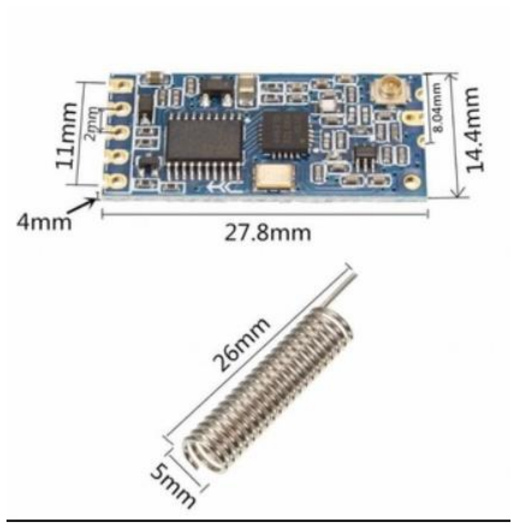
```
/* Arduino Long Range Wireless Communication using HC-12
Example 01
by Dejan Nedelkovski, www.HowToMechatronics.com
*/

#include <SoftwareSerial.h>

SoftwareSerial HC12(10, 11); // HC-12 TX Pin, HC-12 RX Pin

void setup() {
Serial.begin(9600); // Serial port to computer
HC12.begin(9600); // Serial port to HC12

}

void loop() {
while (HC12.available()) { // If HC-12 has data
Serial.write(HC12.read()); // Send the data to Serial monitor
}
while (Serial.available()) { // If Serial monitor has data
HC12.write(Serial.read()); // Send that data to HC-12
}
}
```

<u>**Spring antenna or SMA antenna**</u>

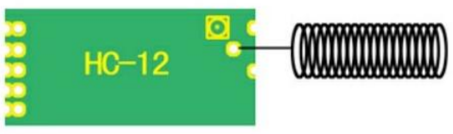


Stamp hole package modules, can SMD soldering .
Module Size: 27.8mmX14.4mmX4m (including antenna cap, not including spring antenna), it is easy to be embedded within the client application systems.
Antenna module PCB with ANT1, the user can coaxial 433M band using th external antenna; welding hole antenna ANT2 is in the module, the user welded spring antenna.
Users can use requirements, select one of the antenna.

VCC: 3.2v ~ 5.5v
GND: ground
RXD: TTL level input port
TXD: TTL-level output
SET: parameter setting control pin, active low
ANT1: PCB Antenna
ANT2: Antenna welding hole



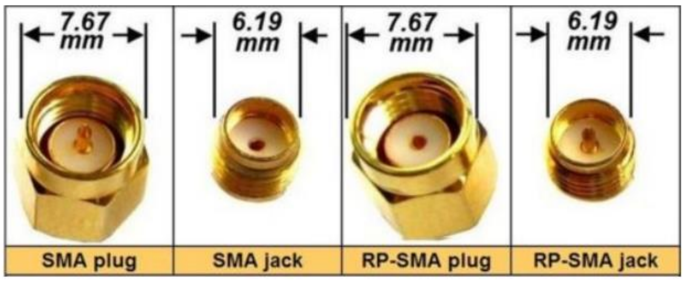
Med til HC-12 følger en spole-antenne til at lodde i boardet. Men der kan også monteres en ekstern antenne.
På printet er der et stik, der passer.



Electronic circuitry can interfere with an antenna and therefore the HC-12 has a IPEX RF socket so you can separate the antenna from the board.

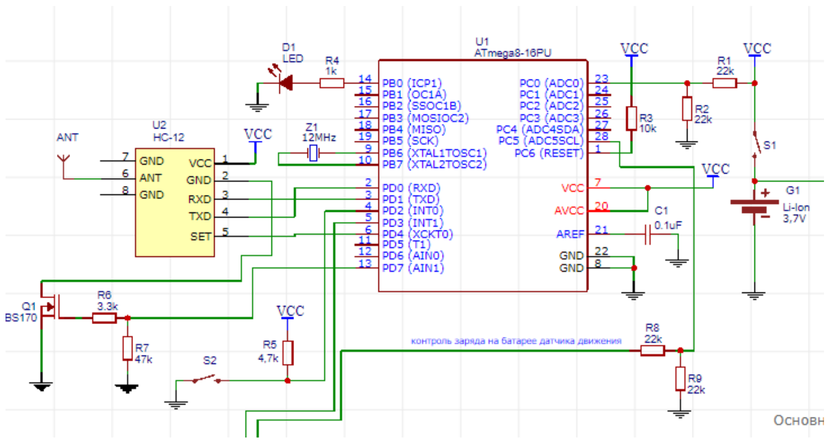
What you need is a IPEX to SMA extension cord and an SMA antenna.

