

Arduino, ESP8266 & Raspberry Pi stuff

Arduino and related stuff (including Attiny and ESP8266) and the Raspberry Pi

Tag: manchester code

Coil loaded 433 MHz Antenna

In my 433 MHz projects I have been using a cheap (1 euro) pair of Tx/Rx modules. I have mostly used the transmitter and that is actually fairly good: if used to trigger commercially available remote switches, it reaches reasonably far with just a simple $1/4$ lambda antenna.

The receiver however is a bit crappy: without antenna the reach is maybe no further than a meter, but even with a $1/4$ lambda antenna it is marginally more, even with free Line of Sight.

For any serious project that involved receiving data it seemed I needed the much better (and more expensive) RXB8 receiver.

However, when mining the internet for a coil antenna (trying to improve on the lengthy 17.2 cm stick antenna) I came across a [design of Ben Schueler](#), apparently once published in elektor magazine.

It is a so called coil loaded design consisting of 0.6mm wire wrapped around a 2.5mm core. Ben's pdf gives a good description with pictures.



The results with this antenna are very good. The distance (with the cheap receiver) that can be covered easily goes to 25 m with line of sight, but also in-house the distance will be increased reaching other rooms with concrete walls in between.

Antenna in use

Advertisements




`aHR0cDovL3dwLm11L2N3LWI2NA==`



`aHR0cDovL3dwLm11L2N3LWI2NA==`

[Report this ad](#)

[Report this ad](#)

July 25, 2015 / Uncategorized / 433 MHz, KAKU, manchester code, RF / 20 Comments

Sending wireless data from one Attiny85 to another

I have struggled a lot with sending RF data between two Attiny85 chips, so I thought it might be helpful if I just explain how I did it. There are a number of such projects being described on the internet, but yet, it didn't prove to be such a straightforward project, which i found to be mostly due to not using the right libraries and cores.

BOM

Transmitter:

Attiny85 – [0.82 euro/10](#)

10k resistor

433 MHz transmittermodule – [1 euro per set](#)

mini breadboard – [58 cts/piece](#)

Receiver:

Attiny85

10k resistor

433 MHz Receiver module

mini breadboard

Optional: [2 wire LCD](#)

There are two main libraries used to send and receive data on an Arduino platform: VirtualWire and Manchester code.

As Virtualwire is a bit older, no longer supported, and supposedly can only send characters (though there is a way around that) I decided to use Manchester code.

To make a long story short, it didn't work. I had the MANCHESTER.h and MANCHESTER.cpp file and ran into a lot of trouble, until I discovered that was the wrong/old library, you need the Manchester.h and Manchester.cpp file [from here](#). When I used that I got my transmitter to work, I could send data to an Arduino, that was already a big relief.

However..... whatever I did, I did not get my receiver to work. In testing something on an Attiny it is very frustrating to just stare at an LED that is supposed to light, but doesnt, without knowing what and why and how. So i decided to add an LCD to the Attiny, so at least I could see what was happening..

However, the LCD on my attiny gave me other problems... and when I solved those, that proved to be the solution for my receive problem as well: I was using the wrong core. I was using the 'tiny core' rather than the 'attiny core' The latter is the one from [Highlowtech](#).

NOTE: it is generally accepted that the '[tiny core](#)' works with the Manchester code and the [attiny](#) core does not, so it is possible that I mixed up the two.

However, I had a line added to the Attiny core that I forgot about that will make it work with the Manchester code. Open up the

"variants/tiny8/pins_arduino.h" file and add the line" #define
__AVR__ Attinyx5__"

The build of the transmitter is easy:

Plug the attiny chip into your breadboard,

Connect a 433 MHz Transmitter module with its data in to pin PBo (that is pin 5 on the chip).

Connect Vcc and Ground of the transmitter module to Vcc (pin 8) and ground (pin 4) of the Attiny

Insert a 10 k resistor between pin 1 (Reset) and pin 8 (Vcc)

Connect Vcc and ground to 5 Volt

Take a 17 cm stiff wire and attach that to the antenna hole of the transmitter.
use the following program:

```
#include <Manchester.h>
/*
  Manchester Transmitter example
  In this example transmitter will send one 16 bit number
  per transmission.
  Try different speeds using these constants, your maximum
  possible speed will depend on various factors like transmitter
  type, distance, microcontroller speed, ...
  MAN_300 0
  MAN_600 1
  MAN_1200 2
  MAN_2400 3
  MAN_4800 4
  MAN_9600 5
  MAN_19200 6
  MAN_38400 7
*/
#define TX_PIN 0 //pin where your transmitter is connected
uint16_t transmit_data = 2761;
void setup() {
  man.setupTransmit(TX_PIN, MAN_1200);
}
void loop() {
  man.transmit(transmit_data);
  delay(200);
}
```

Just a word on the 2716 that I am sending. The program is not mine, I found it as such and since it worked and I was very happy to see the '2716' appear in my Arduino after days of fruitless trying, I decided to leave it there as a tribute. ([it might have found it here](#))

Building the receiver is easy:

Put the programmed attiny chip in your breadboard.

