`TN.h` — Arduino Library for Tangible Networks TN-04

Espen Knoop

This is an Arduino library for use with Tangible Network TN-04 nodes. It handles inputs, outputs, reading switches etc. For serial communication, use the standard Arduino functions.

The library defines a class `TN`, so using it requires creating a `TN` object and calling its methods. Here is a minimal example of use:

```
#include<TN.h> // Requires TN.h, TN.cpp, Keywords.txt in folder <Arduino>/Libraries/TN/
TN Tn = TN(-1.0,1.0); // Create TN object with range [-1.0, 1.0]
void setup () {} // Don't need anything in here - inputs/outputs set up in constructor
void loop () {
  Tn.colour(255,255,255); // Set LED to white
  delay(500);
  Tn.colour(0,0,0); // Set LED to off
  delay(500);
}
```

Nodes have 3 inputs and 3 outputs. Inputs/outputs are numbered 1-3 so as to be in keeping with labels on the PCB/units. Nodes also have a pot (potentiometer, knob), a pushbutton switch and 3 DIP configuration switches that can be switched with a small screwdriver or similar.

Most models will probably use `analogRead()` and `analogWrite()`, but `digitalRead()` and `digitalWrite()` are also provided for models requiring only binary information (on/off) to be sent between the nodes.

The library defines the following methods:

`TN(double minVal=0.0, double maxVal=1.0)`	Constructor for `TN` object. Input arguments specify range of `analogRead()` and `analogWrite()`: values outside range will be clipped. If arguments are not specified, range is set to $[0.0, 1.0]$.
`void colour(int r, int g, int b)`	Set LED colour. Integer arguments $\in [0, 255]$.
`void colour(double r, double g, double b)`	Set LED colour. double arguments $\in [0.0, 1.0]$
`boolean isConnected(int input)`	Returns `true` if input is connected, `false` otherwise.
`double analogRead(int input)`	Read the value of an input. Returns `minVal` if input is not connected.
`void analogWrite(int output, double value)`	Write a value to an output. Value is clipped if outside `[minVal,maxVal]` range.
`int digitalRead(int input)`	Read the value of an input as `true` or `false`. Only use in conjunction with `digitalWrite()`.
`void digitalWrite(int output, int value)`	Write an output to `true` (`maxVal`) or `false` (`minVal`).
`boolean dip1()`	Get state of DIP switch 1 (`true` is on).
`boolean dip2()`	Get state of DIP switch 2 (`true` is on).
`boolean dip3()`	Get state of DIP switch 3 (`true` is on).
`boolean sw()`	Get state of pushbutton switch (`true` is pressed).
`double pot()`	Get position of pot. Returns `double` between 0.0 (fully CCW) and 1.0 (fully CW).
`boolean masterConnected()`	Returns `true` if master controller is connected.
`double masterRead()`	Get value of master controller. Returns `double` between 0.0 (fully CCW) and 1.0 (fully CW). Returns 0.0 if master controller is not connected.
`boolean masterSw()`	Returns true if master switch is pressed. Returns false if master switch not connected. Do not combine with `masterRead()`.
`void printState()`	For debugging. Prints out the current state (ins, outs, switches etc) to serial. Requires `Serial.begin(115200)` in `setup()`. Runs in approx. 5 ms.

The libary also includes fast functions for max and min, `MAX(x,y)` and `MIN(x,y)`, as well as `MINMAX(x,l,u)` which returns x if $l < x < u$, l if $x < l$ and u if $u < x$.