You can push the cord on the IPEX connector and solder it on. On the other site of the cord you can screw on the SMA antenna.



IPX IPEX to SMA Female Jack U.FL RF Pigtail Antenna Cable Fra [ArduinoTech]()

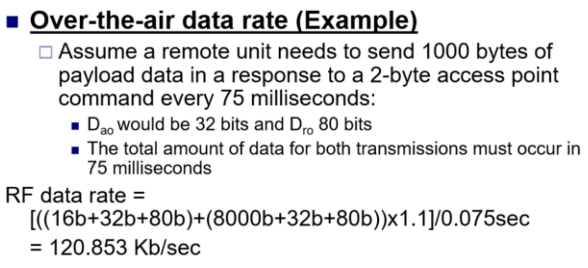
Diagram taget fra EasyEDA.



Modulation

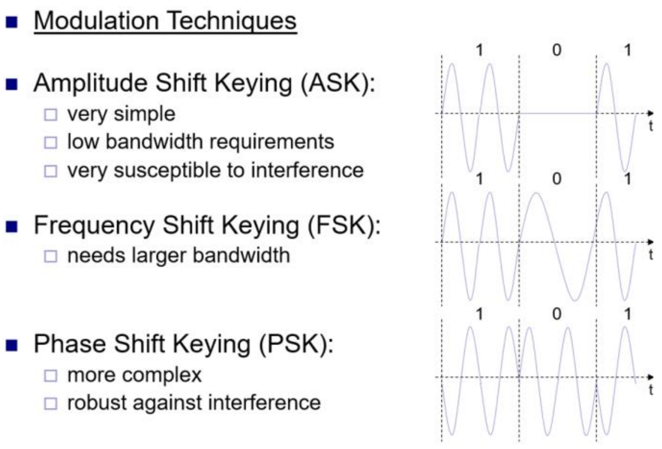
HC-12 433MHz FSK RF

The 434Mhz Serial RF Module HC-11 (1-40M) serial RF module is a low cost, high performance transparent FSK transceiver with operating at 434 MHz



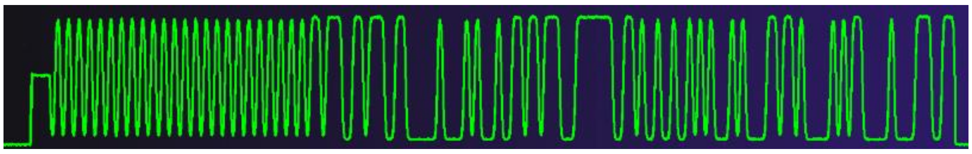
- **Over-the-air data rate (Example)**
  - Assume a remote unit needs to send 1000 bytes of payload data in a response to a 2-byte access point command every 75 milliseconds:
    - D_{ao} would be 32 bits and D_{ro} 80 bits
    - The total amount of data for both transmissions must occur in 75 milliseconds

RF data rate =
[((16b+32b+80b)+(8000b+32b+80b))x1.1]/0.075sec
= 120.853 Kb/sec



- Modulation Techniques
- Amplitude Shift Keying (ASK):
  - very simple
  - low bandwidth requirements
  - very susceptible to interference
- Frequency Shift Keying (FSK):
  - needs larger bandwidth
- Phase Shift Keying (PSK):
  - more complex
  - robust against interference

http://www.dartmouth.edu/~engs112/presentations/RF-for-WirelessComm-Jose.ppt



Om IC-erne på boardet

The Si4463 Transceiver

The Si4463 provides the wireless communication in this circuit. It has a maximum transmit power of 20 dBm (100 mW) and receive sensitivity of -129 dBm. Two 64-byte Rx and Tx FIFO memories are built into the chip along with a great many advanced features that are not implemented in the HC-12 design. See the datasheet for more information on multiband operation, frequency hopping, etc.

