MorseEnDecoder

MorseEnDecoder.h is a library for encoding and decoding morse. The encode can be done at any programmable WPM, but the decode only works on a known, fixed WPM. See code.google.com/p/morse-endecoder/ for details.

There are two classes:

```
    morseDecoder
    morseEncoder
```

The decoder is not described here as it useless with only a fixed WPM rate...

```
morseEnDecoder morseObj(OUTPIN); // create morse object
morseOut.encode(); // call once per loop
morseOut.available(); // true if available for encoding
morseOut.write(letter); // output char letter
```

Here’s the code for the encoder sending data from the serial console:

```
#include <MorseEnDecoder.h?

#define OUTPIN 3
#define WPM 10

// create a morse object
morseEncoder morseOut(OUTPIN);

void setup()
{
  pinMode(OUTPIN, OUTPUT); // output pin

  morseOut.setspeed(WPM); // set output speed

  Serial.begin(): // string to output will be sent from console of IDE
}

void loop()
{
  char letter;

  morseOut.encode(); // must be called once every loop

  if(Serial.available()) && morseOut.available()) // if ports available
  {
    letter = Serial.read(); // read an input letter
    morseOut.write(letter); // output the morse code H/L level pulses on OUTPIN
  }
}
```

The output has to drive a replay to key the transmitter, and an oscillator to hear the audio signal. See `ganymedeT.blogspot.com`

From the author...

Instantiating a Morse decoder or encoder object

For the Morse decoder

morseDecoder objectname(input pin nr, input type, input logic);

Where

- objectname is a name for the morse decoder object
- input pin nr. is which Arduino pin used for Morse signal input
- input type is:
  - MORSE_KEYER - for a digital input (like a push button).
  - MORSE_AUDIO - for an analog input (Morse audio signal).
- input logic is:
  - MORSE_ACTIVE_LOW - will also use the internal pull-up resistor.
  - MORSE_ACTIVE_HIGH - for "normal logic" signals.

Note The input pin nr differs between each type of input, IE digital 3 is not the same as analog 3.

For the Morse encoder

morseEncoder objectname(output pin nr);

- It only needs one argument, which Arduino pin to use as an output.

Functions or methods

Morse Decoder

- objectname.decode();
  - Returns nothing. This needs to be called at least once every main loop in the decoding process.
- objectname.available();
  - Returns boolean true or false, depending on whether or not a Morse code character is decoded (received).
- objectname.read();
  - Returns a char, the last morse code character received and decoded.

    This only contains the last character decoded, until it is read.

- objectname.setspeed();
    - Returns nothing. Sets the Morse speed in wpm (words per minute). If this method is not called, a default wpm of 13 is used.

Members

- objectname.debounceDelay
    - Sets or returns the debounce timer for the digital input signal. Keep well below the time for a Morse dot (dot time in ms = 1200/wpm).
- objectname.AudioThreshold
    - Sets or returns the threshold used for analog input. It is a simple clipping filter setting.
- obectname.morseSignalState
    - Returns (or sets, but of no use) the current state of the input Morse signal.

Morse Encoder

- objectname.encode();
    - Returns nothing. This needs to be called at least once every main loop during the encoding process.
- objectname.available();
    - Returns boolean true or false, depending on whether or not it is ready to receive the next Morse code character to encode and send.
- objectname.write();
    - Returns nothing. Sets the next character to encode and send. The output pin is then toggled in Morse code (if subsequent calls to the .encode method is used).
- objectname.setspeed();
    - Returns nothing. Sets the Morse speed in wpm (words per minute). If this method is not called, a default wpm of 13 is used.

Members

- objectname.morseSignalString
    - Returns a C-type string with the encoded Morse signals as ASCII dots or dashes, for sending Morse.
    - Note that this string will be destroyed while sending the Morse, so if you want to use it, you must read it at once after sending a character with the write() method.

Example

This example takes a Morse signal on digital input pin 7, decodes it and write the text to the serial output. The input is active-low, hence it will also use the internal pull-up resistor.

It also takes text received on the serial interface, and encodes it to the digital output pin nr. 13. It also shows the letters and morse code back on the serial output.

If no morse code is being sent, it also use the output pin to show the debounced input.

This example also comes with the library.

```
#include "MorseEnDecoder.h"


int morseInPin = 7;
int morseOutPin = 13;

morseDecoder morseInput(morseInPin, MORSE_KEYER, MORSE_ACTIVE_LOW);
morseEncoder morseOutput(morseOutPin);


void setup()
{
  Serial.begin(9600);
  Serial.println("Morse EnDecoder demo");

  // Setting Morse speed in wpm - words per minute
  // If not set, 13 wpm is default
  morseInput.setspeed(13);
  morseOutput.setspeed(13);
}



void loop()
{
  // Needs to call these once per loop
  morseInput.decode();
  morseOutput.encode();


  // Morse output:
  // Encoding text received from the serial input
  if (Serial.available() && morseOutput.available())
  {
    char sendMorse = Serial.read();
    morseOutput.write(sendMorse);
    // Also write it back to serial
    morseOutput.encode();    // This is just to get
morseSignalString before it is destroyed
    Serial.write(' ');
    Serial.write(sendMorse);
    Serial.write(morseOutput.morseSignalString);
  }

  // Morse input:
  // If a character is decoded from the input, write it to serial
output
  if (morseInput.available())
  {
    char receivedMorse = morseInput.read();
    Serial.print(receivedMorse);

    // A little error checking
    if (receivedMorse == '#') Serial.println("< ERROR:too many morse
signals! >");
  }


  // Morse feedback from input if not sending Morse.
  if (morseOutput.available()) digitalWrite(morseOutPin,
morseInput.morseSignalState);
}
```