Mchr3k - Arduino Manchester Encoding

This is a Manchester encoding RF library which works on Arduino and ATTiny.

Library supports:

- various microcontrollers
  - ATmega328
  - ATmega8
  - ATMega32U4
  - ATtiny84 (44, 24)
  - ATtiny85 (45, 85)
- various microcontroller's speed both RX and TX
  - 1Mhz (see notes for TX)
  - 8Mhz
  - 16Mhz
  - 16.5Mhz ATtiny85
- various transmission speeds
  - 300baud
  - 600baud
  - 1200baud
  - 2400baud
  - 4800baud
  - 9600baud
  - 19200baud

With this library you can sucessfully transmit data between various microcontrollers runnning at various speeds even if their clock speed varies up to 100%. Works great with innacurate internal oscilator. Your maximum transmission speed will depend on many factors like transmitter type, antenna, distance, enviroment noise, microcontroller type and speed

Getting started on Arduino

1. Download MANCHESTER.cpp and MANCHESTER.h and save these in the following location

   ```
   workspace_root/libraries/Manchester
   ```

2. Import the library from your Arduino sketch

   ```
   #include <Manchester.h>
   ```

3. Modify your setup() method.

   For sending data

   ```
   void setup()
   {
     man.setupTransmit(TX_PIN, MAN_1200);
   }
   ```

   For receiving data

```
void setup()
{
  man.setupReceive(RX_PIN, MAN_1200);
  man.beginReceive();
}
```

4. 4) Modify your main loop

   For sending data

   ```
   void loop() {
     man.transmit(transmit_data);
   }
   ```

   For receiving data

   ```
   void loop() {
     if (man.receiveComplete()) {
       uint16_t m = man.getMessage();
       man.beginReceive(); //start listening for next me
       //do something with your message here
     }
   }
   ```

5. For more advanced uses like transmitting with checksum and sender/receiver address or sending array of bytes see attached examples

Getting started on ATtiny85

This is broadly the same as on Arduino with one extra step. You need to download the ATtiny hardware files (a list of alternative ATtiny implementations is listed on this site which is where I got the files in tiny.zip). These should be unzipped into the following location.

```
workspace_root/hardware/tiny
```

As of now (June 2013) most of tiny cores does not support micro delay at 1Mhz and for that reason if you want to TRANSMIT (receive is OK) from your 1Mhz ATtiny85/84 you need to call this method before you call transmit setup:

```
man.workAround1MhzTinyCore(); //add this in order for trans
man.setupTransmit(TX_PIN, MAN_1200);
```

Notes about transmitters

The library has been tested with common 315Mhz and 433Mhz transmitters using ASK OOK. Tips to improve your transmit distance: Attaching an antenna will greatly improve transmit distance (from few cm to few m) for 315Mhz use 23.81 cm straight wire, for 433Mhz use 17.28cm straight wire as antenna. If possible keep the wire straight, away from ground wire, and away from the rest of the circuit and battery pack. Transmitter can use anything from 3.3V to 12V. Increasing voltage will increase transmit distance. If you are using voltage regulator, attach transmitter directly to the battery. Receiver needs 5V, it doesn't work on 3.3V

Speed: I was able to achieve 9600 bauds between two 16Mhz Arduinos, or 2400 bauds between two 1Mhz ATTiny85, you can try different speeds to see which works the best for you.

Full duplex: for bidirectional communication use both 315Mhz and 433Mhz transmitters. This way they can transmit at the same time without interfering with each other. If you have just one type, wait for receiver to finish receiving before transmitting.

Credits

- Library originally from carl47 on this thread
- Contributions from mchr3k, Mike and Cano

Blog posts

I used this library in a series of blog posts where I put together a battery powered temperature sensor:

- Wireless sensor node (part 1) - All on one breadboard
- Wireless sensor node (part 2) - Rf link working
- Wireless sensor node (part 3) - Manchester lib
- Wireless sensor node (part 4) - code fixes
- Wireless sensor node (part 5) - Sleeping
- Wireless sensor node (part 6) - Turning off subsystems
- Wireless sensor node (part 7) - Background rx
- Wireless sensor node (part 8) - New hardware
- Wireless sensor node (part 9) - Watch battery
- Wireless sensor node (part 10) - Prototype node
- Wireless sensor node (part 11) - Enclosures
- Wireless sensor node (part 12) - Enclosure mark 1
- Wireless sensor node (part 13) - Getting started with an SD card
- Wireless sensor node (part 14) - Enclosure mark 2
- Wireless sensor node (part 15) - Enclosure mark 2 (follow up)
- Wireless sensor node (part 16) - Software