The STM8S003FS Microcontroller

This is an 8-bit microcontroller with 8 kB of flash memory, 128 bytes of EEPROM, and a 10-bit ADC. It supports UART, SPI, and I²C and has multiple I/O pins. It offers many of the same capabilities as its ATMega and XMC counterparts. It is programmed to control the Si4463 as well as handle the UART communication between the HC-12 and whatever it is connected to on the other end.

The HC-12 Transceiver Module

Combined with other components, the Si4463 and STM8S003 create the HC-12 transceiver, which provides a 4-pin TTL-level UART interface (Vcc, Gnd, Tx, Rx), with a 5th pin that is used to enter "command" mode for changing the module's configuration. The HC-12 has 100 supported channels spaced 400 kHz apart, eight transmit levels, eight supported baud rates, and three different working modes.

The 5th pin on the HC-12 is labeled "Set" and, when driven to logic low, allows various settings to be selected on the HC-12 using AT commands sent to the "RXD" pin.

The default configuration of the HC-12 is FU3—on Channel 1, FU3 is a fully automatic and transparent (to other devices) setting that adapts to the transmission rate of the connected device (although 9600 baud is still required to program it in Command mode).

Note that as the transmission rate increases, the sensitivity of the receiver decreases. You can return to the default state by sending AT+DEFAULT once in command mode.

Mulige Baudrates

Serial Port Baud Rate	Over-the-Air Baud Rate	Receiver Sensitivity
1200 bps	5000 bps	-117 dBm
2400 bps	5000 bps	-117 dBm
4800 bps	15000 bps	-112 dBm
9600 bps	15000 bps	-112 dBm
19200 bps	58000 bps	-107 dBm
38400 bps	58000 bps	-107 dBm
57600 bps	236000 bps	-100 dBm
115200 bps	236000 bps	-100 dBm

Features
IEEE 802.15.4g compliant
Frequency range = 119–1050 MHz
Receive sensitivity = –126 dBm
Serial TTL (RX, TX, GND) interface
Modulation (G)FSK, 4(G)FSK, (G)MSK, OOK
Max output power +20 dBm
Low active power consumption 10/13 mA RX, 18 mA TX at +10 dBm
Ultra low current powerdown modes 30 nA shutdown, 50 nA standby

Data rate = 100 bps to 1 Mbps
Fast wake and hop times
Highly configurable packet handler TX and RX 64 byte FIFOs
Auto frequency control (AFC)
Automatic gain control (AGC)
Low battery detector
FCC Part 90 Mask D, FCC part 15.247, 15, 231, 15, 249, ARIB T-108, T-96, T-67, RCR STD-30
ETSI Class-I Operation with SAW

Fra <http://lovelyelectronics.com/search/low-frequency>

Advanced setup

HC-12 indeholder en uC, der kan programmeres vha. såkaldte ”AT”-kommandoer.

Når HC-12 er forbundet til Uno-ens UART, eller Softserial, kan man – ligesom man vil sende data til radiosending – sende Setup-kommandoer til den. Det kræver blot, at dens Set-pin er gjort lav.

Vha. af AT-kommandoerne kan man ændre indstillinger, Baudrate, Sendekanal mm, - men man kan også tjekke om kommunikationen til HC-12 virker. Ligeledes kan man bede om at få udskrevet hvordan HC-12 er indstillet.

AT-kommandoer

De følgende bør sammenskrives engang !!

Changing the baud rate

Type “AT+Bxxxx”.

The baud rate can be set to 1200bps, 2400bps, 4800bps, 9600bps, 19,200bps, 38,400bps, 57,600bps, or 115,200bps. The default value is 9600bps.

Example: type “AT+B4800”. The module returns “OK+B4800”.

Changing the communication channel

Type “AT+Cxxx”.

The value can be a number from 001 to 127.

Every number is a 400KHz step. The working frequency of channel 100 is 473.0MHz.

Example: type “AT+C021”. The module returns “OK+C021”.