Connect a 10 k resistor between pin 1 and pin 8

Put your 433 MHz Receiver module in the breadboard

Connect the datapin (usually either one of the two middle pins) to pin PB1 (physical pin6) on the attiny.

Connect the Vcc and Ground of the transmitter module to Vcc (pin 8) and Ground (pin4) of the Attiny

Connect the LCD interface to pin PBo (pin 5) (Clock) and pin PB2 (pin 7) (Data/Enable)

Connect Vcc and ground of the LCD to Vcc and Ground.

Attach a 17 cm ($1/4$ lambda for 433 MHz) to the Receiver module.

Use the following program in your chip:

```
#include <Manchester.h>
#include <LiquidCrystal_SR.h>
LiquidCrystal_SR lcd(0,2,TWO_WIRE);
/*
Manchester Receiver for Attiny
In this example receiver will receive one 16 bit number per
transmission to switch a relay on or off.
try different speeds using these constants, your maximum possible
speed will depend on various factors like transmitter type,
distance, microcontroller speed, ...

MAN_300 0
MAN_600 1
MAN_1200 2
MAN_2400 3
MAN_4800 4
MAN_9600 5
```

```

    MAN_19200 6
    MAN_38400 7
*/
#define RX_PIN 1 // = pin 6
uint8_t moo = 1;
void setup()
{
  lcd.begin(16,2);
  lcd.home();
  lcd.print("Received");
  man.setCursor(0,1);
  man.setupReceive(RX_PIN, MAN_1200);
  man.beginReceive();
}
void loop() {
  if (man.receiveComplete()) {
    uint16_t m = man.getMessage();
    man.beginReceive(); //start listening for next message right
                        //after you retrieve the message

    moo = ++moo % 2;
    lcd.print(m);
  }
}

```

The LCD is ofcourse optional. You can use an LED that gets the variable 'moo' send to its pin and thus will flash on complete receival of the signal, but I just wanted to make sure that what I got was what I sent

The results are good, no garbage is being received, but the reach is about 4-5 meters, with the antennas. The antenna on the receiver only makes a small difference. The one on the Transmitter makes a big difference. Still, this is surprising as the transmitter module is capable of switching remote switches at greater distances even on different floors.

With regard to the length of the Antennas:

as $f * \lambda = c$ (frequency * wavelength = lightspeed)

$\lambda = c / f$

$\lambda = 299,792,458 / 433,920,000$

The wavelength is 0.690893386 meters.

Antenna would be $\lambda/4 = 0.172723346$ meters (17.3 cm)

That is about 6.80013 inches.

If you would be using 315 MHz modules, the antenna would be: 0.238 m or 23.8 cm

You can also use the [wavelength calculator](#).

Supposedly the transmitter module can take 12 Volts and still be steered by a 5 Volt microcontroller pin and will have a further reach then. Increasing the voltage on the receiver of course makes no difference

Once you have established The link between two attiny's, linking one of them with an arduino should not be a problem. I use mine to send data to an Arduino (e,g. temperature, or the status of a tripwire), or to receive data from an Arduino to steer a servo or an RGB LED.

If you want to trigger a relay, you would do it like this:

```
void loop() {  
  if (man.receiveComplete()) {  
    uint16_t m = man.getMessage();  
    man.beginReceive(); //start listening for next message  
                        //right after you retrieve the message  
  
    moo = ++moo % 2;  
    lcd.print(m);  
    if (m==2761){digitalWrite(Relay,HIGH);}  
    if (m==0000){digitalWrite(Relay,LOW);}  
  }  
}
```

Ofcourse you need to define the Relay pin in the 'setup'

ADVERTISEMENT





March 29, 2015 / 433MHz, Attiny / 433 MHz, arduino, Attiny85, manchester code, RF / 42
Comments

Arduino, ESP8266 & Raspberry Pi stuff / Create a free website or blog at WordPress.com.