The module is now set to a working frequency of 441.4MHx

Note that both sending and receiving modules need to have the same frequency to communicate.

Changing the working mode of the module.

This can be FU1, FU2, FU4 or FU4 (FU4 at a baud rate of 1200 sets the chip to transmit up to 1800 meter in open air). See documentation for a full explanation.

Example: Type “AT+FU4”. The module returns “OK+FU4”.

Obtain all parameters from the module.

Type “AT+RX”.

The module should return something like this:

“OK+FU3

OK+B9600

OK+C001

OK+RP:+20dBm”.

```
// https://forum.arduino.cc/index.php?topic=355783.0

// setup of the hc12 module
 digitalWrite(7,LOW); // enter AT command mode, SET low.
 hc12.print(F("AT+DEFAULT\r\n")); // 9600, CH1, FU3, (F) to bypass flash memory
 delay(100);
 digitalWrite(7,HIGH); // enter transparent mode
```

HC-12 Wireless Transceiver Modules 433Mhz – 1000 Meters

HC-12 Wireless Transceiver Modules 433Mhz are wireless serial port communication modules, It is based on SI4463 RF chip, it has built in microcontroller, and can be configured using **AT commands**, Maximum output power is 100mW (20dBm) and receiver sensitivity differs from -117dBm to -100dBm, depending on transmission speed. It accepts 3.2V-5.5V and can be used with 3.3V and 5V UART voltage devices (3.3V safe).

Each *HC-12* can work in one of following modes:

1. FU1 – moderate power saving mode with 250000bps “over the air” baud rate. Serial port baud rate can be set to any supported value
2. FU2 – extreme power saving mode with 250000bps “over the air” speed. Serial port rate is limited to 1200bps, 2400bps, 4800bps
3. FU3 – default, general purpose mode. “Over the air” speed differs depending on serial port speed. The same goes for maximum range:
    - 1200bps ~ 1000m
    - 2400bps ~ 1000m
    - 4800bps ~ 500m
    - 9600bps ~ 500m
    - 19200bps ~ 250m
    - 38400bps ~ 250m
    - 57600bps ~ 100m
    - 115200bps ~ 100m
4. FU4 (available in version 2.3 or newer) – long range mode. “Over the air” speed is limited to 500bps and serial port speed to 1200bps. Because air speed is lower than port speed, only small packets can be send: max 60 bytes with interval of 2 seconds. In this mode range is increased to 1800m.

Pair of HC-12 that creates a wireless link has to work in the same mode (FU1, FU2, FU3, FU4) and with the same speed.

Configuration

HC-12 can be configured using AT command. The best way to do it, is to use USB-to-serial converter like CP2102. To put HC-12 into AT mode, pull *SET* pin to GND like this:

Most important commands:

1. `AT` – test command. It will return `OK` if AT interface is enabled

2. `AT+Bxxxx` – set serial port baud rate. For example, `AT+B57600` set baud rate to 57600bps
3. `AT+Cxxx` – set radio channel. Channels start from `001` at 433,4MHz. Each next channel adds 400kHz. Channel `100` is 473,0MHz. `AT+C002`will set frequency to 433,8MHz. Two HC-12 devices that creates a wireless link have to operate on the same frequency
4. `AT+FUx` – set device mode: FU1, FU2, FU3 or FU4. Two HC-12 devices that creates a wireless link have to use the same mode
5. `AT+Px` – set device transmitting power. For example `AT+P2` sets power to 2dBm (1.6mW)
    1. -1dBm (0.8mW)
    2. 2dBm (1.6mW)
    3. 5dBm (3.2mw)
    4. 8dBm (6.3mW)
    5. 11dBm (12mW)
    6. 14dBm (25mW)
    7. 17dBm (50mW)
    8. 20dBm (100mW)
6. `AT+RX` – retrieve all parameters: mode, channel, baud rate, power
7. `AT+V` – retrieve module version
8. `AT+DEFAULT` – reset module parameters to default settings

Kilde: https://www.makerlab-electronics.com/product/433mhz-serial-rf-module-hc-12-1000m/

the **FU4** mode has a 1 sec (1000 ms) delay. Try instead the factory default full speed **FU3** mode which has delay of just 4-80 ms.

Fra: https://picaxeforum.co.uk/threads/hc-11-and-hc-12-transceiver-modules.28893/

First in order to Program Connect Module “SET” Pin to GND and then power on Module
Make sure you are in 9600 BAUD rate and ‘Carriage Return’

Use These Commands:
'AT+RP' – read power
'AT+RC' read current channel

check by writing 'AT' make sure you get 'OK' if you got this so you are OK!
'AT+C001' – This Choose Channel1
'AT+P1' – Lowest Power out (0.8mW)
Read Power: 'AT+RC' you will get 'RP:-01dBm'
'AT+P8' – Highset power 100mW (reading you will get 'OK+RP:+20dBm')
'AT+B19200' – Set Baud rate to 19200
'AT+RB' – read out Baud rate, You will get 'OK+B19200'
'AT+V' – read SW Version on Mdule, My Module gives back 'HC-12_V2.3'
'AT+DEFAULT' get everything to Default Mode
'AT+Udps' – This Set UART as followos: Set data bits (d), parity (p), and stop bits (s) for serial port communication. For parity, N means none, O means odd check, and E means even check. For stop bits, 1 means one stop bit, 2 means two stop bits, and 3 means 1.5 stop bits

http://www.electronicsfreak.net/hc-12-module-testing-code/

Kodeeksempel på AT-kommandoer

```
/*
 * All codes by Milan Karakas, http://wildlab.org
 * So far, I use library "SoftwareSerial.h", but later it will be changed
 * with manual UART protocol to allow me various tricks and more reliable
 * work. This code(s) are protected by fictive "beerware union", where you may
 * use all codes, change it, copy, sell, exchange, etc. In other words, no
 * any copyright, but no waranty that it will work properly as well, so as-is.
 */

#include <SoftwareSerial.h>

SoftwareSerial mySerial(4, 5); // RX, TX
//Remember that TX of the HC-12 goes to RX of the Arduino board,
//and RX of the HC-12 goes to the TX of the same board!
float volt;
int P1, P2, P3, P4;
byte packet[20];
int j;

void setup()
{
  pinMode(A6, INPUT); //just a reminder - it is input by default
  pinMode(6, OUTPUT); //set mode AT command, connected to RX of the HC-12 module
  pinMode(13, OUTPUT);//debug LED onboard Arduino nano or Arduino pro mini
  Serial.begin(9600);
  mySerial.begin(9600);
  analogReference(EXTERNAL); //not used exactly for displaying status of the local battery.
                          //it is planed to use with resistive voltage divider 2:1
                          //and 3.3 V input on "REF" pin on Arduino nano, but on
                          //Arduino pro mini, such pin does not exist, or at least
                          //it is not on pin header - require delicate soldering
                          //on board... If used "default" option, then pay attention
                          //to 5V power supply - if in error (for example 4.8V instead 5V),
                          //then voltage readings may be wrong as well
  while (!mySerial);
  digitalWrite(6,0);
  delay(280);
  mySerial.print("AT+B9600");
  delay(40);
  if  (mySerial.read() != 79) digitalWrite(13,1);// if first letter is not "O" from HC-12 feedback "OK+B9600"
  //Serial.println("Error setting speed of 9600 bps ");
  else digitalWrite(13,0);
  // Serial.println("Uart speed is 9600 ");
  for (int i=0;i<12;i++) mySerial.read(); //just read 12 bytes to flush out RX buffer
  delay(40);
  mySerial.print("AT+C001");
    delay(40);
  if  (mySerial.read() != 79) digitalWrite(13,1);// if first letter is not "O" from HC-12 feedback "OK+C001"
  //Serial.println("Error setting channel");
  else digitalWrite(13,0);
  //Serial.println("Channel is 001 - 433.400 MHz ");
  for (int i=0;i<12;i++) mySerial.read(); //just read 12 bytes to flush out RX buffer
  delay(40);
  mySerial.print("AT+P5");
  /* P1=0.8 mW
   * P2=1.6 mW
   * P3=3.2 mW
   * P4=6.3 mW
   * P5=12.6 mW
   * P6=25.1 mW
   * P7=50 mW
   * P8=100 mW
   */
  delay(40);
  if  (mySerial.read() != 79) digitalWrite(13,1);// if first letter is not "O" from HC-12 feedback "OK+P1"
  //Serial.println("Greska kanala");
  else digitalWrite(13,0);
  //Serial.println("Power is 12.6 mW "); //only if P5 is set, your choice.
  //For indoor testing, please set P1 or 0.8 mW, then for field you may test at higher power
  //Pay attention to good antennas on TX, to prevent damage! 0.8 mW is "error free" even with
  //shitty spring antennas, but those antennas are small and handy - just for testings

  for (int i=0;i<12;i++) mySerial.read(); //just read 12 bytes to flush out RX buffer
    delay(80);
   mySerial.print("AT+U8N2");
    delay(80);
  if  (mySerial.read() != 79) //digitalWrite(13,1);// if first letter is not "O" from HC-12 feedback "OK+U8N2"
  Serial.println("Errpr setting 1 startbit, 8 data bits, no parity, and 2 stop bits");
  else digitalWrite(13,0);
  //Serial.println("8 bits, no parity, 2 stop bits ");
  delay(80);
  delay(40);
  digitalWrite(6,1);
  delay(80);
}
```

Kilder:
https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/
http://www.instructables.com/id/Long-Range-18km-Arduino-to-Arduino-Wireless-Commun/
https://www.quora.com/How-can-I-send-data-over-the-HC12-RF-module-using-Arduino

Datablad, PDF: https://arduinoshoppen.dk/help/HC-12_User_Manual.pdf

Link: https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/

https://www.instructables.com/id/Long-Range-18km-Arduino-to-Arduino-Wireless-Commun/

AT-kommandoer

Setup:

We can use the same code for sending AT Commands and configuring the module parameters. All we have to do is connect the "Set" pin of the module to Ground or any digital pin of the Arduino and set the pin to low logic level.

To test whether we have successfully enter the mode, in the serial monitor we can type "AT" and we should get a response message "OK". There are total of 12 AT Commands, and they are used for changing various parameters like the baud rate, the channel, the transmitting power etc. For example, if we type "AT+B38400" the baud rate of the module will be set to 38400.

AT Commands:

1. AT – Test command.

Example: Send "AT" to module, and the module returns "OK".

2. AT+Bxxxx – Change the serial port baud rate.

Available baud rates: 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps, 57600 bps, and 115200 bps. Default: 9600 bps.

Example: Send "AT+B38400" to module, and the module returns "OK+B19200".

3. AT+Cxxxx – Change wireless communication channel, from 001 to 100.

Default: Channel 001, with working frequency of 433.4MHz. Each next channel is 400KHz higher.

Example: If we want to set the module to channel 006, we need to send “AT+C006” command to the module, and the module will return “OK+C006”. The new working frequency will be 435.4MHz.

Code Example.
Here we will use two push buttons for selecting different communication channels and see a different method of storing the incoming data.

Note: The “Set” pins of both HC-12 modules are connected to the pins number 6 of the two Arduinos and the two buttons, at the first Arduino, to the pins 4 and 3.

Arduino-kode: for første Arduino:

```
/* Arduino Long Range Wireless Communication using HC-12
Example 02 - Changing channels using push buttons - Buttons side
by Dejan Nedelkovski, www.HowToMechatronics.com
*/

#include <SoftwareSerial.h>

#define setPin 6
#define button1 4
#define button2 3

SoftwareSerial HC12(10, 11); // HC-12 TX Pin, HC-12 RX Pin

byte incomingByte;
String readBuffer = "";

int button1State = 0;
int button1Pressed = 0;
int button2State = 0;
int button2Pressed = 0;

void setup() {
Serial.begin(9600); // Open serial port to computer
HC12.begin(9600); // Open serial port to HC12
pinMode(setPin, OUTPUT);
pinMode(button1, INPUT);
pinMode(button2, INPUT);
digitalWrite(setPin, HIGH); // HC-12 normal, transparent mode
}

void loop() {
// ==== Storing the incoming data into a String variable
while (HC12.available()) { // If HC-12 has data
incomingByte = HC12.read(); // Store each icoming byte from HC-12
readBuffer += char(incomingByte); // Add each byte to ReadBuffer string variable
}
delay(100);
// ==== Sending data from one HC-12 to another via the Serial Monitor
while (Serial.available()) {
HC12.write(Serial.read());
}

// ==== If button 1 is pressed, set the channel 01
button1State = digitalRead(button1);
if (button1State == HIGH & button1Pressed == LOW) {
button1Pressed = HIGH;
delay(20);
}
if (button1Pressed == HIGH) {
HC12.print("AT+C001"); // Send the AT Command to the other module
delay(100);
//Set AT Command Mode
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(100); // Wait for the HC-12 to enter AT Command mode
HC12.print("AT+C001"); // Send AT Command to HC-12
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
Serial.println("Channel successfully changed");
digitalWrite(setPin, HIGH); // Exit AT Command mode
button1Pressed = LOW;
}

// ==== If button 2 is pressed, set the channel 02
button2State = digitalRead(button2);
if (button2State == HIGH & button2Pressed == LOW) {
button2Pressed = HIGH;
delay(100);
}
if (button2Pressed == HIGH) {
HC12.print("AT+C002"); // Send the AT Command to the other module
delay(100);
//Set AT Command Mode
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(100); // Wait for the HC-12 to enter AT Command mode
HC12.print("AT+C002"); // Send AT Command to HC-12
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
Serial.println("Channel successfully changed");
digitalWrite(setPin, HIGH);
button2Pressed = LOW;
}
checkATCommand();
readBuffer = ""; // Clear readBuffer
}
// ==== Custom function - Check whether we have received an AT Command via the Serial Monitor
void checkATCommand () {
```

```
if (readBuffer.startsWith("AT")) { // Check whether the String starts with "AT"
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(200); // Wait for the HC-12 to enter AT Command mode
HC12.print(readBuffer); // Send AT Command to HC-12
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
digitalWrite(setPin, HIGH); // Exit AT Command mode
}
}
```

2. arduino:

```
/* Arduino Long Range Wireless Communication using HC-12
Example 02 - Changing channels using push buttons
by Dejan Nedelkovski, www.HowToMechatronics.com
*/

#include <SoftwareSerial.h>

#define setPin 6

SoftwareSerial HC12(10, 11); // HC-12 TX Pin, HC-12 RX Pin

byte incomingByte;
String readBuffer = "";

void setup() {
Serial.begin(9600); // Open serial port to computer
HC12.begin(9600); // Open serial port to HC12
pinMode(setPin, OUTPUT);
digitalWrite(setPin, HIGH); // HC-12 normal mode
}

void loop() {
// ==== Storing the incoming data into a String variable
while (HC12.available()) { // If HC-12 has data
incomingByte = HC12.read(); // Store each icoming byte from HC-12
readBuffer += char(incomingByte); // Add each byte to ReadBuffer string variable
}
delay(100);
// ==== Sending data from one HC-12 to another via the Serial Monitor
while (Serial.available()) {
HC12.write(Serial.read());
}
// === If button 1 is pressed, set channel 01
if (readBuffer == "AT+C001") {
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(100); // Wait for the HC-12 to enter AT Command mode
HC12.print(readBuffer); // Send AT Command to HC-12 ("AT+C001")
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
Serial.println("Channel successfully changed");
digitalWrite(setPin, HIGH); // Exit AT Command mode
readBuffer = "";
}
// === If button 2 is pressed, set channel 02
if (readBuffer == "AT+C002") {
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(100); // Wait for the HC-12 to enter AT Command mode
HC12.print(readBuffer); // Send AT Command to HC-12
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
Serial.println("Channel successfully changed");
digitalWrite(setPin, HIGH); // Exit AT Command mode

readBuffer = "";
}
checkATCommand();
readBuffer = ""; // Clear readBuffer
}
// ==== Custom function - Check whether we have received an AT Command via the Serial Monitor
void checkATCommand () {
if (readBuffer.startsWith("AT")) { // Check whether the String starts with "AT"
digitalWrite(setPin, LOW); // Set HC-12 into AT Command mode
delay(100); // Wait for the HC-12 to enter AT Command mode
HC12.print(readBuffer); // Send AT Command to HC-12
delay(200);
while (HC12.available()) { // If HC-12 has data (the AT Command response)
Serial.write(HC12.read()); // Send the data to Serial monitor
}
digitalWrite(setPin, HIGH); // Exit AT Command mode
}
}
```

fra: http://www.14core.com/wiring-the-hc11-hc12-434433mhz-transceiver/

1. AT Test command, Example:
Send “AT” to module
Return: “OK”

2. AT+B
Change the serial port baud rate, and it will be valid after exiting from AT mode. The baud rate can be set to: 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200. The default value is 9600.

Example:
Send "AT+B19200" to module to set the serial port baud rate to be 19200.
Return: "OK+B19200"

3. AT+C
Change the wireless communication channel of module, optional from 001 to 127, and the default value is 001, and the working frequency is 433.4MHz. The channel stepping is 400KHz, and the working frequency of Channel 100 is 473.0MHz.

Example:
Send "AT+C021" to module to set module channel to be 021 working frequency is 441.4MHz
Return: "COK+C021"

Note: As the wireless receiving sensitivity of module is relatively high, when the serial port baud rate is greater than 9600bps, five adjacent channels shall be staggered to use. When the serial port baud rate is not greater than 9600bps, in short-distance (within 10m) communication, again, five adjacent channels shall be staggered to use.

4. AT+FUx
Set module to wireless UART function. The value of x is optional within 1~ 3. The default mode of module is FU3, and only when serial port function mode of two modules is set to be the same the communication can be realized.

F means function, and U means UART

Example:
Send "AT+FU1" to module configuring module to be come UART FU1.
Return: "AT+OK"

5. AT+Px
Set transmitting power of module, x is optional from 1 to 8, and the corresponding transmitting power of module.

X Value 1 - Transmit Power (dBm) -1
X Value 2 - Transmit Power (dBm) 2
X Value 3 - Transmit Power (dBm) 5
X Value 4 - Transmit Power (dBm) 8
X Value 5 - Transmit Power (dBm) 11
X Value 6 - Transmit Power (dBm) 14
X Value 7 - Transmit Power (dBm) 17
X Value 8 - Transmit Power (dBm) 20

6. AT+Ry
Obtain module parameters, y is any letter among B, C, F and P, respectively representing: baud rate, channel, serial port transparent transmission function mode and transmitting power.

Example1:
Send “AT+RB” to module
Return: “OK+B9600”

Example2:
Send “AT+RC” to module
Return: “OK+RC001”

7. AT+RX
Obtain all common parameters of module. Return serial port transparent transmission function mode, baud rate, channel, and transmitting power in order.

Example:
Send “AT+RX” to module
Return: “OK+FU3\r\n OK+B9600\r\n OK+C001\r\n OK+RP:+20dBm\r\n”

8. 12. AT+Uxyz
Set data bits, check bit and stop bit of serial port communication. Where x is data bit, y is parity check, z is stop bit.

Option for parity check:
N: No check
O: odd
E: even

Option for stop bit
1: one stop bit
2: two stop bits
3: 1.5 stop bits

Example:
To send serial port format to be eight data bits,odd parity check, and one stop bit
Send “AT+U8O1” to module
Return: “OK+U8O1”

9. AT+V
Inquire firmware version information of module

Example:
Send “AT+V” to module
Return: “HC-12_V1.1”

10. AT+SLEEP

After receiving the command, the module enters sleep mode after exiting from AT mode, and this mode doesn't allow serial port data transmission. When the module enter AT mode again, the module will exit from sleep mode automatically.

Example
Send "AT+SLEEP" to module when it's not needed to transmit data, to save power.
Return: "OK+SLEEP"
11. AT+DEFAULT
Set serial port baud rate, communication channel, and serial port transparent transmission mode to default value.

Example
Send "AT+DEFAULT" to module
Returns: "OK+DEFAULT" and the default value is restored.

12. AT+UPDATE
Put the module in the status of waiting for software update.
After sending the command, the module will not respond to command anymore until it is re-energized.

Beskrivelse af koden: se: https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/

Kilder: https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/